Surrogate-Assisted Evolutionary Algorithms
Ilya Loshchilov

To cite this version:

HAL Id: tel-00823882
https://tel.archives-ouvertes.fr/tel-00823882
Submitted on 18 May 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.
SURROGATE-ASSISTED
EVOLUTIONARY ALGORITHMS

PH.D. THESIS
ÉCOLE DOCTORALE D’INFORMATIQUE, ED 427
UNIVERSITÉ PARIS-SUD 11

By Ilya Gennadyevich LOSHCHILOV

Presented and publicly defended:

On January 8 2013
In Orsay, France

With the following jury:

Frédéric Bonnans, DR INRIA, INRIA Saclay, Ecole Polytechnique, France (Examiner)
Michael Emmerich, Assistant Professor, Leiden University, The Netherlands (Examiner)
Christian Igel, Professor, University of Copenhagen, Denmark (Reviewer)
Una-May O’Reilly, Principal Research Scientist, MIT CSAIL, USA (Examiner)
Marc Schoenauer, DR INRIA, INRIA Saclay, LRI/TAO, France (Thesis Advisor)
Michèle Sebag, DR CNRS, University of Paris-Sud 11, LRI/TAO, France (Thesis Advisor)
François Yvon, Professor, University of Paris-Sud 11, LIMSI/TLP, France (Examiner)

Reviewers:
Christian Igel, Professor, University of Copenhagen, Denmark
Yaochu Jin, Professor, University of Surrey, UK

Inria Saclay - Île-de-France – TAO Project Team
1 rue Honoré d’Estienne d’Orves 91120 Palaiseau Cedex, France
LES ALGORITHMES ÉVOLUTIONNAIRES À LA BASE DE MÉTA-MODELES SCALAIRES

THÈSE DE DOCTORAT
ÉCOLE DOCTORALE D’INFORMATIQUE, ED 427
UNIVERSITÉ PARIS-SUD 11

Par Ilya Gennadyevich LOSCHCHILOV

Présentée et soutenue publiquement

Le 8 Janvier 2013
À Orsay, France

Devant le jury ci-dessous :

Frédéric BONNANS, DR INRIA, INRIA Saclay, Ecole Polytechnique, France (Examinateur)
Michael EMMERICH, Assistant Professor, Leiden University, The Netherlands (Examinateur)
Christian IGEI, Professor, University of Copenhagen, Denmark (Rapporteur)
Una-May O’REILLY, Principal Research Scientist, MIT CSAIL, USA (Examinatrice)
Marc SCHOENAUER, DR INRIA, INRIA Saclay, LRI/TAO, France (Directeur de Thèse)
Michèle SEBAG, DR CNRS, University of Paris-Sud 11, LRI/TAO, France (Directrice de Thèse)
François YVON, Professor, University of Paris-Sud 11, LIMSI/TLP, France (Examinateur)

Rapporteurs :

Christian IGEI, Professor, University of Copenhagen, Denmark
Yaochu JIN, Professor, University of Surrey, UK

Inria Saclay - Île-de-France – TAO Project Team
1 rue Honoré d’Estienne d’Orves 91120 Palaiseau Cedex, France
Abstract

Evolutionary Algorithms (EAs) have received a lot of attention regarding their potential to solve complex optimization problems using problem-specific variation operators. A search directed by a population of candidate solutions is quite robust with respect to a moderate noise and multi-modality of the optimized function, in contrast to some classical optimization methods such as quasi-Newton methods. The main limitation of EAs, the large number of function evaluations required, prevents from using EAs on computationally expensive problems, where one evaluation takes much longer than 1 second.

The present thesis focuses on an evolutionary algorithm, Covariance Matrix Adaptation Evolution Strategy (CMA-ES), which has become a standard powerful tool for continuous black-box optimization. We present several state-of-the-art algorithms, derived from CMA-ES, for solving single- and multi-objective black-box optimization problems.

First, in order to deal with expensive optimization, we propose to use comparison-based surrogate (approximation) models of the optimized function, which do not exploit function values of candidate solutions, but only their quality-based ranking. The resulting self-adaptive surrogate-assisted CMA-ES represents a tight coupling of statistical machine learning and CMA-ES, where a surrogate model is build, taking advantage of the function topology given by the covariance matrix adapted by CMA-ES. This allows to preserve two key invariance properties of CMA-ES: invariance with respect to i). monotonous transformation of the function, and ii). orthogonal transformation of the search space. For multi-objective optimization we propose two mono-surrogate approaches: i). a mixed variant of One Class Support Vector Machine (SVM) for dominated points and Regression SVM for non-dominated points; ii). Ranking SVM for preference learning of candidate solutions in the multi-objective space. We further integrate these two approaches into multi-objective CMA-ES (MO-CMA-ES) and discuss aspects of surrogate-model exploitation.

Second, we introduce and discuss various algorithms, developed to understand, explore and expand frontiers of the Evolutionary Computation domain, and CMA-ES in particular. We introduce linear time Adaptive Coordinate Descent method for non-linear optimization, which inherits a CMA-like procedure of adaptation of an appropriate coordinate system without losing the initial simplicity of Coordinate Descent. For multi-modal optimization we propose to adaptively select the most suitable regime of restarts of CMA-ES and introduce corresponding alternative restart strategies. For multi-objective optimization we analyze case studies, where original parent selection procedures of MO-CMA-ES are inefficient, and introduce reward-based parent selection strategies, focused on a comparative success of generated solutions.
Les Algorithmes Évolutionnaires (AEs) ont été très étudiés en raison de leur capacité à résoudre des problèmes d’optimisation complexes en utilisant des opérateurs de variation adaptés à des problèmes spécifiques. Une recherche dirigée par une population de solutions offre une bonne robustesse par rapport à un bruit modéré et la multi-modalité de la fonction optimisée, contrairement à d’autres méthodes d’optimisation classiques telles que les méthodes de quasi-Newton. La principale limitation de AEs, le grand nombre d’évaluations de la fonction objectif, pénalise toutefois l’usage des AEs pour l’optimisation de fonctions chères en temps calcul.

La présente thèse se concentre sur un algorithme évolutif, Covariance Matrix Adaptation Evolution Strategy (CMA-ES), connu comme un algorithme puissant pour l’optimisation continue boîte noire. Nous présentons l’état de l’art des algorithmes, dérivés de CMA-ES, pour résoudre les problèmes d’optimisation mono- et multi-objectifs dans le scénario boîte noire.

Une première contribution, visant l’optimisation de fonctions coûteuses, concerne l’approximation scalaire de la fonction objectif. Le meta-modèle appris respecte l’ordre des solutions (induit par la valeur de la fonction objectif pour ces solutions) ; il est ainsi invariant par transformation monotone de la fonction objectif. L’algorithme ainsi défini, saACM-ES, intègre étroitement l’optimisation réalisée par CMA-ES et l’apprentissage statistique de meta-modèles adaptatifs ; en particulier les meta-modèles reposent sur la matrice de covariance adaptée par CMA-ES. saACM-ES préserve ainsi les deux propriétés clé d’invariance de CMA-ES : invariance i) par rapport aux transformations monotones de la fonction objectif; et ii) par rapport aux transformations orthogonales de l’espace de recherche.


Acknowledgements

First and foremost I want to thank my advisors, Marc and Michèle, for a rich and friendly scientific environment they created. With a greatest pleasure I will remember all the adventures of thought we had together, some beautiful landscapes that we have noticed are still waiting for us. I will never forget the enormous forces you applied with great talent making me a better researcher.

I would like to use an opportunity to thank my previous advisors: my chess teacher, Vladimir Andreevich Zhukov who showed me that there is the pleasure of thinking, my history teacher, Yury Moiseevich Jarema who taught me to ask right questions, and my first scientific advisor, Leonid Ivanovich Babak who showed me the power of evolution.

I deeply thank Christian Igel and Yaochu Jin for closely reviewing my thesis and for many valuable comments and suggestions. I also would like to thank Frédéric Bonnans, Michael Emmerich, Una-May O’Reilly and François Yvon for kindly accepting to be members of the Jury, it is an honor for me.

I would like to thank all my friends from the TAO team and especially (in alphabetical order): Alvaro, Anne, Dawei, Gaetan, Jacques, Nikolaus, Mostepha, Mouadl, Ouassim, Riad, Weijia, Youhei and Zyed. I am especially grateful to Nikolaus and Anne for sharing their experience and for many fruitful discussions we had. This thesis would not have been possible without Marc, Michèle, Marie-Carol and Valérie who always were around to help me during the thesis.

I also thank all my new and old closest friends: Alexander, Arseniy, Gaetan, Ekaterina, Nikita, Petr, Timur, Valentin, Victor and Yury. The lovely time we spent together will forever stay in my memory.

Finally, I would like to warmly thank my Family: my parents Olga and Gennady, my brother Anton and my wife Anna. You are my happiness and my source of confidence that since we are together, tomorrow will be at least as good as today. When I get up in the morning and I feel myself falling in love, I still can not believe how lucky I am to be with you, my beauty and my treasure, my beloved Anna.
To Anna.
Part III Contributions

4 Single-Objective Surrogate-Assisted CMA-ES

4.1 Ordinal Regression
  4.1.1 Learning to Rank using Ranking SVM
  4.1.2 Choice of Kernel Function
  4.1.3 Ordinal Regression vs Metric Regression
  4.1.4 Ordinal Regression using Metric Regression Approaches
  4.1.5 Ranking SVM vs Rank-based SVR

4.2 ACM with Pre-selection
  4.2.1 Toward Optimal Kernel Function
  4.2.2 Overview of ACM-ES
  4.2.3 Experimental Validation
    4.2.3.1 Results and Discussion
  4.2.4 Preliminary Conclusion

4.3 Self-adaptive Surrogate-assisted CMA-ES
  4.3.1 Preliminary Analysis
    4.3.1.1 Model Quality Estimation
    4.3.1.2 Model Exploitation
4.3.3 Experimental Validation ........................................... 106
  4.3.3.1 Experimental Setting ..................................... 107
  4.3.3.2 Offline Tuning: Number of Training Points ............. 108
  4.3.3.3 Online Tuning: Surrogate Hyper-parameters ............ 109
  4.3.3.4 Comparative Performances ................................ 110
  4.3.3.5 Scalability w.r.t. Population Size ...................... 111
  4.3.3.6 Ranking SVM vs Rank-based SVR ........................ 111
  4.3.3.7 Benchmarking on Noiseless BBOB-2012 Testbed ......... 113
  4.3.3.8 Benchmarking on Noisy BBOB-2012 Testbed .......... 120
  4.3.3.9 CPU Timing Experiment ................................ 124
  4.3.3.10 Quadratic Programming Problem Optimization .......... 124
  4.3.4 Conclusion and Perspectives ................................. 128

4.4 Discussion .......................................................... 129

5 Multi-Objective Surrogate-Assisted CMA-ES ..................... 131
  5.1 Aggregated Surrogate Model ................................... 132
    5.1.1 Rationale and Assumption .............................. 132
    5.1.2 Lagrangian formulation ................................ 133
  5.2 Rank-based Aggregate Surrogate Model ....................... 134
    5.2.1 Learning Preference Information in Multi-objective Space 134
    5.2.2 Dominance-based Preference Learning .................... 135
  5.3 ASM-assisted MOEAs with Pre-selection ......................... 136
    5.3.1 Discussion ................................................ 136
    5.3.2 The Algorithm ............................................ 137
      5.3.2.1 Model Update ..................................... 138
      5.3.2.2 Informed Operators ................................. 140
  5.4 Experimental Validation ......................................... 142
    5.4.1 Experimental Setting .................................... 142
    5.4.2 Performance Measures .................................... 143
    5.4.3 Model Validation ......................................... 143
      5.4.3.1 ASM model ........................................ 143
      5.4.3.2 RASM model ....................................... 145
    5.4.4 Benchmarking of ASM and RASM assisted MOEAs .......... 146
  5.5 Discussion ........................................................ 149

6 Exploring new frontiers using CMA-like algorithms ............. 151
  6.1 Adaptive Coordinate Descent .................................. 152
    6.1.1 Introduction ............................................. 152
    6.1.2 Algorithms ............................................... 153
      6.1.2.1 Adaptive Encoding ................................ 153
6.1.2.2 Coordinate Descent by Dichotomy ........................................ 155
6.1.2.3 Adaptive Coordinate Descent ................................................ 160
6.1.3 Experimental Validation .......................................................... 161
  6.1.3.1 Adaptive Encoding Calibration .............................................. 163
  6.1.3.2 Coordinate Descent Calibration ........................................... 164
  6.1.3.3 Experimental Settings ....................................................... 164
  6.1.3.4 Results on Noiseless BBOB Testbed ...................................... 165
  6.1.3.5 Extension to Noisy Optimization ........................................... 166
  6.1.3.6 Extension to Large-Scale Optimization ................................... 166
6.1.4 Conclusion and Perspectives .................................................... 168
6.2 Reward-based Parent Selection ................................................... 170
  6.2.1 New Parent Selections for Steady-State MO-CMA-ES .................... 171
    6.2.1.1 Tournament Selection .................................................... 171
    6.2.1.2 Multi-Armed Bandit Selection ......................................... 171
    6.2.1.3 Defining Rewards .......................................................... 173
    6.2.1.4 Discussion ................................................................. 176
  6.2.2 Experimental Validation ....................................................... 177
    6.2.2.1 Experimental Setting ...................................................... 177
    6.2.2.2 Constraints Handling in MO-CMA-ES .................................. 178
    6.2.2.3 Result Analysis ............................................................ 179
  6.2.3 Conclusion and Perspectives .................................................. 181
6.3 Alternative Restart Strategies for CMA-ES ..................................... 184
  6.3.1 The CMA-ES with Restarts: IPOP and BIPOP CMA-ES .................... 185
  6.3.2 Preliminary Analysis ............................................................ 185
  6.3.3 Alternative Restart Strategies .............................................. 187
  6.3.4 Experimental Validation ....................................................... 188
    6.3.4.1 Performance on Rastrigin Function .................................... 188
    6.3.4.2 Benchmarking with BBOB Framework .................................. 189
    6.3.4.3 Interplanetary Trajectory Optimization ............................... 190
  6.3.5 Conclusion and Perspectives .................................................. 191
6.4 Other Approaches ................................................................. 192
  6.4.1 Kernel PCA-based CMA-ES ..................................................... 193
  6.4.2 High-precision BFGS ............................................................ 194
  6.4.3 Comparison-based Multi-objective Optimization ......................... 195
6.5 Discussion ..................................................................................... 196

Part IV General Conclusion

7 Conclusion ....................................................................................... 201
  7.1 Summary of Contributions ........................................................ 201
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>7.2</td>
<td>Future Directions</td>
<td>203</td>
</tr>
<tr>
<td>Bibliography</td>
<td></td>
<td>205</td>
</tr>
<tr>
<td>Appendix A</td>
<td></td>
<td>233</td>
</tr>
<tr>
<td>Appendix B</td>
<td></td>
<td>239</td>
</tr>
<tr>
<td>B.1</td>
<td>Dual Form of ASM Learning Problem</td>
<td>239</td>
</tr>
<tr>
<td>B.2</td>
<td>Solving the Dual Problem</td>
<td>240</td>
</tr>
<tr>
<td>B.2.1</td>
<td>The One-dimensional Maximization Problem</td>
<td>241</td>
</tr>
</tbody>
</table>
List of Figures

2.1 Results from CEC-2005 optimization contest .................................. 32
2.2 Results from BBOB-2009 workshop ................................................. 33
2.3 The preference-based and ideal principles of multi-objective optimization .. 40

3.1 Publications reporting the use of the major surrogate techniques .............. 50
3.2 Linear separating hyperplanes for separable case of classification .......... 55
3.3 Linear separating hyperplanes for non-separable case of classification .... 57
3.4 Linear separating hyperplanes of non-linear classification .................... 58
3.5 Support Vector Regression .......................................................... 60
3.6 One-Class Support Vector Machine ............................................... 61
3.7 Ranking Support Vector Machine .................................................. 62

4.1 Polynomial and RBF kernel functions ............................................. 83
4.2 Model error of Ranking SVM for different parameters ......................... 84
4.3 Model error of Ranking SVM with different kernels ......................... 85
4.4 Model error of Ranking SVM and SVR .......................................... 87
4.5 Model error of SVR with different scaling factors ........................... 88
4.6 Model error with different number of training points ....................... 89
4.7 Advantages of covariance matrix based RBF kernel ............................ 92
4.8 Optimization loop of the ACM-ES algorithm with pre-selection ............. 94
4.9 ACM model learning/testing cost ................................................. 97
4.10 Rank-based surrogate error versus number of function evaluations ......... 98
4.11 The speedup of IPOP-aACM-ES over IPOP-aCMA-ES for different \( \hat{n} \) . . 101
4.12 Number of generations \( \hat{n} \) versus surrogate model error Err ............ 102
4.13 Optimization loop of the \( \ast \)ACM-ES .............................................. 105
4.14 The speedup of IPOP-aACM-ES over IPOP-aCMA-ES for different \( \ell \) .. 108
4.15 Ranking SVM model hyper-parameters optimization .......................... 109
4.16 Comparison of the surrogate-assisted versions of IPOP-CMA-ES ......... 110
4.17 Speedup of IPOP-\( \ast \)aACM-ES for large population sizes .................. 112
4.18 SVR model hyper-parameters optimization .................................. 113
4.19 Performance of 45 optimization algorithms on BBOB benchmark problems 115
4.20 BIPOP-\( \ast \)aACM-ES and IPOP-\( \ast \)aACM-ES on noiseless BBOB problems . . 117
4.21 ECDF plots on 20-dimension noiseless functions ............................. 118
4.22 BIPOP-\( \ast \)aACM-ES and IPOP-\( \ast \)aACM-ES on noisy BBOB problems .... 121
4.23 ECDF plots on 20-dimension noisy functions ................................... 122
4.24 CPU cost per function evaluation of IPOP-aACM-ES ....................... 124
4.25 Optimization of the hyper-parameters of the Ranking SVM model ......... 125
4.26 Uniform and gradient-based selection for 10-D Sphere ....................... 126
4.27 Convergence to the optimum of QP problem ................................ 127
### List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>An idealistic schematic view of Aggregated Surrogate Model</td>
<td>133</td>
</tr>
<tr>
<td>5.2</td>
<td>Constraints involved in Rank-based Aggregated Surrogate Models</td>
<td>136</td>
</tr>
<tr>
<td>5.3</td>
<td>Training data selection for ASM models</td>
<td>139</td>
</tr>
<tr>
<td>5.4</td>
<td>Learning time of RASM model</td>
<td>141</td>
</tr>
<tr>
<td>5.5</td>
<td>ASM model for 30-dimensional ZDT1 and IHR1 problems</td>
<td>144</td>
</tr>
<tr>
<td>5.6</td>
<td>RASM model for 30-dimensional ZDT1 problem</td>
<td>145</td>
</tr>
<tr>
<td>5.7</td>
<td>Performance of original and ASM-informed MO-CMA-ES and NSGA-II</td>
<td>147</td>
</tr>
<tr>
<td>6.1</td>
<td>Adaptive Coordinate Descent</td>
<td>154</td>
</tr>
<tr>
<td>6.2</td>
<td>Coordinate Descent on Sphere and Rosenbrock functions</td>
<td>157</td>
</tr>
<tr>
<td>6.3</td>
<td>Convergence rates of several versions of ES and ACiD</td>
<td>159</td>
</tr>
<tr>
<td>6.4</td>
<td>Performance analysis of ACiD in $\alpha_{ecov} \times \alpha_{cpath}$ space</td>
<td>162</td>
</tr>
<tr>
<td>6.5</td>
<td>The performance of ACiD on BBOB noiseless testbed</td>
<td>165</td>
</tr>
<tr>
<td>6.6</td>
<td>ECDF plots of ACiD and CMA-ES algorithms</td>
<td>166</td>
</tr>
<tr>
<td>6.7</td>
<td>Results for ACiD with linear time complexity</td>
<td>167</td>
</tr>
<tr>
<td>6.8</td>
<td>Examples of parent selection in MO-CMA-ES</td>
<td>170</td>
</tr>
<tr>
<td>6.9</td>
<td>Multi-Armed Bandits-based parent selection</td>
<td>175</td>
</tr>
<tr>
<td>6.10</td>
<td>Reward-based multi-objective optimization with bounded population</td>
<td>177</td>
</tr>
<tr>
<td>6.11</td>
<td>Constraints handling in MO-CMA-ES</td>
<td>178</td>
</tr>
<tr>
<td>6.12</td>
<td>Baseline and proposed variants of steady-state MO-CMA-ES</td>
<td>180</td>
</tr>
<tr>
<td>6.13</td>
<td>Typical behavior of $(\mu + 1_{succ})$-MO-CMA on sZDT2 and IHR3 problems</td>
<td>181</td>
</tr>
<tr>
<td>6.14</td>
<td>MO-CMA-ES with different parent selection schemes on LZ09-1.5 problems</td>
<td>183</td>
</tr>
<tr>
<td>6.15</td>
<td>CMA-ES with restarts for different population size and initial mutation step</td>
<td>186</td>
</tr>
<tr>
<td>6.16</td>
<td>Hyper-parameters space exploration by different CMA-ES algorithms</td>
<td>187</td>
</tr>
<tr>
<td>6.17</td>
<td>One run of NIPOP-aCMA-ES on 20-dimensional Rastrigin function</td>
<td>189</td>
</tr>
<tr>
<td>6.18</td>
<td>ECDF plots for several CMA-ES algorithm with restarts</td>
<td>190</td>
</tr>
<tr>
<td>6.19</td>
<td>Comparison of all CMA-ES restart strategies on the Tandem problem</td>
<td>191</td>
</tr>
<tr>
<td>6.20</td>
<td>Kernel PCA-based CMA-ES on 2-dimensional Rosenbrock function</td>
<td>193</td>
</tr>
<tr>
<td>6.21</td>
<td>ECDF plots for high-precision BFGS</td>
<td>194</td>
</tr>
</tbody>
</table>

1. Ranking SVM with different kernel functions | 234
2. SVR for different scaling factors | 235
3. Ranking SVM and rank-based SVR on Sphere function | 236
4. Ranking SVM and rank-based SVR on Rosenbrock function | 237
## List of Tables

4.1 Test functions for ACM-ES ........................................ 95  
4.2 Results of ACM-ES, (n)lm-ES and CMA-ES .................. 96  
4.3 Surrogate hyper-parameters, default value and range of variation .... 107  
4.4 BIPOP-*aACM-ES and IPOP-*aACM-ES on noiseless BBOB problems . 119  
4.5 BIPOP-*aACM-ES and IPOP-*aACM-ES on noisy BBOB problems .... 123  
5.1 Results of NSGA-II, MO-CMA-ES and their ASM-assisted versions .... 148  
6.1 Results of MO-CMA-ES with different parent selection strategies .... 182  
6.2 NBIPOP-aCMA-ES and NIPOP-aCMA-ES on multi-modal BBOB problems192
# List of Algorithms

2.1 The (1+1)-ES with the 1/5th Rule ........................................... 25  
2.2 The ($\mu/\mu_w, \lambda$)-ES with Cumulative Step-Size Adaptation .......... 26  
2.3 The ($\mu/\mu_w, \lambda$)-CMA-ES .............................................. 28  
2.4 ($\mu+\lambda$)-NSGA-II ...................................................... 43  
2.5 Tournament Selection ..................................................... 44  
2.6 ($\mu+\lambda$)-MO-CMA-ES Generic MO-CMA-ES scheme .................. 45  

4.1 Rank SVM Dual problem optimization ....................................... 82  
4.2 ACM-ES ........................................................................... 106  
4.3 Objective function $\text{Err}(\alpha)$ of surrogate model .................. 106  

5.1 Selection of Constraints for RASM model .................................. 137  
5.2 SVM Optimization .................................................................. 138  
5.3 Standard MOEA ...................................................................... 138  
5.4 PARETO-SVM ...................................................................... 139  
5.5 UpdateModel(Archive, Pop) ................................................... 140  
5.6 InfOp(Parents, $F$) ............................................................... 140  

6.1 CMA-ES = Adaptive Encoding + ES ........................................ 154  
6.2 Adaptive Encoding .................................................................. 156  
6.3 ACiD ...................................................................................... 161  
6.4 ($\mu+\lambda$)-MO-CMA-ES Reward-based Parent Selection ............. 172  
6.5 Parent Selection .................................................................... 173  
6.6 Compute Rewards .................................................................... 174
List of Acronyms

**BBOB**  Black-Box Optimization Benchmarking
**CMA-ES**  Covariance Matrix Adaptation Evolution Strategy
**DOE**  Design of Experiments
**EA**  Evolutionary Algorithm
**EC**  Evolutionary Computation
**ECDF**  Empirical Cumulative Distribution Function
**ERT**  Expected Running Time
**ES**  Evolution Strategy
**GA**  Genetic Algorithm
**GP**  Gaussian Process regression
**MO-CMA-ES**  Multi-objective CMA-ES
**MOEA**  Multi-objective Evolutionary Algorithm
**NSGA-II**  Non-dominated Sorting Genetic Algorithm II
**PR**  Polynomial Regression
**RBF**  Radial Basis Function
**SVM**  Support Vector Machine
Part I

General Introduction
Chapter 1

Introduction

1.1 Context/Motivation

The complexity of real-world problems is steadily growing, alongside with the computational power of supercomputers, clusters and graphics processing units. This growth maintains a permanent demand on powerful simulation and modelling frameworks, coupled with robust optimization algorithms; overall, the resolution of advanced real-world problems remains ever more expensive in terms of time and/or money. It is not uncommon to deal with real-world problems where one evaluation of a candidate solution requires a full laboratory experiment (e.g., protein’s folding stability optimization [Chaput and Szostak, 2004], experimental quantum control [Roslund et al., 2009], multi-objective optimization of Diesel combustion [Yagoubi, 2012]). A most common example of real-world problem consists of optimally tuning the hyper-parameters of a complex system (e.g., compiler optimization [Fursin et al., 2005], job allocations in a computational grid [Tesauro et al., 2007]), as opposed to using expert hand-crafted parameter settings.

Evolutionary Algorithms (EAs) have been thoroughly investigated in this context due to their ability to solve complex optimization problems by coupling problem-specific variation operators and selection operators:
Firstly, a search directed by a population of candidate solutions is quite robust with respect to a moderate noise and multi-modality of the objective function, in contrast to some classical optimization methods such as quasi-Newton methods.
Secondly, the role of variation operators is to explore the search space of potential solutions, taking into account already retrieved information about the problem. The – usually randomized – variation operators, together with the representation of the candidate solutions, encapsulate extensive prior knowledge about the problem domain.
Finally, selection is responsible for directing the search toward more promising regions on the basis of the current solutions, controlling the exploration versus exploitation trade-off.

These features make EAs very flexible and suitable for virtually any optimization problem provided that parametrized solutions can be comparatively assessed. Specifically, EAs are meant to address a wide range of optimization problems: involving a unimodal or multi-modal objective function; with or without noise; in a low- or high-dimensional search space; constrained or without constraints; involving a stationary or a dynamic objective; involving a computationally expensive or cheap objective function; and last but not least, involving a single or multiple objective functions.

The Evolutionary Computation (EC) community has introduced a variety of approaches to address all abovementioned types of optimization problems. On the one
hand, these approaches have been empirically validated by practitioners, reporting a number of breakthrough applications (see, e.g., [Coello and Lamont, 2004, Yu et al., 2008, Chiong et al., 2011] among many others). On the other hand, the theoretical properties of some simple variants of the most successful approaches have been established and related to the first principles of optimization, specifically the natural gradient [Wierstra et al., 2008, Arnold et al., 2011, Akimoto and Ollivier, 2013].

In the meanwhile, rare position papers discuss the growing gap between the theory and the practice of EAs [Michalewicz, 2012]. The latter paper, reflecting the author’s personal point of view based on 20 years of experience in the field of EC, invites to discuss the question of self-assessment in EC. It suggests that most generally and like many other scientific fields, EC is driven by two main forces: research and applications. In practice, some approaches are known to be the most efficient ones with respect to the current benchmark or real-world problems, whereas some other and perhaps less efficient approaches are considered to have more solid theoretical foundations, or to be more general, more flexible or more elegant. In this perspective, EC could itself be viewed as a black-box, non-stationary, multi-objective problem per se, with its deployment being shaped under simultaneous researcher and practitioner dynamic pressures. Along this line, the EC community collectively aims at building an optimal Pareto set of optimization approaches.

The lessons learned from multi-objective optimization are that, in order to build a dense Pareto front, one must mandatorily preserve the population diversity, and avoid discarding too easily the solutions which are dominated w.r.t. the current objectives. Despite their shortcomings, some solutions are found to pave the way toward truly non-dominated solutions in unfrequented regions of the Pareto front. In other words, the celebrated Exploitation versus Exploration trade-off should be considered at the level of algorithm design, too. Along this line, the contributions presented in this thesis and detailed below can be divided into two categories:

- In a first category (the exploitation category, Chapters 4 and 5), we present principled and efficient extensions of the prominent Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [Hansen et al., 2003], aimed at surrogate-based single-objective optimization [Jin, 2005] or surrogate-based multi-objective optimization [Knowles and Nakayama, 2008] and based on the tight coupling of stochastic optimization and statistical machine learning [Vapnik, 1995, Joachims, 2005].

- In a second category (the exploration category, Chapter 6), we present new stochastic optimization schemes, based on Adaptive Coordinate Descent (Section 6.1), with kernel-based change of representation (Section 6.4.1), or tackling the CMA-ES hyper-parameter tuning problem as a (very noisy) optimization problem (Section 6.3).

## 1.2 Main Contributions

As already said, the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) and its variants are acknowledged to be the most efficient approaches in continuous black-box optimization [Hansen et al., 2010b]. The first part of the presented contributions (the
1.2 Main Contributions

"exploitation" part) proceeds by coupling statistical machine learning algorithms with CMA-ES, in order to address specific issues such as surrogate-based optimization and multi-objective optimization, while preserving all the invariance and robustness properties of CMA-ES.

The second part (the "exploration" part) investigates independent issues: hyper-parameter tuning; non-linear adaptive representations; reward-based multi-objective optimization.

1.2.1 Coupling CMA-ES with Statistical Machine Learning

One of the main limitations of EAs, the large number of function evaluations required for a reasonable accuracy of optimization, prevents EAs from being used on computationally expensive problems, where one evaluation takes more than one second and up to a few hours or a day\(^1\).

A common way to reduce the overall number of function evaluations required to solve expensive optimization problems is to build surrogate (approximation) models and carefully treat the information available from the evaluated candidate solutions (see [Box and Wilson, 1951, Jin, 2005], Chapter 3).

ACM-ES In the spirit of the comparison-based CMA-ES approach, we used a comparison-based surrogate-learning approach, referred to as Ranking SVM [Herbrich et al., 1999, Joachims, 2005]. The surrogate learning phase thus does not exploit the objective values of the available solutions, but only their ranks. In this way, the overall surrogate-based optimization approach is invariant under monotonous transformations of the objective function, a most desirable property.

A key contribution compared to the state of the art [Runarsson, 2006, Bouzarkouna et al., 2010] is to reuse the covariance matrix built by CMA-ES itself within Ranking SVM; along this coupling, the overall approach called ACM-ES inherits CMA-ES invariance under orthogonal transformations of the search space. A second contribution is to automatically and adaptively control the refreshment rate of the surrogate model (surrogate lifelength), depending on the empirical error of the previous surrogate model, under the assumption of a smooth variation of the optimal (unknown) surrogate along the search.

A third contribution is to interleave surrogate learning with the automatic online adjustment of the learning hyper-parameters, being reminded that the surrogate model quality critically depends on the learning hyper-parameters.

\(^1\) This limitation is especially observable in the special, but quite common case of the unimodal noiseless continuous optimization, where gradient information is useful and quasi-Newton methods such as BFGS [Shanno, 1970], proposed 40 years ago, usually outperform most of advanced EAs [Hansen et al., 2010b]. Unfortunately, this has become, directed by natural selection, a common practice in EC community to not view BFGS as an alternative approach, arguing that, anyway, the latter does not perform well on multi-modal and noisy functions. This may create a false impression about the performance of EAs in EC community, but does not create this impression among practitioners who do compare EAs with non-EC algorithms, such as BFGS and recently proposed NEWOUA [Powell, 2006]. We shall return to this point in Chapter 4.
The proposed algorithm, extensively benchmarked on the Black-Box Optimization Benchmarking (BBOB) [Hansen et al., 2010a] framework against classical (BFGS [Shanno, 1970], NEWUOA [Powell, 2006], GLOBAL [Pal et al., 2012]) and Evolutionary Algorithms, is shown to improve by a factor 2 to 4 on the state of the art.

ASM In the same spirit, Aggregated Surrogate Models (ASMs) inspired from the Support Vector Machine framework are used for multi-objective optimization. A first approach aims at characterizing the current Pareto-front in the spirit of [Vapnik, 1995], as follows. On the one hand, One-Class SVM is used to characterize the (low-dimensional) region of the Pareto front like [Schölkopf et al., 2001]; on the other hand, Regression SVM [Vapnik, 1995] is used to characterize the distance to the Pareto front. Both models together are used to estimate the progress toward the (true) Pareto front, by filtering the offspring estimated on the surrogate. A second approach uses Ranking SVM to directly learn the Pareto dominance relation. A main originality of the approach compared to the state of the art in surrogate-based multi-objective optimization, is to characterize the Pareto front using a single surrogate (as opposed to, using a surrogate for each objective function), based on preference relations among solutions, e.g., based on Pareto dominance, Quality Indicators and Decision Maker preferences. Likewise, an extensive benchmark of the two proposed approaches establishes that they improve on the state of the art by a factor 2.

1.2.2 Exploratory contributions

Three exploratory contributions inspired from CMA-ES are proposed:

- The first one called Adaptive Coordinate Descent (ACiD) hybridizes the simple Coordinate Descent approach with the CMA-ES adaptation of the problem representation. On the one hand, ACiD demonstrates a linear empirical complexity with respect to the problem dimensionality (as opposed to, at least quadratic in CMA-ES). On the other hand, ACiD can be extended to non-linear change of the problem representation by using the famed kernel trick [Vapnik, 1995]. Comprehensive validation on BBOB shows that ACiD is competitive with CMA-ES.

- The second contribution is concerned with multi-objective optimization. In some cases, the failure of existing approaches is investigated and explained from the weakness of the parent selection procedures. A multi-objective reward function, measuring how much a parent contributes to the future populations, is designed and experimented. This approach measuring the "value" of an individual as opposed to its instant value, in the perspective of reinforcement learning [Sutton and Barto, 1998].

- Finally, the restart strategies of CMA-ES on multi-modal fitness landscape are analyzed. The optimal restart strategy is viewed as a sequential decision problem, setting in each restart the two CMA-ES hyper-parameters (the population size and
The restart strategy is viewed as (yet another, but very noisy) optimization problem, and new restart strategies, with adaptive selection of the most suitable regime of restarts, are defined and validated.

1.3 Thesis Outline

Chapter 2 presents a historical overview of Evolutionary Computation and introduces Evolutionary Algorithms in the context of continuous black-box optimization, focusing on state-of-the-art algorithms for single-objective (respectively, multi-objective) optimization such as CMA-ES (respectively, MO-CMA-ES).

Chapter 3 presents a survey of popular techniques for surrogate model learning, exploitation and control, and reviews single- and multi-objective surrogate-assisted optimization algorithms.

Chapter 4 introduces surrogate-assisted versions of CMA-ES based on pre-selection of promising individuals and direct optimization of surrogate model, focusing on questions of surrogate hyper-parameters adaptation and surrogate exploitation control.

Chapter 5 extends surrogate-assisted CMA-ES with pre-selection to multi-objective optimization and discusses several ways to learn and incorporate preference information into surrogate models.

Chapter 6 provides a collection of CMA-like algorithms to explore prospective directions of the research in EC, including linear time Adaptive Coordinate Descent, new parent selection strategies for MO-CMA-ES and alternative restart strategies for CMA-ES.

Chapter 7 concludes the thesis and summarizes our contributions toward understanding of efficient continuous black-box optimization.
Part II

Background Review
Chapter 2

Evolutionary Algorithms for Continuous Black-Box Optimization

It may be said that natural selection is daily and hourly scrutinising, throughout the world, every variation, even the slightest; rejecting that which is bad, preserving and adding up all that is good; silently and insensibly working, whenever and wherever opportunity offers, at the improvement of each organic being in relation to its organic and inorganic conditions of life.

Charles Darwin, 1859, On the Origin of Species, Chapter 4.

The origins of the idea of natural selection can be traced as far as back to ancient Greeks philosophers Empedocles, Aristotle et al. A more contemporary and widely accepted version of this idea by Charles Darwin assumes the evolution of organic beings. In the present thesis, as well as in Evolutionary Computation, we assume that natural selection is a driving force of evolution not only for organic beings, but also for non-organic ones. In Section 2.1, we give a brief historical overview of the evolution of this concept in Evolutionary Computation domain. Section 2.2 introduces continuous black-box optimization and discusses main properties of black-box problems, which motivate the development of specific approaches to efficient optimization. We present the state-of-the-art Evolutionary Algorithms for single-objective (Section 2.3) and multi-objective (Section 2.4) continuous optimization. Finally, in Section 2.5, we conclude the Chapter and discuss some open questions of the field.

2.1 Historical Overview of Evolutionary Computation

Evolutionary Computation (EC) is the research field of nature- and evolution-inspired computational methods used to solve real-world problems, and their toy scientific models. We divide the history of the EC field into two parts: early approaches proposed before the first International Conference on Genetic Algorithms (ICGA) in 1985, and modern approaches proposed after this conference. According to an analysis by [Alander, 1994] of 2500 papers published on Genetic Algorithms, Evolution Strategies, Evolutionary Programming, etc., only 215 papers were published between 1957 and 1984, compared to 928 published between 1985 and 1990. Now the number of published papers per year is still growing, and while the exact number of papers is difficult to estimate, it might lie in order of tens of thousands, given that for one of the most cited researchers of the field of
Chapter 2. Evolutionary Algorithms for Continuous Black-Box Optimization

evolutionary multi-objective optimization, Prof. Kalyanmoy Deb, the number of citation in the year of 2011 is about 6000 and about 45000 citations overall.

2.1.1 Early Approaches


Fisher claimed that natural selection is not Evolution, as it was identified in biological sciences, but an independent principle worthy of scientific study. He notices that the first decisive experiments, which opened in Biology this field of exact study, were due to a young mathematician, Gregor Mendel, whose statistical interests extended to physical and biological sciences (note that, experimental clarity of Mendel’s work was later criticized by Fisher [Fisher, 1936]). According to Fisher, while the types of mind which result from training in mathematics and biology differ profoundly, this difference is not in the intellectual, but rather in imaginative faculty. The biologists are early introduced to the world with its immense variety of living things and amazing complexity, and they study this world as it is. The mathematicians instead deal with barest abstractions and often try to work in a wider domain than actual for a far better understanding of the actual. The difference of these two approaches lies not in intellectual methods and even less in the intellectual ability, but in an enormous and specialized extension of the imaginary faculty which each field has experienced.

It is interesting to note that Fisher studied the probability of improvement of mutation on a simple example of Sphere function $\sum_{i=1}^{n} x_i^2$. The initial solution is located on a distance $d/2$ from the center of the Sphere; the mutation operator performs a shift with radius $r$; the mutation is successful if new solution lies inside the Sphere, and unsuccessful otherwise. The limit of probability of improvement in this case is $0.5(1 - r/d)$, meaning that large mutations are usually unsuccessful and destructive, while small mutations are more successful, but lead to a slow progress of evolution. The latter observation was later confirmed and motivated the development of randomized optimization approaches with the adaptation of mutation step-size. Nowadays, both Evolutionary Computation and Population Genetic [Orr, 2000] show that the rate of adaptation is maximized for mutations of intermediate size.

Apart from Fisher’s enormous contribution to statistics, he made the terms of natural selection and mutation mathematically more comprehensible for a wider scientific public. Another of his books, "The Design of Experiments" [Fisher, 1935] became the foundation for the field of Design of Experiments (DOE). This research on DOE for determining optimal conditions of chemical investigations was continued by a statistician George P. E. Box [Box and Wilson, 1951], the husband of one of Fisher’s daughter Joan.

Probably inspired by Fisher’s idea of small mutations and motivated by development of DOE methods which do not interrupt manufacturing processes during its optimization, George P. E. Box proposed Evolutionary Operation (EVOP) [Box, 1957]. One of the first real-world applications of Evolutionary Operation was described in [Box and Hunter, 1959] for 2-dimensional and 3-dimensional cases. In EVOP, a reason-
ably small hyper-rectangle centered in the current solution (current mean) is constructed such that the quality of the manufacturing process changes only slightly by using one of possible solutions from the hyper-rectangle. Then, the quality of solutions which lie on the corners of the hyper-rectangle is estimated and the solution with the best quality is chosen as the next mean of the hyper-rectangle. EVOP performs coordinate search with adaptive step-size and resembles the famous pattern/direct search algorithm of Hooke and Jeeves proposed later [Hooke and Jeeves, 1961]. At least 50 articles about EVOP were published during the first 10 years after its introduction. Box popularized the idea that artificial evolution can be useful for real-world optimization problems as well as Fisher showed that the natural selection can be mathematically studied.

The first computer-assisted experiments of Evolutionary Algorithms are often referred to Nils Aall Barricelli, who was visiting Princeton University in 1953, 1954. Barricelli emulated evolution of "organisms" to find the "decision function", that determines the next move for a player of several games such as Chess, Tax Tix et al. The work was published first in [Barricelli, 1954], then translated to [Barricelli, 1957]. David B. Fogel truly said [Fogel, 2006]: "Nils Barricelli remains a virtually unknown figure in the Evolutionary Computation community, but that is no reflection on his pioneering contributions and creative thoughts." Unfortunately, many other great researchers were also relatively unknown, such as Alex Fraser who simply proposed and implemented [Fraser, 1957] a pre-Genetic Algorithm with the population of individuals coded using binary strings, so-called ($\mu, \lambda$) selection scheme and specific crossover and mutation operators (!).

[Bremmerman, 1958, Bremermann, 1962] discussed the idea of using evolution processes to solve optimization problems, making an analogy with DNA and using sequences of binary variables to code the solution. The author suggested to apply bit-flip mutation operation to each gene with a probability $p$, claiming that "In many cases optimal speed of improvement is reached for $p = 1/n$, where $n =$ number of genes". This mutation probability has been proven by [Droste et al., 1998] to be optimal on linear functions in order to reach expected running time $O(n \log(n))$ for (1+1)-Evolutionary Algorithm, where the generated by mutation offspring replaces the parent solution if it has better fitness function. The term of "fitness function" is also associated with the objective function $f : \mathbb{R}^n \mapsto \mathbb{R}$ in [Bremermann, 1962].

[Rastrigin, 1963b] (see also [Rastrigin, 1960, Rastrigin, 1963a]) discussed the advantages of randomized search methods on a real-world problem of lathe machine-tool parameter tuning. This machine-tool operates using many relatively simple single-point cutting tools, controlled by arms to produce objects of different geometry. The goal is to minimize a function $f(x)$ of the error of geometry of products by tuning $x$, the positions of the arms. The author analyses possible scenarios of using Gauss-Seidel and Gradient-based methods and proposes to use a randomized search method. In this (1+1)-Evolutionary Algorithm, before each search step, all arms hold positions of the best solution (with the minimum error) found so far. Then, all arms should be simultaneously tuned by some random values to produce a new solution, which is accepted to be the new best solution only if it leads to a smaller $f(x)$. The mutation operation is not specified, and actually is not called "mutation", while this was probably meant, because the first part of the book discusses the importance of mutation for organic beings. The author also suggests to
adapt the mutation distribution ([Rastrigin, 1963b], p. 76): "One can use a reward-based system, increasing the probability of those changes, which lead to improvements of quality of the system, i.e., to minimization of the criterion of similarity". The latter proposal was realized 30 years later in CMA-ES.

An idea of simultaneous mutation of problem parameters very similar to the Rastrigin's one was implemented by Hans-Paul Schwefel and Ingo Rechenberg in the 1960s. From Hans-Paul Schwefel's great book "Evolution and Optimum Seeking" [Schwefel, 1993] about "classical" and evolutionary optimization approaches of the 1960s and 1970s: "In 1963 two students at the Technical University of Berlin met and were soon to collaborate on experiments which used the wind tunnel of the Institute of Flow Engineering. During the search of the optimal shapes of bodies in a flow, which was then a matter of laborious intuitive experimentation, the idea was conceived of proceeding strategically. However, attempts with the coordinate and simple gradient strategies were unsuccessful. The one of the students, Ingo Rechenberg, now Professor of Bionics and Evolutionary Engineering, hit upon the idea of trying random changes in the parameters defining the shape, following the example of natural mutations. The evolution strategy was born." First simulations of different versions of two membered (one parent generates one offspring) (1+1)-Evolution Strategies (ESs) were performed on the first digital computer Zuse Z23 of Technical University of Berlin [Schwefel, 1965]. Theoretical analysis of these strategies and first proposals of $\mu$ multi-membered strategies ($\mu + 1$) were published later in Ingo Rechenberg’s doctorate thesis [Rechenberg, 1973]. Unfortunately, between 1976 and 1985 the research on ESs suffered from insufficient financial support [Schwefel, 1993].

Another key sub-field of Evolutionary Computation, Evolutionary Programming was invented by Lawrence J. Fogel also in the 1960s, leading to the book "Artificial Intelligence through Simulated Evolution" [Fogel et al., 1966]. Fogel proposed to evolve the population of Finite State Machines (FSMs) to solve problems of prediction and control in an environment, defined as a set of sequences from a finite alphabet. All FSMs of the population are initially identical. Each FSM uses known part of the environment (previously observed symbols) to predict the next symbol from the unknown environment. The rate of this prediction is used as the fitness function. At each iteration of the algorithm, a mutation operator is applied to each FSM to generate an offspring solution. The best half of a mixed population of parents and offspring becomes the new population. This loop is repeated for several iterations, then the best machine is taken to predict the new symbol of the environment, the output symbol is attached to the environmental string. Fogel proposed several mutation operators to add and delete states and change links of FSMs. The process is named Evolutionary Programming probably in contrast to Linear Programming, Dynamic Programming and Quadratic Programming.

[Schumer and Steiglitz, 1968] proposed Adaptive Step-Size Random Search (ASSRS), an adaptive version of Fixed Step-Size Random Search (FSSRS) [Rastrigin, 1960, Rastrigin, 1963a, Mutseniyekhs and Rastrigin, 1964]. They provided proofs for Optimal Step-Size Random Search (OSSRS) on unimodal functions and showed that: i). the optimal step-size is proportional to the distance to the optimum, similar results were obtained later for (1+1)-ES [Auger and Hansen, 2006]; ii). there is an optimal probability of improvements for ASSRS (see the 1/5th rule proposed later in [Rechenberg, 1973]); iii). the
average number of function evaluations to reach a desired accuracy (relative to starting point) is linear in the problem dimension. Since the distance to the optimum is usually unknown, only an approximation of the optimal step-size can be estimated. In ASSRS strategy two alternative solutions are generated by mutation of the current best solution by a random sampling in the hyper-sphere with two different radii/step-sizes. The step-size which generated the best solution (made the best improvement) is used for the next iteration. Experimental results demonstrated that Newton-Raphson method with Hessian approximation by finite differences outperforms ASSRS by a factor of about 2 on Sphere function \( f(x) = \sum_{i=1}^{n} x_i^2 \) for \( n > 20 \), while on \( \sum_{i=1}^{n} x_i^4 \) function ASSRS outperforms Newton-Raphson and simplex methods already for \( n > 10 \). It should be noted that ASSRS finds target fitness function \( f_{\text{target}} = 10^{-8} \) after about 80\( n \) function evaluations, i.e., it shows the same performance as (1+1)-ES with the 1/5-th rule [Auger, 2009].

In 1975 John H. Holland wrote his book "Adaptation in Natural and Artificial Systems" [Holland, 1975], where he proposed Genetics Algorithms (see also Genetic/Adaptive Plans [Holland, 1969]). Indeed, Genetic Algorithms (GAs) were born from the "soup" of ideas of earlier researchers (some of these researchers were mentioned above). The key contribution of Holland was not only to identify what is relevant in this "soup", but to respect the following scientific trade-off: i). provide a simplified and at the same time rich theory of how the adaptation and evolution work; ii). provide a working algorithmic framework to simulate the evolution for solving real-world problems and their scientific toy models. In Holland’s GA, the population were represented by chromosomes, candidate solutions of the problem, using binary coded "genes". The selection operator allows chromosomes with better fitness function (objective function) to produce offspring more often. The recombination procedure consists of crossover, mutation and inversion. Crossover exchanges sub-parts of parents chromosomes, mutation randomly changes randomly selected "genes", inversion reverses the ordering of "genes" in the chromosome. Holland also proposed the so-called Schema Theorem to provide theoretical evidence of GA convergence, however it was later criticized [Burjorjee, 2008, Antonisse, 1989] for making unacceptably strong assumptions on the distribution of fitness over the genome set.

After World War II the basic research attracted significant funding resources. In 1950s and 1960s major research agencies over the world, such as DARPA, had been promised to solve key military problems using Artificial Intelligence (AI): speech recognition and understanding, automatic machine translation, creation of general purpose robots, etc. However, most of these ambitious problems were not solved (and still are open problems in our 21-th century) or used approaches were criticized (e.g., in "Perceptrons" book by Marvin Minsky [Minsky and Seymour, 1969]), that led to funding cuts. So-called AI Winter took place from the early 1970s to the early 1980s, when major agencies switched to "goal-oriented" research and AI research continued in a limited way, because major funding for projects was difficult to find. A frustration about AI also influenced the research in a future of the "newborn" field of Evolutionary Computation: "The general interest in this type of optimum seeking algorithms was not broad enough for there to be financial support" [Schwefel, 1993]. Another important aspect was the experimental nature of Evolutionary Algorithms, that required significant CPU resources for algorithm design phase in contrast to exact methods. The latter problem was partially solved with
the emergence of personal computers in the 1980s and 1990s, that led to a "boom" of Evolutionary Computation.

2.1.2 Modern Approaches


The number of papers published on so-called modern approaches is at least by a factor of 1000 larger than for so-called early approaches. Therefore, we do not pretend to cover this topic in a short Section. Instead, we list the most widely used approaches and give their brief comparison analysis.

Genetic Algorithms

Genetic Algorithms (GAs) is a core of Evolutionary Computation, and probably the most common approach used for initial pedagogical studies of evolution-inspired optimization and search. A good starting point of such studies is Simple Genetic Algorithm (SGA), a Holland’s GA, where binary-coded parent solutions compete with offspring solutions, generated after multi-point crossover and bit-flip mutation of parents. Real-Coded Genetic Algorithms (RCGA) [Wright, 1991] popularized the real-value representation of decision variables and enriched the set of variation operators, an example of state-of-the-art RCGA algorithm is the generalized generation gap (G3) model with efficient parent-centric recombination operator (G3-PCX) [Deb, 2005]. The key aspect of design of GA is the ”right” choice of representation of candidate solutions for a given problem, such that the variation operators favor successful exploration and exploitation of the search space. An illustrative example is the Travelling Salesman Problem (TSP), where a candidate solution can be represented as a sequence/tour of cities which should be visited. To make new prospective sequences from the existed ones, problem-specific variation operators can be used instead of classical crossover and mutation operators, which may create infeasible solutions (e.g., with more than one visit of the same city [Radcliffe and Surry, 1994]).

It would be unfair to think of Genetic Algorithms only in terms of their comparative performance on specific optimization problems. The GAs is, first of all, a quite general and flexible factory/template of new approaches. The flexibility of GAs allowed to explore a wide range of ideas, such as the study of diversity preservation in multi-modal optimization [Mahfoud, 1995, Sareni and Krahenuh, 1998]. Despite a huge progress made in 1990s, the research on GAs substantially stagnated in 2000s, at least in the domain of continuous single-objective optimization, where problem-specific approaches such as Evolution Strategies became the method of choice. In multi-objective optimization, GAs
made a long step forward (e.g., Non-dominated Sorting Genetic Algorithms-II (NSGA-II) [Deb et al., 2002]), which other Evolutionary Computation approaches still try to follow.

**Evolution Strategies**

Evolution Strategies from the beginning have addressed continuous optimization (though discretized in the early approaches). Several attempts to extend ESs to mixed-integer optimization [Bäck and Schütz, 1995, Li et al., 2006], unfortunately, have not attracted enough attention in the field. The latest results show that mixed-integer optimization is challenging and the premature convergence is possible even for relatively simple problems [Hansen, 2011, Li et al., 2011].

In this thesis, we focus on ESs as a prospective approach specifically developed for continuous optimization, we give a detailed description of ESs in Section 2.3.1.

**Estimation of Distribution Algorithms**

In contrast to many EAs, where the distribution of candidate solutions is defined implicitly by variation operators, in Estimation of Distribution Algorithms (EDAs) [Larrañaga and Lozano, 2002] the sampling of new solutions is explicitly defined by a chosen probabilistic model, e.g., multi-variate normal distribution. The baseline EDA algorithms are: Univariate Marginal Distribution Algorithm (UMDA) [Mühlenbein and Mahning, 2002], Estimation of Multivariate Normal Algorithm (EMNA) [Larrañaga and Lozano, 2002], Cross-Entropy Method (CEM) [Rubinstein and Kroese, 2004].

**Ant Colony Optimization**

Ant Colony Optimization (ACO) was proposed by Marco Dorigo in his PhD thesis [Dorigo, 1992]. ACO is an optimization technique (often viewed as an EDA) which imitates the behavior of ants to solve (usually) combinatorial optimization problems. As well as natural ants, simulated ants are seeking for a path between their colony and a source of the food. They lay down pheromone trails such that other ants will more likely to follow these trails and may reinforce attractiveness of the shortest path to the food. Trail evaporation reduces pheromone values of all trails over time, that allows to preserve some diversity in optimal path seeking.

Early ACO algorithms were aimed to solve TSP problem, but further were extended to a wide range of combinatorial problems [Dorigo and Stützle, 2003]: scheduling, vehicle routing, quadratic assignment, set covering problems, etc.

The main criticism of ACO algorithms can be demonstrated on the TSP problem (a similar criticism is also relevant for EAs in general). Problem-specific approaches, such as modified Lin-Kernigan heuristic, implemented in LKH-2.0 library are able to solve this toy problem optimally for tens of thousands of cities (85,900 cities in [Applegate et al., 2009]) and efficiently for millions of cities in linear time [Helsgaun, 2009]. The best known solution for the World TSP Tour of 1,904,711 cities was found by LKH-2.0 in October 25, 2011. It has length at most 0.0477% greater than the length of an optimal tour (estimated as the best lower bound established using Concorde TSP code). It seems that these results are better than the results of ACO algorithms, for which the instances of few thousands cities are already difficult to solve with and without 3-opt local search [Oliveira et al., 2011].

A common and quite fair answer to this criticism is: i). ACO algorithms are aiming to find a general approach, which will be useful for a wide range of combinatorial problems
(this is also true for "classical" approaches); ii). ACO algorithms investigate optimization problems in a nature-inspired way, trying to fulfill the gap of understanding between optimization theory and evolutionary biology. These two things are both relevant for other EC approaches. Some progress in generalization was recently made when ACO algorithms for continuous optimization (ACOR) were proposed [Socha and Dorigo, 2008] and uncovered some similarity with the CMA-ES algorithm.

**Genetic Programming**

Genetic Programming (GP) was proposed by Nichael L. Cramer during the first ICGA conference in 1985 [Cramer, 1985] and later popularized by John R. Koza [Koza, 1990, Koza, 1992]. Genetic Programming is an evolution-inspired technique which imitates artificial evolution of computer programs that perform a used-predefined task. GP can be viewed as a GA with a specific representation of individuals, usually in tree structure, and variation operators defined for this tree. In contrast to other optimization techniques, the number of components (variables) representing a candidate solution is usually dynamically changing during the search. This often favors extensive search in the space of possible topologies, instead of intensive parametric search for a given topology. As a consequence, the final programs are often too complex for relatively simple problems. To control this bloat, the minimization of the program size/complexity can be set as the second objective after the program’s fitness, leading to a multi-objective version of GP [Koza, 1992]. It is reported by John R. Koza, that analog electronic circuits, designed using GP often reinvent or/and outperform patented solutions, found by humans. However, the search in the space of possible topologies is usually heavily constrained by a designer and user of Electronic Design Automation tools. In Microwave frequency range, this becomes especially important, because lumped elements are not sufficiently precise anymore and should be replaced by approximated model for a given technological process, where each element typically has several variables. The fitness computation also becomes more expensive. Thus, parametric optimization becomes much more important than it is usually assumed in GP, and a reasonable trade-off between topological and parametric search should be found [Loshchilov, 2009].

**Differential Evolution**

Differential Evolution (DE) was proposed by Rainer Storn and Kenneth Price [Storn and Price, 1995] and became especially popular in recent few years thanks to relative simplicity and efficiency of this method. Basically, the original DE adds the weighted difference between two population vectors to a third vector. Each individual $x_i$ of current population generates one offspring, which will replace the parent in the next generation if it has better fitness. The offspring $y_i$ is a copy of the parent $x_i$, except that some variables $x_{i,j}$ should be modified: i). randomly picked variable index $j_{rand}$ among $n$ variables; ii). all variables selected with the crossover probability $CR$. If a modification of offspring’s variable $y_{i,j}$ is needed, the new value is computed in the following way:

$$y_{i,j} = x_{a,j} + F(x_{b,j} - x_{c,j}),$$

where $F \in [0, 1]$ is so-called differential weight and $a$, $b$, $c$ are indexes of individuals from the current population. These three or more individuals are usually different from $y_i$ and are chosen depending on the specific mutation strategy of DE.
DE typically has only few parameters (population size, CR, F) and several mutation strategies. This allowed DE community to develop a set of techniques for offline [Storn and Price, 1997] and online parameter tuning [Brest et al., 2006], as well as efficient adaptive operator selection methods [Fialho et al., 2010]. The latter methods add to DE the robustness it was missing, and would be useful also in other fields of EC.

Particle Swarm Optimization

The concept of Particle Swarm Optimization (PSO) was discovered by James Kennedy and Russell Eberhart [Kennedy and Eberhart, 1995] through simulation of a simplified social model of bird flocking, fish schooling, etc. In PSO the population is represented as a swarm of particles (individuals), which are seeking for the optimum of a function \( f(x) \). Each particle \( i \) has its current position \( x_i \), best position \( p_i \) it visited so far and the velocity \( v_i \), which is simply the difference (shift) between its new and old position in the decision space. The shift for each particle \( i \) and decision variable \( j \) is computed in the following way:

\[
v_{i,j} = \omega v_{i,j} + c_1 r_p (p_{i,j} - x_{i,j}) + c_2 r_g (g_j - x_{i,j}),
\]

where \( r_p, r_g \sim U[0,1] \) are drawn uniformly, \( g \) is the swarm’s best known position and \( \omega, c_1, c_2 \) are control parameters of the algorithm.

The use of current best solution in variation operators favors fast convergence on unimodal functions, and may lead to premature convergence on multi-modal functions. A number of techniques to avoid the premature convergence to local optima have been proposed such as update of velocity of a particle using best \( p \) locations of other particles [Liang et al., 2006]. Another approach to preserve the diversity is, instead of using global swarm’s best particle \( g \) for variation, to use the best one of some local ”sub-swarm”, supposing that the swarm has some topological structure. This structure can be fixed (original global star, local star, von Neumann neighborhood [Kennedy and Mendes, 2002]), but there are also approaches to adapt the topological structure during the search [Elshamy et al., 2007].

Other Approaches

Particle Swarm and Ant Colony Optimization algorithms proposed in the early 1990s inspired a wide range of so-called Swarm Intelligence (SI) approaches, which have become especially popular since the early 2000s, one can say that they have been replacing GAs. Swarm Intelligence is usually associated with a population of relatively simple agents, whose collective interaction leads to the emergence of ”intelligent” global behavior. Many SI algorithms have been proposed in recent years [Blum and Merkle, 2008]: artificial bee colony algorithm, bat algorithm, Cuckoo search, firefly algorithm, harmony search and many others. Some of these approaches imitate interesting processes, observed in nature, that probably will lead to their better understanding.

However, a common criticism, which is also relevant to the whole EC community, is that most of these approaches have very weak theoretical background and a lack of mathematical proofs of their efficiency. In some cases the lack of theoretical analysis of the algorithm can be compensated by empirical observations of its good performance. To assess the performance correctly, a precise experimental setup and procedure should be defined. Unfortunately, this is often not the case for many papers published in EC commu-
nity every year, that makes the evolution of optimization approaches even more noisy, than probably it is already because of No Free Lunch Theorem [Wolpert and Macready, 1997]. To reduce the risk of biased benchmarking, in this thesis, we extensively use BBOB framework. Another point of criticism is well illustrated in the paper ”A Rigorous Analysis of the Harmony Search Algorithm: How the Research Community can be Misled by a ”Novel” Methodology” [Weyland, 2010], there the author discusses obvious similarities between Harmony Search Algorithm and Evolution Strategies.

To answer the mentioned above criticism, we should again re-think the evolution of the EC field as multi-objective optimization, as described in Section 1.1. From this point of view, we indeed need some diversity of algorithms, because it might be too ”greedy” to view the performance of the algorithm as the only (single) objective. However, we are more or less sure that we do not want to have ”bad clones” of already existing approaches, without giving any credits, because this adds an artificial noise to the algorithm diversity measure. We also do not want to add any noise to the algorithm performance measure due to messy experimental validation procedures.

2.2 Continuous Black-Box Evolutionary Optimization

In the black-box scenario, we want to minimize (without loss of generality) an objective (cost, fitness) function

\[ f : \mathbb{R}^n \mapsto \mathbb{R} \]  

The function is called black-box, because no specific assumption on function is made, \( f \) is also called continuous black-box function as decision variables \( \mathbf{x} \) are continuous variables.

The objective of black-box optimization is to find one or a set of solutions \( \mathbf{x} \) with as small values \( f(\mathbf{x}) \) as possible, using as small number of function evaluations as possible. We suppose that all information (except dimension \( n \)) about the black-box function \( f \) comes from evaluations of candidate solutions \( \mathbf{x} \) with \( f \). The number of function evaluations used is a common search costs measure, which characterizes the amount of information acquired about the \( f \) during the optimization process.

There are several properties of \( f \) which can make its black-box optimization difficult. Below we list and comment some of these properties (see also [Hansen et al., 2008]).
2.2 Continuous Black-Box Evolutionary Optimization

Multi-Modality

A local minimum (optimum of minimization) of an unconstrained function \( f \) is a vector \( x^l \in \mathbb{R}^n \) such that there is a neighborhood \( N \) of \( x^l \) with no better points than \( x^l \):

\[
\forall x \in N \cap \mathbb{R}^n, f(x) \geq f(x^l) \quad (2.4)
\]

A global optimum is a vector \( x^g \in \mathbb{R}^n \), such that there is no \( x^l \in \mathbb{R}^n \) better than \( x^g \):

\[
\forall x \in \mathbb{R}^n, f(x) \geq f(x^g) \quad (2.5)
\]

An objective function \( f \) is called multi-modal if it has more than one optimum. The optimization of multi-modal function is difficult, because usually there are no guarantees that the optimum that was found after the optimization by the algorithm is the global one. Therefore, to increase the chance to find the global optimum, several local optima should be explored in one optimization run or during several restarts. There is no consensus about the best global strategy for multi-modal optimization, two the most commonly used are: i). to preserve some diversity of solutions during the search, using niching approaches to explore local optima and hopefully find the global one(s) [Singh and Deb, 2006]; ii). to increase the probability to find the global optima performing multiple restarts with the same or different parameter settings of the algorithm (see Section 6.3.3). Multi-modal functions have typically much more complicated fitness landscapes than the unimodal ones, because of (sometimes large) basins of attraction located around the local optima.

High-Dimensionality

The volume of the search space increases exponentially with \( n \), the number of decision variables. This can be illustrated on a simple example: first, place 100 points in a real interval \([0, 1]\); then to have a similar coverage in terms of distance between points in 10-dimensional space requires \( 100^{10} = 10^{20} \) points. Therefore, some optimization techniques, useful for small dimensions such as full coverage of the search space, become useless for large dimensions. This effect is called the curse of dimensionality [Richard Bellman, 1957]. Many important concepts such as proximity, distance, or neighborhood, become less meaningful with increasing dimensionality, due to a loss of contrast of distances [Houle et al., 2010].

When comparing different optimization techniques it is important to mention the dimension range for which this comparison is considered. Many optimization algorithms do not scale in practice for extremely large dimensions \((n \gg 100)\), because their computational complexity and/or memory requirements grow too quickly with \( n \), say cubically and quadratically, respectively.

Non-Separability

A function \( f \) is called separable if the optimum of this function can be achieved by performing \( n \) independent one-dimensional searches along each independent coordinate. Optimization of separable functions can break the curse of dimensionality, because the problem complexity grows linearly with \( n \). A function \( f \) is called non-separable if the optimum of \( f \) cannot be achieved by one-dimensional searches and partially-separable if \( f \) is non-separable and has blocks of coordinates which can be optimized separately.
Many real-world problems are partially-separable, because humans tend to view the world as a structured system/object. Structured view of the world favors extensive use of decomposition, which allows to separately solve sub-problems of a large partially-separable problem. The decomposition is a common approach for complex design and optimization problems, such as design of a new car or airplane. We also often tend to simplify our daily life using decomposition: we first brush our teeth, then we drive a car to our workplace, then we work. However, some people find that it is better to do these things simultaneously.

The separability is probably the first property of the problem which should be checked when analyzing the performance of an optimizer on a benchmark or real-world problem. Some algorithms explicitly or implicitly exploit the separability, and, therefore, usually perform quite well on separable problems. The number of function evaluations to reach the optimum or some target objective value then may scale almost linearly with \( n \). The separability of a problem or any other property of course should be exploited to improve the search when automatically detected. However, most of the well-known benchmark problems are separable and many optimizers have become overfitted [Hawkins, 2004] to this property - they perform well on benchmark problems, but fail on real-world problems which are often non-separable or partially-separable. A simple check against this kind of overfitting is to rotate the search space, such that \( f \) still has the same topology, but is not separable anymore. The overfitting to separability has misled many designers of algorithms and may be viewed as a drawback of such approaches as GAs, PSO, etc (a detailed analysis can be found in [Salomon, 1995, Hansen et al., 2008]). A prospective direction of the research is to adaptively identify and learn the linkage between the variables [Tezuka et al., 2004, Oliwa and Rasheed, 2012], that may allow to exploit the separability when it is detected.

Noisy

A function \( \tilde{f} \) is called noisy if for a given \( x \) different \( \tilde{f}(x) \) values can be observed, perturbed by some random component \( \xi \). Optimization of noisy \( \tilde{f} \) is called noisy optimization, which also often distinguishes two popular cases: i). the variance of the noise decreases to zero when approaching the optimum (multiplicative noise): \( \tilde{f}(x) = f(x)(1 + \xi) \); ii). the variance of the noise is lower bounded (additive noise): \( \tilde{f}(x) = f(x) + \xi \). Both cases are widely presented in real-world problems, both make the optimization more difficult, because the information about the optimized function acquired from one function evaluation is less precise than in the noise-less case.

Theoretical analysis of optimization algorithms in noisy scenario usually becomes much more complicated and often requires to make some specific assumptions on the noise model \( \xi \), hidden behind the evaluations of \( f \). In a noisy environment, first derivatives are usually difficult or impossible to obtain, that limits the using of gradient-based methods. In a recently published paper by Jorge J. More and Stefan M. Wildy "Do You Trust Derivatives or Differences?" [More and Wild, 2012], the authors discuss why it is difficult to reliably compute derivatives or difference approximations for problems with high levels of noise. This fact among others motivated the development of derivative-free algorithms starting from Nelder-Mead algorithm [Nelder and Mead, 1965] and early Evolutionary Algorithms in the 1960s.

Ill-Conditioning

A convex-quadratic function \( f(x) = \frac{1}{2} x^T H x \), where \( H \) is symmetric positive definite,
is called *ill-conditioned* if the condition number of $H$ is much larger than 1. The condition number of $H$ is the ratio between its largest and smallest eigenvalue, and this is also equal to the squared rate between the longest and the shortest principal axes of the ellipsoid $\{ x | x^t H x = 1 \}$. If the condition number is large, then this ellipsoid is very elongated. On an ill-conditioned function the length of search steps, performed in different directions of the decision space may differ by orders of magnitude to produce the same improvements of the objective function.

Ill-conditioned problems are difficult for optimization, because before we learn an appropriate metric we often make too short or too long search steps. Variable-metric methods, such as quasi-Newton methods (e.g., BFGS [Shanno, 1970]), learn the inverse of the Hessian of $f$ and are able to "renormalize" the search steps. However, these algorithms often estimate the gradient using finite-difference methods, which suffer from numerical instability due to round-off errors when the condition number is large, say, larger than $10^4$. Evolutionary variable-metric algorithms, such as CMA-ES [Hansen and Ostermeier, 2001], are usually more stable, because they do not need to estimate the gradient to learn an appropriate metric (e.g., CMA-ES learns the covariance matrix of successful steps, see Section 2.3.2).

**Dynamic**

A function $f(x, t)$ is called *dynamic*, because the objective value for a given $x$ depends on the time-step $t$. It means that the function is changing in time: it may be a simple shift or rotation of the search space or any other change of $f$. In dynamic optimization the goal is to find the global optimum of $f$ for a time-step $t$ and, and in the best case, predict its location for time-steps $T > t$. However, this is often difficult to do, therefore, a less ambitious objective is considered - to track the progression of the optimum in through the space as closely as possible [Branke, 2001, Arnold and Beyer, 2002].

The dynamic version of any $f$ usually becomes more difficult for optimization. This is especially true for the algorithms, which once make the decision about some part of the search space, exclude it forever from the later stages of the search (e.g., search space partition-based methods [Shi and Ólafsson, 2000]).

**Deceptiveness**

There is no clear definition of deceptive functions, a common example of deceptive function is a function, where a local optimum has much larger volume of the basin of attraction than the global optimum (a definition for binary settings can be found in [Goldberg *et al.*, 1992]). As a result, many optimizers can be misled by the local optimum and simply "do not see" the global one. This example, however, is not the best one and is rather related to the multi-modality.

It would be more fair to say that a function cannot be deceptive itself, but only with respect to a given (class of) optimization algorithm(s). A function $f$ can be called *deceptive* for an algorithm $A$, if the properties of $f$ do not correspond to (hidden) assumptions of $A$ on $f$ and this leads to a bad performance of $A$ on $f$ with respect to a set of other optimizers and test problems (e.g., the rank of performance of $A$ drops quickly on $f$). The deceptiveness is closely related to the questions of overfitting of algorithms to a given set of problem, and to No Free Lunch Theorem [Wolpert and Macready, 1997], which states that all algorithms have identically distributed performance when objective functions are
drawn uniformly at random.

Expensive

A function \( f \) is called \textit{expensive} or \textit{costly} if one evaluation of \( f(x) \) for a given \( x \) is more costly in terms of chosen metric (time, money, etc.) than the cost of its optimization (excluding function evaluations) per function evaluation.

The optimization of expensive functions is difficult, because we are constrained by the relatively small number of function evaluations we can use. The fact that functions can be expensive in the sense described above motivates the existence of the whole field of optimization, because, otherwise, a pure random search would be probably a method of choice (the cheapest generator of \( x \) + almost free evaluations of \( f(x) \)).

Multi-objective

A problem is called \textit{multi-objective} if there are \( m \) objectives \( f_i(x) \) for \( i = 1 \ldots m \), which should be simultaneously optimized. If no specific preferences are given, the desirable output of \textit{multi-objective optimization} is a Pareto set of optimal solutions, instead of a single solution, that is usually sufficient for single-objective optimization.

Multi-objective optimization is difficult because the multi-objective problem inherits properties of all single objectives and the optimization itself should take into account the criteria of multi-objective optimality, which are less obvious than for the \textit{single-objective} case. We will discuss the multi-objective optimization in detail in Section 2.4.

2.3 Single-objective Continuous Evolutionary Algorithms

In this Section, we give a detailed description of Evolution Strategies and Covariance Matrix Adaptation Evolution Strategy for single-objective optimization. Despite the general name of this Section, we are biased to the discussion of ESs, mostly because their theoretical foundations are relatively well investigated [Hansen et al., 2013], and they perform quite well on benchmark and real-world continuous optimization problems [Hansen et al., 2010b, Hansen, 2012].

2.3.1 Evolution Strategies (ESs)

Evolution Strategies, introduced by Ingo Rechenberg and Hans-Paul Schwefel in the 1960s and 1970s [Schwefel, 1965, Rechenberg, 1973], are based on the idea of mutations of intermediate size, mainly represented by multivariate normal (Gaussian) mutations. In ESs a parent \( m^t \) of iteration/generation \( t \) generates its \( k \)-th offspring \( x_k \in \mathbb{R}^n \)

\[
x_k = \mathcal{N}(m, \sigma^2 C) = m + \sigma \cdot \mathcal{N}(0, C),
\]

(2.6)

where \( C \in \mathbb{R}^{n \times n} \) is a (positive definite) covariance matrix and \( \sigma \) is a mutation step-size. The parent \( m \) also can be viewed as a mean of current mutation distribution. In each iteration of \((\mu+\lambda)\)-ES, \( \lambda \) offspring individuals are created from existing \( \mu \) parent individuals. In a \textit{plus} selection strategy, \((\mu + \lambda)\), \( \mu \) best out of \( \mu + \lambda \) individuals are chosen to become parents in the next generation. In a \textit{comma} selection strategy, \((\mu, \lambda)\), parents never survive and \( \mu < \lambda \) best offspring are chosen.
Algorithm 2.1: The (1+1)-ES with the 1/5th Rule

1: given \( n \in \mathbb{N}_+ \), \( d_\sigma \approx 1 + n/3 \)
2: initialize \( m^t=0 \in \mathbb{R}^n \), \( \sigma^t=0 > 0 \)
3: repeat
4: \( x_1 = m^t + \sigma^t \mathcal{N}(0, I) \) // mutation
5: \( \sigma^{t+1} \leftarrow \sigma^t \exp(\frac{1}{d_\sigma} (\mathbb{E} [ f(x_1) < f(m^t) ) - 1/5]) \) // step-size update
6: if \( f(x_1) \leq f(m^t) \) then
7: \( m^{t+1} = x_1 \) // selection of \( x_1 \) as the new parent if it is better than \( m^t \)
8: end if
9: \( t \leftarrow t + 1 \)
10: until stopping criterion is met

The research on ESs mainly focuses on questions of adaptation of \( C \) and (usually faster adaptation of) \( \sigma \) during the optimization, and the optimal parametrization of this adaptation. For a recent comprehensible overview of Evolution Strategies, the interested reader is referred to [Hansen et al., 2013].

The 1/5th Success Rule

The 1/5th success rule is a basic step-size control strategy for randomized search, discovered in [Schumer and Steiglitz, 1968] and later, under its current name, in [Rechenberg, 1973]. Algorithm 2.1 implements (1+1)-ES with the 1/5th rule, where at iteration \( t \) a parent \( m^t \) generates an offspring \( x_1 \) (line 4) with Gaussian mutation, defined by the step-size \( \sigma \) and an identity covariance matrix \( C = I \). Setting the covariance matrix to the identity means that the variations of all variables are independent of each other, they are uncorrelated.

The generated offspring replaces its parent (lines 6-7) if it has at least as good fitness as its parent. The update of the mutation step-size (line 5) depends on the empirical probability that \( f(x_1) < f(m^t) \), or in other words, \( \mathbb{E} [ f(x_1) < f(m^t) ] \). This information can be obtained from previous iterations. The target success rate 1/5 is chosen by Rechenberg [Rechenberg, 1973] after investigations of the optimal success rates of (1+1)-ES on Corridor (success rate \( \approx 0.184 \)) and Sphere (success rate \( \approx 0.270 \)) functions for \( n \to \infty \). The use of uncorrelated mutations limits the success of (1+1)-ES on non-linear functions. The preservation of the best solution (elitism) limits its application on multi-modal, noisy and dynamic functions.

Cumulative Step-Size Adaptation

The Cumulative Step-Size Adaptation Evolution Strategy (CSA or CSA-ES) proposed by [Hansen and Ostermeier, 1996] suggests to consider a path the population takes over a number of generations, an evolution path. The evolution path records the sum of consecutive successful steps to make a decision about possible corrections of the step-size. If successful steps are oriented into the same direction, they will sum up and the evolution path
Algorithm 2.2: The $(\mu/\mu_w, \lambda)$-ES with Cumulative Step-Size Adaptation

1: given $n \in \mathbb{N}_+$, $\lambda = 4 + 3 \lfloor \ln n \rfloor$, $\mu = \lfloor \lambda/2 \rfloor$, $w_i = \frac{\ln(\mu+1)-\ln i}{\sum_{j=1}^{\mu}(\ln(\mu+1)-\ln j)}$ for $i = 1 \ldots \mu$, $\mu_w = \frac{1}{\sum_{i=1}^{\mu} w_i}$, $c_\sigma = \frac{\mu_w+2}{\mu_w+\lambda}$, $\sigma = 1 + c_\sigma + 2 \max(0, \sqrt{\frac{\ln n}{\sigma+1}} - 1)$
2: initialize $m^t = 0 \in \mathbb{R}^n$, $\sigma^t = 0 > 0$, $p^t = 0$
3: repeat
4: for $k = 1, \ldots, \lambda$ do
5: $x_k = m^t + \sigma^t \cdot N(0, I)$
6: $f_k = f(x_k)$
7: end for
8: $m^{t+1} = \sum_{i=1}^{\mu} w_i x_i : \lambda$ // the symbol $i : \lambda$ denotes $i$-th best individual on $f$
9: $p^t = (1 - c_\sigma) p^t + \sqrt{c_\sigma(2 - c_\sigma)} \sqrt{\mu_w m^{t+1} - m^t}$
10: $\sigma^{t+1} = \sigma^t \exp(c_\sigma \frac{|p^t - m^t|}{\|N(0, I)\| - 1})$
11: $t \leftarrow t + 1$
12: until stopping criterion is met

will be relatively long in this direction. Since the same distance in this direction can be covered with larger steps, the step-size should be increased. If successful steps are opposite to each other, they sum up making the evolution path relatively short. In this case, the used step-size is too large and should be decreased. The fundamental adaptation principle of CSA is to decorrelate successfully selected mutation steps [Hansen and Ostermeier, 1996]: the adaptation should reduce the difference between the distributions of the actual evolution path and an evolution path under random selection.

Algorithm 2.2 outlines $(\mu/\mu_w, \lambda)$-ES with Cumulative Step-Size Adaptation. Initially chosen mean $m^t$ of the mutation distribution (can be interpreted as an estimation of the optimum) is used to generate $\lambda$ new solutions (lines 4-5) by adding a random uncorrelated Gaussian mutation. These $\lambda$ solutions then should be evaluated on $f$ (line 6). The old mean of the mutation distribution is stored in $m^t$ and a new mean $m^{t+1}$ is computed as a weighted sum of the best $\mu$ parent individuals selected among $\lambda$ generated offspring individuals (line 8). The weights $w$ are used to control the impact of selected individuals, weights are usually higher for better ranked individuals (line 1), but equal weights are also commonly used, leading to an $(\mu/\mu_1, \lambda)$-ES.

To update the evolution path $p_\sigma$ using local information about a successful mutation step $s^{t+1} - m^t$ (line 9), a decay/relaxation factor $c_\sigma$ is used to decrease the importance of previously performed steps with time. A weight factor $c_w = \sqrt{c_\sigma(2 - c_\sigma)} \mu_w$ for the local successful step is set to normalize the variance of $p_\sigma$ by solving the equation $1^2 = (1 - c_\sigma)^2 + c_w^2$. The step-size update rule increases the step-size if the length of the evolution path $p_\sigma$ is longer than the expected length of the evolution path under random selection $\mathbb{E} \|N(0, I)\|$, and decreases otherwise (line 10). Expectation of $\|N(0, I)\|$ is approximated by $\sqrt{\ln(1 - \frac{1}{4n} + \frac{1}{21n^2})}$. A damping parameter $d_\sigma$ controls the change of the step-size.

The step-size adaptation of CSA is similar to the one of the 1/5th rule, but benefits
from using the global information stored in the evolution path. Non-elitist multi-parent selection strategy of \((\mu/\mu_w, \lambda)\)-ES has many advantages over elitist \((1+1)\)-ES (and \((1+\lambda)\)-ES), it is suitable for (moderate) noisy [Arnold and Beyer, 2008] and dynamic optimization [Arnold and Beyer, 2002]. For a recent theoretical analysis of CSA on linear functions, the interested reader is referred to [Chotard et al., 2012].

### 2.3.2 Covariance Matrix Adaptation Evolution Strategy

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) proposed by [Hansen and Ostermeier, 1996, Hansen and Ostermeier, 2001, Hansen et al., 2003] has become a standard for continuous black-box evolutionary optimization. The main advantage over CSA comes from the use of correlated mutations instead of axis-parallel ones. The adaptation of the covariance matrix \(C\) allows to steadily learn appropriate mutation distribution and increase the probability of repeating the successful search steps.

The \((\mu/\mu_w, \lambda)\)-CMA-ES is outlined in Algorithm 2.3: the original CMA-ES is recovered if set \(c^- = 0\) in line 14, otherwise the pseudo-code presents a so-called weighted active CMA-ES [Hansen and Ros, 2010a].

#### Original CMA-ES

The procedure of the adaptation of the step-size \(\sigma\) in CMA-ES is inherited from CSA-ES and is controlled by \(p_a\). However, it should be noted that successful steps are tracked in the space of sampling, i.e., in the isotropic coordinate system defined by principal components of \(C\). To find a pre-image \(z_k\) of a sampled point \(x_k\) in the space of sampling, a transform \(C^{t-\frac{1}{2}} \frac{x_k}{\sigma_t}\) is applied (as for \(m^{t+1} - m^t\) in line 9), where \(C^{t-\frac{1}{2}}\) is symmetric with positive eigenvalues such that \(C^{t-\frac{1}{2}}C^{t-\frac{1}{2}}\) is the inverse of \(C^t\). If \(B \Lambda B^T = C\) is an eigendecomposition into an orthogonal matrix \(B\) and a diagonal matrix \(\Lambda\), we have \(C^{-\frac{1}{2}} = BA^{-\frac{1}{2}}B^T\).

The covariance matrix update consists of two parts (line 14): rank-one update and rank-\(\mu\) update. The rank-one update computes evolution path \(p_c\) of successful moves of the mean \(m^{t+1} - m^t\) of the mutation distribution in the given coordinate system (line 11), adapted in a similar way as for the evolution path \(p_a\) of the step-size. To stall the update of \(p_c\) when \(\sigma\) increases rapidly, a \(h_\sigma\) trigger is used (line 10).

The rank-\(\mu\) update computes a covariance matrix \(C^+\) as a weighted sum of covariances of successful steps of \(\mu\) best individuals (line 12). In a similar way \(C^-\) is computed for \(\mu\) worst individuals to be used for covariance matrix \(C\) update in weighted active CMA-ES. The update of \(C\) itself is a replace of previously accumulated information by a new one with corresponding weights of importance (line 14): \(c_1\) for covariance matrix \(p_c^{t+1}p_c^{t+1T}\) of rank-one update [Hansen and Ostermeier, 2001], \(c_\mu\) for \(C^+\) of rank-\(\mu\) update [Hansen et al., 2003], and \(c^-\) for \(C^-\) of "active" rank-\(\mu\) update [Hansen and Ros, 2010a]. The coefficients \(c_1, c_\mu, c^-\) and \(\alpha^\text{old}\) are defined such that \(c_1 + c_\mu - c^-\alpha^\text{old} \leq 1\).

The "right" parametrization of CMA-ES algorithm indeed is an open problem, most of the parameters of the algorithm were not proven mathematically to be optimal, but are chosen mostly based on algorithm designer’s (including the author, Nikolaus Hansen)
views on robustness of the CMA-ES. This relative robustness is also supported by empirical observations of CMA-ES performance on a few benchmark problems, such as Sphere function and rotated Ellipsoid function.

Weighted active CMA-ES

In the original CMA-ES only best $\mu$ out of $\lambda$ individuals are used to estimate the local covariance matrix of successful steps to increase the probability of successful samples in the next iteration. The information about the remaining (worst $\lambda - \mu$) solutions is used only implicitly during the selection process.

In active $(\mu/\mu_I, \lambda)$-CMA-ES however, it has been shown that the worst solutions can be exploited to reduce the variance of the mutation distribution in unpromising directions [Jastrebski and Arnold, 2006], yielding a performance gain of up to a factor 2 for the active $(\mu/\mu_I, \lambda)$-CMA-ES with no loss of performance on any of tested functions. A recent extension of the $(\mu/\mu_w, \lambda)$-CMA-ES, weighted active CMA-ES [Hansen and Ros, 2010a]
2.3 Single-objective Continuous Evolutionary Algorithms

(referred to as aCMA-ES for brevity) shows comparable improvements on a set of noiseless and noisy functions from the BBOB benchmark suite [Hansen et al., 2010a]. In counterpart, aCMA-ES no longer guarantees the covariance matrix to be positive definite, possibly resulting in algorithmic instability. The instability issues can however be numerically controlled during the search; as a matter of fact they are never observed during the benchmarking on the BBOB test suite.

The main novelty of aCMA-ES is the exploitation of the worst solutions to compute
\[
C_{\mu}^{-1} \mu = \sum_{i=0}^{\mu-1} w_i + 1 \mathbf{y}_{\lambda-i:i} \mathbf{y}_{\lambda-i:i}^T
\]
(line 13), where
\[
\mathbf{y}_{\lambda-i:i} = \frac{C_{t-1/2}^{1/2}(x_{\lambda-i:i} - \mu + 1 + i: \lambda - m)}{\sigma_t}
\]
The covariance matrix estimation of these worst solutions is used to decrease the variance of the mutation distribution along these directions (line 14).

For a more detailed description of the original and active CMA-ES algorithms, the interested reader is referred to [Hansen et al., 2003] and [Hansen and Ros, 2010a], respectively.

CMA-ES with restarts

There are several properties of black-box problems which may lead to a premature convergence of CMA-ES, among the most common are multi-modality and uncertainty. To increase the probability of finding the global optima, IPOP-CMA-ES [Auger and Hansen, 2005] and BIPOP-CMA-ES [Hansen, 2009] restart strategies for CMA-ES were proposed. In IPOP-CMA-ES, CMA-ES is restarted with double the population size each time the stopping criterion is met. BIPOP-CMA-ES has two regimes: one with large population, whose size doubles at each restart and one with small population, whose size is randomly chosen each restart. CMA-ES restarts under the first and the second regimes sequentially, such that two regimes have almost the same budget of function evaluations used so far. We will discuss these strategies in detail and propose two new alternative restart strategies in Section 6.3.3.

Natural Evolution Strategies

Natural Evolution Strategies (NES [Wierstra et al., 2008]) is a "hot" topic in EC community. NES showed the attractiveness of the natural gradient as a general means to derive CMA-like optimization algorithms from the first principles. Natural gradient, first introduced by [Amari, 1998], has several advantages over plain gradient, it allows an algorithm to be invariant w.r.t. orthogonal transformations of the search space and an order-preserved transformations of the objective function. Let the vector \( \theta \) represents all parameters used to define a sampling distribution for new offspring in the search space, then the natural gradient can be used to estimate an ascent in the \( \theta \)-space toward higher expected fitness, i.e., the ascent in the original \( x \)-space and in the space of the algorithm parameters. If \( \theta \) represents mean and covariance matrix of the multivariate normal distribution, a rank-\( \mu \) CMA-like optimization algorithm can be recovered.

In this thesis, we do not use NES algorithms, while being elegant they are usually outperformed by CMA-ES algorithms and the most powerful versions of NES resembles CMA-ES. However, there are several very interesting features discovered for NES in the first
Chapter 2. Evolutionary Algorithms for Continuous Black-Box Optimization

PhD thesis on the subject [Schaul, 2011]: hyper-parameters adaptation through *adaptive sampling* and reuse of already sampled individuals through *importance mixing*. Another interesting study of natural gradient-based optimization is the Information-Geometric Optimization (IGO) framework proposed in [Arnold et al., 2011].

### 2.3.3 Why CMA-ES? Empirical Comparison with other Approaches

In this thesis, we intensively use CMA-ES as a baseline optimization algorithm for comparison and as a starting point to further improve optimization techniques in single- and multi-objective scenario (Section 2.4). It is used both in exploitation (Chapter 4 and Chapter 5) and exploration (Chapter 6) parts of this thesis, and in this Section, we present some arguments to convince the reader that the CMA-ES is a good candidate optimization algorithm.

#### Invariance properties of CMA-ES

The importance of invariances in science has long been acknowledged. In Computer Science in particular, the invariance of an algorithm with respect to a given transformation of the problem domain is a source of robustness, as any theoretical or empirical result that is demonstrated for a given problem instance can be extended to the whole class of problems obtained by applying the transformation. For instance, many bio-inspired optimization algorithms such as tournament-based EAs, PSO, or DE only rely on comparisons of the fitness function, making them invariant under any monotonous transformation of the fitness. From a theoretical perspective, this invariance property is a source of robustness [Gelly et al., 2007]; from an algorithmic perspective, it removes the need to tune the algorithm hyper-parameters according to some (generally unknown) scale of the fitness function.

CMA-ES exhibits the following invariances:

- Invariance against rank-preserving transformation of $f$: if we apply a strictly increasing function $g : \mathbb{R} \mapsto \mathbb{R}$ to $f$, the optimization of $f$ and $g \circ f$ is indifferent whether we minimize, e.g., $f$, $f^3$ or $f \times |f|^2$. As well as in all comparison-based optimization algorithms, the algorithm only depends on the *ranking* of function values (see Algorithm 2.3, line 8). Therefore it is as "easy" to optimize some non-convex or non-smooth functions as well as convex ones (it is also true for all comparison-based optimization algorithms).

- Invariance against angle preserving transformation of the search space (rotation, reflection, translation) if the initial search point(s) are transformed accordingly.

- Scale invariance if the initial scaling, e.g., $\sigma^0$, and the initial search point(s) are chosen accordingly.

The key invariance property which contributes the most into the good performance of the CMA-ES is invariance with respect to the rotation of the search space. Most of
Evolutionary Algorithms lack of this property and are therefore outperformed by CMA-ES on problems, where the algorithm may benefit from the learning of correlations between the variables, e.g., on non-separable problems [Hansen et al., 2011]. Another important property of the CMA-ES is that it is almost parameter-less algorithm, and there is virtually only one parameter suggested to be tuned by the user - population size \( \lambda \). Even for this parameter there is a default value, which depends on the problem dimension \( n \) and is "well-tuned" for unimodal functions (see also [Smit and Eiben, 2010] for offline parameter tuning of CMA-ES).

**Performance on BBOB framework.**

The number of function evaluations used to find a target function value \( f_{\text{target}} \) is a common metric, adopted from the very beginning of the field [Schumer and Steiglitz, 1968, Rechenberg, 1973] for "greedy" evaluation and comparison of EAs. The first International Contest on Evolutionary Optimization (1st ICEO) organized during the International Conference on Evolutionary Computation (ICEC) in 1996 was probably the first attempt to compare different continuous optimizers on a set of benchmark problem in a regime of competition. Differential Evolution managed to finish 3rd among 8 algorithms and after two non-evolutionary optimizers (one exploited function separability), this result definitely boosted the popularity of DE. [Bersini et al., 1996] summarizes the results of the competition and presents a brilliant argumentation why organization of such competitions is important. We are agree with the vast majority of these arguments, which have become even more relevant in the last decade.

A more recent competition of optimizers, Congress on Evolutionary Computation 2005 - Session on Real-Parameter Optimization (CEC-2005), provided a more rigorous investigation of performance of 11 Evolutionary Algorithms on 25 unimodal and multi-modal 10- and 30-dimensional benchmark problems [Suganthan et al., 2005]. A fitness-distance analysis of functions landscapes is given in [Müller and Sbalzarini, 2011]. Figure 2.1 from [Hansen, 2006] shows a comparative performance of 11 tested algorithms, including G-CMA-ES (a version of IPOP-CMA-ES [Auger and Hansen, 2005]). The higher the curve on the graph, the better the algorithm performs. For a given algorithm a value on axis-x (x-value) can be interpreted as a loss factor \( \text{FEs}/\text{FEs}_{\text{best}} \) of number of function evaluations the algorithm has on \( (1 - (y\text{-value})) \cdot 100\% \) of tested benchmark problems.

A relatively good performance of G-CMA-ES on CEC-2005 testbed attracted a lot of attention and in a long term has made CMA-ES "an algorithm to compare with". It has become a standard tool for real-world problem optimization, it also has more than one hundred real-world applications [Hansen, 2012] which can be considered as an empirical validation of its robustness. A detailed statistical analysis of the results of CEC-2005 contest is presented in [García et al., 2009].

COMparing Continuous Optimizers (COCO [Finck et al., 2010]) is a recent and quite successful platform to quantify and compare the performance of optimization algorithms in a scientifically decent and rigorous way. COCO provides means to run an optimization algorithm on a set of benchmark optimization problems, stores the information about evaluated points, post-processes the obtained information and, finally, prepares tables and
Chapter 2. Evolutionary Algorithms for Continuous Black-Box Optimization

Figure 2.1: The graphs show the empirical cumulative distribution function of \( \text{FEs}/\text{FEs}_{\text{best}} \) over different sets of function in dimension 10 (Left) and 30 (Right) for 11 tested Evolutionary Algorithms during the CEC-2005 optimization contest. \( \text{FEs} \) and \( \text{FEs}_{\text{best}} \) denote the expected number of function evaluations to reach the target function value by a given and the best performed algorithm, respectively. Large values of the graphs (the cumulative distribution function) are preferable, they correspond to smaller values of \( \text{FEs}/\text{FEs}_{\text{best}} \).

figures to analyze the performance of the algorithm itself and in comparison with a set of already benchmarked algorithms. COCO has been used for the Black-Box-Optimization-Benchmarking (BBOB [Finck et al., 2010]) workshops that took place during the Genetic and Evolutionary Computation Conference (GECCO) in 2009, 2010 and 2012.

BBOB provides 24 noise-less [Hansen et al., 2009a] and 30 noisy [Hansen et al., 2009b] benchmark problems with different properties: separable, non-separable, unimodal, multimodal, ill-conditioned, deceptive, functions with and without/weak global structure. All functions are defined and can be evaluated over \( \mathbb{R}^n \), while the actual search domain is given as \([-5, 5]^n\). For the majority of functions the global optimum \( \mathbf{x}_{\text{opt}} \) is randomly drawn uniformly in \([-4, 4]^n\). To derive non-separable functions from separable ones and to control the conditioning of the function, linear transformations are applied. Addition non-linear and symmetry breaking transformations are applied to a portion of \( 1/2^n \) of the search space to make some relatively simple functions less regular. All functions are defined for \( n = 2, 3, 5, 10, 20 \) (and \( n = 40 \) which is optional). The goal of optimization of function \( f \) is to reach \( f_{\text{target}} = f_{\text{opt}} + \Delta f \) (\( \Delta f = 10^{-8} \)) function value using as few function evaluations as possible. The optimum function value \( f_{\text{opt}} \) is shifted in \( f \)-space and is different for each function. As an input for optimization of function \( f \), only problem dimension \( n \) and the initial range for decision variables are provided. To collect "sufficient" amount of data and make statistical analyses of results more meaningful, 15 trials/runs of the benchmarked algorithm should be conducted on different instances of function \( f \).

If an algorithm succeed to reach a target precision value \( \Delta f_t = f_{\text{target}} - f_{\text{opt}} \) in a single run, then its runtime (RT) is the number of function evaluations used. If the algorithm fails, it can be restarted and run again. The Expected Running Time (ERT), the expected
2.3 Single-objective Continuous Evolutionary Algorithms

Figure 2.2: Empirical runtime distributions (runtime in number of function evaluations divided by dimension) on all noise-less functions of BBOB-2009 with $f_t \in [100, 10^8]$ in dimension 10. See text for details.

The number of function evaluations to reach a target function value for the first time is used for performance assessment. The ERT is computed as follows [Finck et al., 2010]:

$$ERT(f_{\text{target}}) = \frac{\#\text{FEs}(f_{\text{best}} \geq f_{\text{target}})}{\#\text{succ}},$$

(2.7)

where the $\#\text{FEs}(f_{\text{best}} \geq f_{\text{target}})$ is the number of function evaluations conducted in all trials, while the best function value was not smaller than $f_{\text{target}}$ during the trial. The $\#\text{succ}$ is the number of successful trials.

To summarize the results from a set of benchmark problems Empirical Cumulative Distribution Functions (ECDFs) are used. The ECDF function $F : \mathbb{R} \rightarrow [0, 1]$ is defined for a given set of real-valued data $S$, such that $F(x)$ equals the fraction of elements in $S$ which are smaller than $x$. The function $F$ is monotonous and a lossless representation of the (unordered) set $S$ [Finck et al., 2010].

Each graph in Figure 2.2 depicts the empirical cumulative distribution of RT of the annotated algorithm from BBOB-2009 on all noise-less functions $f_1 - f_{24}$ in dimension 10 [Hansen et al., 2010b]. For each function-$\Delta f_t$-pair, 100 instances (problems) of RT are
generated, where each $\Delta f_t$ is in $\{10^{1.8}, 10^{1.6}, 10^{1.4}, \ldots, 10^{-8}\}$, i.e., $\Delta f_t \in [10^2, 10^{-8}]$. Any given (x-value) in Figure 2.2 indicates a budget of function evaluations used, divided by dimension, to solve a given (y-value) proportion of problems (function-$\Delta f_t$-pairs). The graphs are monotonous by definition. Crosses indicate the maximum number of function evaluations observed for the respective algorithm. Results to the right of a cross are bootstrapped and are only comparable between algorithms with similar maximum number of function evaluations. The limit value to the right indicates the ratio of solved problems. The horizontal distance between graphs represents a difference in runtime for solving the same proportion of problems. The area between two graphs, up to a given y-value, is the average runtime difference (averaged on the log scale), arguably the most useful aggregated performance measure. The best algorithm covers the largest area under its graph [Hansen et al., 2010b].

Figure 2.2 clearly shows that all benchmarked algorithms outperform Monte Carlo search method in a long run, for most of the algorithms the difference becomes significant already after $10^3 n$ function evaluations. BIPOP-CMA-ES outperforms all other algorithms after about $10^3 n$ function evaluations. It should be noted, that the most difficult function-$\Delta f_t$ problems correspond to multi-modal functions and are located above the line of 0.6 (60% of problems solved) on y-axis. The performance of BIPOP-CMA-ES above this line, i.e., on multi-modal functions, can be explained by restart strategies used to increase the probability of finding the global optimum. The BIPOP restart scheme also outperforms IPOP scheme (see IPOP-SEP-CMA-ES) on the most difficult multi-modal functions. All BBOB-2009 algorithms with good performance on multi-modal functions strongly resemble CMA-ES (AMaLGaM IDEA, iAMaLGaM IDEA) or use CMA-ES as a local search (MA-LS-Chain, VNS).

The (1+1)-CMA-ES is a CMA version of (1+1)-ES presented by [Igel et al., 2006]. The elitist (1+1) selection scheme tends to work well on unimodal functions, but suffers from premature convergence on multi-modal functions. State-of-the-art optimization algorithms from non-evolutionary community, derivative-free NEWUOA [Powell, 2006], quasi-Newton BFGS [Shanno, 1970] and GLOBAL method (clustering with local search using BFGS or Nelder-Mead [Pal et al., 2012]), are very competitive with (1+1) CMA-ES algorithm on unimodal functions, but are outperformed by BIPOP-CMA-ES on multi-modal ones. However, these ”classical” algorithms are not invariant to linear transformation of the objective function, that may be viewed as a potential drawback. Nevertheless as well as CMA-ES, they clearly outperform other Evolutionary Algorithms (PSO, GA, EDA-PSO, DE-PSO, G3-PCX) at least by one order of magnitude (this depends on $n$ and functions discussed) in terms of ERT. These observations motivated the exploitation (”greedy”) part of this thesis, which consists of improving CMA-ES on unimodal functions, such that the ”domination” of CMA-ES over ”classical” approaches becomes obvious not only on difficult multi-modal and noisy functions, but also on ”easy” unimodal functions.

Limitations of CMA-ES.

While being a robust black-box optimizer, CMA-ES also has several limitations:

- **Space complexity.** CMA-ES adapts $\frac{n^2+n}{2}$ parameters in the covariance matrix.
during the search. For large $n$, the storing of a single matrix with $n^2$ real-valued elements becomes intractable. On a typical PC this limitation is achieved for $n \approx 10^4$.

- **Time complexity.** CMA-ES performs eigendecomposition $B\Lambda B^T = C$ of the covariance matrix, this procedure has a complexity of $n^3$, but can be postponed until after $n/10$ generations. Finally, the computational complexity scales between $10^{-8}$ and $10^{-7} \times n^2$ [Hansen, 2009], i.e., about 10-100ms per function evaluation for $n = 1000$, that can be considered to be too expensive for computationally very cheap functions.

- **Premature convergence.** With no surprise the algorithm may prematurely convergence to a local optimum on multi-modal functions. Surprisingly, a class of unimodal functions called HappyCat was found recently, where CMA-ES fails to find the minimizer [Beyer and Finck, 2012]. The behavior of CMA-ES on this class of problems is similar to the one observed when evolving at the edge of feasibility of constraint problems, where step-length control may fail because the mutation strength decreases exponentially fast such that the CMA-ES is not able to learn the covariance matrix [Beyer and Finck, 2012].

- **Large number of function evaluations required to find an optimum.** This can be viewed as general remark for all optimization algorithms, but becomes a less abstract limitation, when dealing with expensive optimization problems. Chapter 3 will address this limitation.

To address the above-described limitations of time and space complexities, several versions of CMA-ES with smaller than $n^2$ control parameters have been proposed. A recent baseline in this direction is IPOP-SEP-CMA-ES, also called separable IPOP-CMA-ES, with only $n$ degrees of freedom [Ros and Hansen, 2008]. The algorithm has linear in $n$ time and space complexity and, therefore, is suitable for large scale optimization, performing reasonably well on non-trivial optimization problems [Ros and Hansen, 2008].

The reasons of premature convergence of CMA-ES is an open question, which probably will become clearer by studying alternative variants of online parameter control in CMA-ES. For example, for the case of HappyCat problems, the effect of premature convergence can be reduced if a larger population size $\lambda$ is used.

### 2.4 Multi-objective Evolutionary Algorithms

In this Section, we briefly describe the main concepts of multi-objective search and discuss reasons that make it difficult. Then we recall the history of Multi-objective Evolutionary Algorithms (MOEAs) and, finally, present Non-dominated Sorting Genetic Algorithm II (NSGA-II) and Multi-objective Evolutionary Algorithm (MO-CMA-ES), which will be used later in this study (see Chapter 5 and Section 6.2).
2.4.1 Main Concepts of Multi-objective Optimization

The optimization of the problem with 2 or more (often conflicting) objectives is called \textit{multi-objective optimization}. The multi-objective problem is usually difficult to solve, because it inherits properties of individual objectives. Let us consider more formally, \textit{without loss of generality}, a multi-objective \textit{minimization} problem with \( n \) decision variables (parameters) and \( m \) objectives:

\[
\begin{align*}
\text{Minimize} & \quad y = f(x) = (f_1(x), \ldots, f_m(x)) \\
\text{where} & \quad x = (x_1, \ldots, x_n) \in X \\
& \quad y = (y_1, \ldots, y_m) \in Y
\end{align*}
\tag{2.8}
\]

and where \( x \) is called the \textit{decision vector}, \( X \) is the \textit{decision (parameter) space}, \( y \) is the \textit{objective vector}, and \( Y \) is the \textit{objective space}. The multi-objective problem (MOP) may have constraints, in this case \( X \) should be defined as the \textit{feasible decision space}.

\begin{itemize}
\item \textbf{Definition 1.} (Pareto dominance of vectors). The objective vector \( y_1 \) dominates the objective vector \( y_2 \) \( (y_1 \prec y_2) \) \( \iff \) \( y_j \leq y'_j \) for all \( j \in \{1, \ldots, m\} \) and \( y_k \prec y'_k \) for at least one \( k \in \{1, \ldots, m\} \).
\item \textbf{Definition 2.} (Weak Pareto dominance of vectors). The objective vector \( y_1 \) weakly dominates the objective vector \( y_2 \) \( (y_1 \preceq y_2) \) \( \iff \) \( y_j \leq y'_j \) for all \( j \in \{1, \ldots, m\} \).
\item \textbf{Definition 3.} (Strict Pareto dominance of vectors). The objective vector \( y_1 \) strictly dominates the objective vector \( y_2 \) \( (y_1 \prec y_2) \) \( \iff \) \( y_j < y'_j \) for all \( j \in \{1, \ldots, m\} \).
\item \textbf{Definition 4.} (Incomparability of vectors). The objective vectors \( y_1 \) and \( y_2 \) are incomparable \( (y_1 \parallel y_2) \) \( \iff \) \( y_1 \not\preceq y_2 \) and \( y_2 \not\preceq y_1 \).
\item \textbf{Definition 5.} (Pareto Optimality of vectors). The solution \( x^* \) and its corresponding objective vector \( y^* = f(x^*) \) are Pareto optimal \( \iff \) there exists no \( y \in Y \) such that \( y \prec y^* \).
\item \textbf{Definition 6.} (Approximation set/front). Let \( \mathcal{Y} \subseteq Y \) be a set of objective vectors. \( \mathcal{Y} \) is called \textit{approximation set} or \textit{non-dominated set/front} or \textit{Pareto front approximation} \( \iff \) any vector of \( \mathcal{Y} \) does not weakly dominate any other vector in \( \mathcal{Y} \). Let denote \( \Omega \) as a set of all approximation sets. Vectors of the non-dominated set/front are called \textit{non-dominated vectors}.
\item \textbf{Definition 7.} (Optimal Pareto set/front). \textit{Optimal Pareto set/front} is an approximation set of optimal Pareto solutions \( x^* \) and corresponding vectors \( y^* \).
\item \textbf{Definition 8.} (Better set/front). An approximation set \( A \) is better than an approximation set \( B \) \( (A \succ B) \) \( \iff \) every \( y \in B \) is weakly dominated by at least one \( y^1 \in A \) and \( A \neq B \).
\end{itemize}
All dominance relations defined above for vectors are easily extendable for sets of vectors/solutions, by changing relations between variables to relations between vectors.

Performance Assessment

In order to compare the performance of two optimizers, one can run them to minimize some function \( f \) and depending on the output one can say which one performs better on this particular function. If \( f \) is a single-objective function, then this comparison is quite straightforward: the algorithm which usually (according to some statistical test on several runs) finds better solutions (smaller value of \( f \)) has a better performance on \( f \). If \( f \) is a multi-objective function, the definition of the relation better is now more complicated, because the solution of (unimodal) multi-objective problem is not just one point, but a set of Pareto points. The relation better is defined by Definition 8 [Knowles et al., 2006].

The problem arises when two sets are incomparable, but one needs to distinguish between them and find the more preferable one. It is usually difficult to define this preference criterion, because in multi-objective optimization two objectives are usually considered: i). minimize the distance to the optimal Pareto front; ii). maximize the diversity within the optimal Pareto set approximation. The problem of incomparability becomes even more serious as the number of objectives \( m \) increases [Deb et al., 2001].

To compare output sets of multi-objective optimizers we will follow a widely accepted procedure suggested by [Zitzler et al., 2002b, Knowles et al., 2006] and use Quality Indicators.

Quality Indicators

The underlying idea of quality indicators is to quantify differences between approximation sets, by applying common mathematical metrics. Thus, even for incomparable approximation sets one can say which one is better w.r.t. a given quality indicator.

**Definition 9.** (Unary quality indicator). A unary quality indicator is a function \( I : \Omega \rightarrow \mathbb{R} \), which assigns a real value to any approximation set \( \mathcal{Y} \in \Omega \).

It is important to note that there exists no unary quality measure that is able to indicate whether a Pareto set approximation \( \mathcal{Y}_1 \) is better than a Pareto set approximation \( \mathcal{Y}_2 \) [Zitzler et al., 2002b]. A better quality according to unary indicator in the best case can guarantee that \( \mathcal{Y}_1 \) is not worse than \( \mathcal{Y}_2 \), i.e., incomparable or better.

**Definition 10.** (Binary quality indicator). A binary quality indicator is a function \( I : \Omega \times \Omega \rightarrow \mathbb{R} \), which assigns a real value to any pair of approximation sets \( (\mathcal{Y}_1, \mathcal{Y}_2) \in \Omega \times \Omega \).

Properly designed binary quality indicators, in contrast to unary ones, are capable to indicate whether \( \mathcal{Y}_1 \) is better than \( \mathcal{Y}_2 \).

**Definition 11.** (Pareto-compliant indicator). The indicator \( I : \Omega \rightarrow \mathbb{R} \) is Pareto-compliant \( \iff \) for every pair of approximation sets \( \mathcal{Y}_1 \) and \( \mathcal{Y}_2 \), for which \( \mathcal{Y}_1 \preceq \mathcal{Y}_2 \), \( I(\mathcal{Y}_1) \) is not worse than \( I(\mathcal{Y}_1) \).
The quality indicator is Pareto-compliant if it does not contradict the order induced by the Pareto-dominance relations. Many popular quality indicators are Pareto non-compliant (generational distance (GD), inverted generational distance (IGD), spread, coverage, etc. [Zitzler et al., 2002b]), i.e., better quality of \( Y_1 \) than of \( Y_2 \) is possible even if \( Y_2 \triangleleft Y_1 \). The use of such Pareto non-compliant quality indicators may mislead the performance assessment procedure. Detailed descriptions of other Pareto compliant indicators, In this thesis, we study multi-objective optimization algorithms using the Pareto-compliant hypervolume indicator, such as \( \varepsilon \)-indicator and \( R_2 \) indicator are given in [Zitzler and Künzli, 2004] and [Brockhoff et al., 2012], respectively.

**Pareto Ranking**

The Pareto ranks w.r.t. \( A \subseteq Y \) of the points in \( A \) are iteratively determined. All non-dominated points in \( A \) (denoted \( ndom_1(A) \) or simply \( ndom(A) \)), are given rank 1. The set \( ndom(A) \) is then removed from \( A \); from this reduced set, the non-dominated points (denoted \( ndom_2(A) \)) are given rank 2; the process continues until all points of \( A \) have received a Pareto rank. The Pareto rank of point \( a \in A \) is denoted \( PR(a, A) \).

The sorting of points w.r.t. Pareto ranking criterion is called non-dominated sorting [Goldberg, 1989, Deb et al., 2000].

**Hypervolume Indicator**

The hypervolume of a set of points \( A \) is sometimes also called "S-Metric" [Zitzler and Thiele, 1998]. Let \( a_{ref} \) denote a reference point, dominated by all points in \( A \). The hypervolume of \( A \) is then the volume of the union of the hypercubes defined by one point of the set and \( a_{ref} \). Formally,

\[
H(A) = \text{Volume} \left( \bigcup_{i=1}^{i=m} \text{Rect}(a_i, a_{ref}) \right),
\]

where \( \text{Rect}(a, b) \) is the hyper-rectangle whose diagonal is the segment \([ab]\). It is clear that only the non-dominated points in \( A \) contribute to the hypervolume.

The hypervolume contribution of some non-dominated point \( a \) is defined as the difference between the hypervolume of the whole set \( A \) and that of the set from which \( a \) has been removed:

\[
\Delta H(a, A) = H(A) - H(A \setminus \{a\})
\]

**Survival Selection**

In multi-objective Evolutionary Algorithms \( \mu \) parent individuals generate \( \lambda \) offspring. Since the population size is usually bounded, the question of survival selection of \( \mu \) most promising / best individuals among \( \lambda + \mu \) ones often arises. A common approach is to sort individuals / vectors of a set \( X \) with respect to a total preorder relation \( \triangleleft_X \). This relation is often based on two sorting criteria / quality indicators: i). a Pareto dominance-based indicator such as Pareto Ranking or Strength Pareto approach [Zitzler et al., 2002a]; ii). a crowding-distance measure in NSGA-II [Deb et al., 2000], distance to \( k \)-th nearest neigh-
2.4 Multi-objective Evolutionary Algorithms

bor in SPEA2 [Zitzler et al., 2002a], hypervolume in SMS-EMOA [Emmerich et al., 2005] and MO-CMA-ES [Igel et al., 2007b]. Formally, $\prec_X$ is defined as follows:

\[
x \prec_X y \iff I_1(x, X) < I_1(y, X) \quad \text{// better dominance-based quality}
\quad \text{or} \quad \text{// same dominance-based quality and better second sorting quality} \quad (2.9)
I_1(x, X) = I_1(y, X) \quad \text{and} \quad I_2(x, X) > I_2(y, X)
\]

The $\prec_X$ relation can be defined by a binary quality indicator, e.g., the one where the fitness of an individual $x$ is associated with a "loss in quality" if $x$ is removed from the population [Zitzler and Künzli, 2004].

2.4.2 Multiple Approaches to Multi-objective Optimization

The finding of solution(s) of multi-objective problems usually follows either the preference-based principle or the ideal principle [Deb and Kalyanmoy, 2001, Tusar, 2007], illustrated in Figure 2.3.

Following the preference-based principle (see Figure 2.3-Left), the multi-objective problem $f(x)$ is reformulated into a (series of) single-objective problem(s) $g(x)$ using some preference and/or a priori information about $f(x)$, e.g., the weights used for a weighted aggregation $g(x) = \sum_{i=1}^{m} w_i f_i(x)$. The main advantage of the preference-based principle is that the solution of the multi-objective problem $f(x)$, reformulated as $g(x)$, can be found using one of many available single-objective optimizers. This fact made many preference-based "classical" optimization methods very popular in the 1960s-1990s: i). weighted sum methods with the aggregation as the one described above [Zadeh, 1963]; ii). $\epsilon$-constraint method, where only one objective function is minimized, while other are $\epsilon$ upper-constrained [Marglin, 1967]; iii). goal programming techniques, where usually simplex method or linear programming is used to satisfy Decision Maker’s goals and priorities [Charnes and Cooper, 1961, Tamiz et al., 1998]. Some of these methods have several disadvantages: i). uniformly chosen weight vectors does not lead to a uniform set of Pareto-optimal solutions (e.g., in the weighted sum method); ii). non-convex parts of the Pareto front cannot be found (e.g., in the weighted sum method); iii). the results are very parameter-sensitive (e.g., in the $\epsilon$-constraint method). The main disadvantage of the preference-based principle is that it usually provides only one solution of the multi-objective problem.

The ideal principle suggests to first find all Pareto-optimal solutions of the multi-objective problem (see Figure 2.3-Right), then choose one or several solutions taking into account the preference information. The method is ideal in the sense that it does not require any additional preference information to be given before the optimization. Thus, the preference-based choice of solutions is made with the complete information about the optimal Pareto front (or its approximation) found after the optimization. In practice we usually bound the maximum number of the Pareto-optimal solutions we are interested in.

Most multi-objective Evolutionary Algorithms (MOEAs) follow the ideal principle, while preference-based MOEAs also attract attention in the field. A promising
algorithm in this direction would be a MOEA based on Decomposition (MOEA/D [Zhang and Li, 2007]), which simultaneously optimizes a number of single-objective aggregated optimization subproblems. It has been shown that the weights used for aggregation can be randomly uniformly assigned to each individual of the population in each generation or can be changed periodically during the optimization [Jin et al., 2001a]. The second method combined with ES and CMA-ES usually outperforms the first method on high-dimensional ZDT problems [Jin et al., 2001a].

**Early MOEAs**

The first MOEA algorithm, Vector Evaluated Genetic Algorithm (VEGA [Schaffer, 1985]), was a simple genetic algorithm with a modified selection step, where each sub-population has its own fitness-proportional based selection w.r.t. only one ob-


2.4 Multi-objective Evolutionary Algorithms

jective. Since the VEGA, the *environmental selection* has been recognized as the main difference between most MOEAs and their single-objective analogs. An important step of its understanding was made by Goldberg in his seminal book [Goldberg, 1989], where he discussed a concept of the non-dominated sorting, which was later implemented in the Non-dominated Sorting Genetic Algorithm (NSGA [Srinivas and Deb, 1994]).

The history of early approaches mainly represents the search for a suitable environmental selection mechanism for multi-objective optimization, the key contributors in this direction are the following algorithms:

- **First generation.** VEGA [Schaffer, 1985], Multi-Objective Genetic Algorithm (MOGA [Fonseca et al., 1993]), NSGA [Srinivas and Deb, 1994], Niched-Pareto Genetic Algorithm (NPGA [Horn et al., 1994]).

- **Second generation.** Strength Pareto Evolutionary Algorithm (SPEA [Zitzler and Thiele, 1998]), Pareto Archived Evolution Strategy (PAES [Knowles and Corne, 1999]), Non-dominated Sorting Genetic Algorithm II (NSGA-II [Deb et al., 2000]), Strength Pareto Evolutionary Algorithm 2 (SPEA2 [Zitzler et al., 2002a]).

One of the main achievements of the early approaches is the non-dominated sorting procedure and various secondary sorting criteria.

**Modern MOEAs**

After appropriate environmental selection schemes were found, many single-objective optimizers has been translated to their multi-objective versions:


- **Particle Swarm Optimization.** Multi-objective Particle Swarm Optimization (MOPSO [Mostaghim and Teich, 2003]), MOPSO with Time Variant parameters (TV-MOPSO [Tripathi et al., 2007]).

- **Evolution Strategy.** Multi-objective CMA-ES (MO-CMA-ES [Igel et al., 2007b]).

- **Estimation of Distribution Algorithms.** Multi-objective EDA (MIDEA [Bosman and Thierens, 2005]), Regularity Model-Based Multi-objective EDA (RM-MEDA [Zhang et al., 2008]).

However, not all above-described algorithms are "just" generalized versions of their single-objective variants. The RM-MEDA makes certain assumptions about the smoothness of $f(\mathbf{x})$ and induces from the Karush-Kuhn-Tucker condition (KKT) that the Pareto
set in the decision space represents a piecewise continuous \((m - 1)\)-dimensional manifold, where \(m\) is the number of objectives. As variation operations, RM-MEDA performs sampling of new individuals on \((m - 1)\) largest principal components of distributions of local clusters, detected in the population. Thus, the variation operators are designed to increase the probability of successful sampling, taking into account the multi-objective context.

The Indicator Based Evolution Algorithm (IBEA) [Zitzler and Künzli, 2004] showed that the quality indicator itself can be used as a criterion for the environmental selection. Soon after that, the SMS-EMOA [Emmerich et al., 2005] algorithm demonstrated the advantages of using the hypervolume (\(S\)-measure) as the second sorting criterion in the non-dominated sorting. This allowed to make a step toward Many-objective optimization \((m > 3)\), where the widely used crowding distance measure fails to properly estimate the individual contribution of non-dominated points [Ishibuchi et al., 2008]. However, the time complexity of the computation of the hypervolume indicator grows exponentially fast, and to make the hypervolume-based algorithm computationally suitable for large \(m\), several Monte Carlo simulation schemes to approximate the exact hypervolume values have been proposed (e.g., HypE [Bader and Zitzler, 2011]). There is also certain progress in computation of the exact hypervolume: [Emmerich and Fonseca, 2011] proposes an algorithm for 3-dimensional objective space, which performs faster than the best known algorithm by a factor of \(\sqrt{r}\), where \(r\) is the number of points.

Despite the diversity of existed multi-objective algorithms worthy of scientific study, in this thesis, we focus on MO-CMA-ES because: i). the CMA-ES part of the algorithm was shown to be efficient in the single-objective case; ii). MO-CMA-ES already shows good results comparing to NSGA-II [Igel et al., 2007b]; iii). self-adaptation mechanisms of CMA-ES make it more "sensitive/adaptive" to changes of the environment (see Section 6.2), while most MOEAs suffer from the lack of this crucial property.

2.4.3 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

The Non-dominated Sorting Genetic Algorithm II (NSGA-II [Deb et al., 2000]) is probably the most widely used multi-objective optimization algorithm. The NSGA-II (the second and drastically improved version of NSGA [Srinivas and Deb, 1994]) is an extension of GA for multi-objective optimization, where the \(\prec_X\) preference relation of the environmental selection is defined in the following way.

The dominance-based quality indicator \(I_1\) is defined by Pareto ranking criterion: the sorting of individuals w.r.t. \(I_1\), the non-dominated sorting, yields to several approximation sets/non-dominated fronts. Then, to order the individuals of the same non-dominated front \(\mathcal{Y}\) the second quality indicator \(I_2\), defined by crowding distance is applied. The crowding distance of an individual \(x \in \mathcal{Y}\) is computed as follows:

\[
\text{cd}(x, \mathcal{Y}) = \sum_{i=1}^{m} \frac{f_i(x^+) - f_i(x^-)}{\max(f_i(y) | \forall y \in \mathcal{Y}) - \min(f_i(y) | \forall y \in \mathcal{Y})},
\]

where \(x^+\) and \(x^-\) are right and left neighboring individuals of \(x\) on the objective \(i\).
Algorithm 2.4: ($\mu+\lambda$)-NSGA-II

1: given $\mu \leftarrow 100$, $\lambda \leftarrow 100$, $p_c \leftarrow 0.9$, $p_m \leftarrow 1/n$, $\eta_c \leftarrow 15$, $\eta_m \leftarrow 20$, $t_{\text{tour}} \leftarrow 2$
2: initialize $t \leftarrow 0$, random parent population $Q^t = \emptyset$
3: repeat
4: for $k = 1, \ldots, \lambda$ do
5: repeat
6: $i_1 \leftarrow \text{TournamentSelection}(Q^t, t_{\text{tour}})$
7: $i_2 \leftarrow \text{TournamentSelection}(Q^t, t_{\text{tour}})$
8: until $i_1 = i_2$
9: if $\mathbb{U}(0,1) \leq p_c$ then
10: $\{o_1, o_2\} \leftarrow \text{SBXcrossover}(x_{i_1}, x_{i_2}, \eta_c)$
11: else
12: $o_1 \leftarrow x_{i_1}$; $o_2 \leftarrow x_{i_2}$
13: end if
14: $x'_k \leftarrow \text{PolynomialMutation}(x'_k, p_m, \eta_m)$
15: $a_{k+1}^t \leftarrow \{x'_k, y'_k = f(x'_k)\}$
16: $Q^t \leftarrow Q^t \cup \{a_{k+1}^t\}$
17: end for
18: $Q^t+1 \leftarrow \{Q^t_{\prec} \mid 1 \leq i \leq \mu\}$ // Deterministic Selection according to $\prec_{Q^t}$
19: $t \leftarrow t + 1$
20: until stopping criterion is met

The NSGA-II algorithm is outlined in Algorithm 2.4. In iteration $t = 0$, a population $Q^t = \emptyset$ of $\mu$ individuals is randomly initialized. In each iteration $t$, the population $Q^t$ generates $\lambda$ offspring using variation operators of SBX crossover (with probability $p_c$ and parameter $\eta_c$) and Polynomial mutation (with probability $p_m$ and parameter $\eta_m$) [Deb and Goyal, 1996] (lines 4-15). Tournament-based parent selection is outlined in Algorithm 2.5. Each newly generated decision vector $x'_k$ is evaluated and stored as an individual $a_{k+1}^t$ (lines 14-15). After the procedure of variation, $\mu + \lambda$ parent and offspring individuals are sorted (line 17) w.r.t. the partial preorder relation $\prec_{Q^t}$ defined above. The time complexity analysis for NSGA-II as well as for many other multi-objective optimizers is given in [Jensen, 2003]. By setting $\lambda = 1$ and $I_2$ to the hypervolume indicator, the Algorithm 2.4 recovers ($\mu + 1$)-SMS-EMOA [Emmerich et al., 2005]. In all experiments of this thesis we will always use NSGA-II with the hypervolume indicator as the second sorting criterion $I_2$, also referred to as S-NSGA-II.

2.4.4 Multi-objective CMA-ES (MO-CMA-ES)

Multi-objective CMA-ES (MO-CMA-ES) proposed by [Igel et al., 2007b, Igel et al., 2007a] is based on CMA-ES adaptive paradigm to handle multiple objectives. MO-CMA-ES involves a set of $\mu$ $(1 + 1)$-CMA-ES algorithms [Igel et al., 2006], each of which performs step-size and covariance matrix updates based on its own evolution path, and a Pareto-based survival
Algorithm 2.5: TournamentSelection

1: given $t_{\text{tour}}$  // tournament size
2: best $\leftarrow U(1, |A|)$
3: for $k = 2, \ldots, t_{\text{tour}}$ do
4:   $i_{\text{cur}} \leftarrow U(1, |A|)$
5:   if $i_{\text{cur}} < \text{best}$ then
6:      best $\leftarrow i_{\text{cur}}$
7:   end if
8: end for
9: return best

The selection mechanism that selects $\mu$ individuals from the population of size $\mu + \lambda$ built in one iteration of the algorithm.

The $(\mu + \lambda)$-MO-CMA-ES (also referred to as $\mu_{\text{MO}} \times (1+1)$-MO-CMA-ES) is outlined in Algorithm 2.6. In iteration $t = 0$, a population $Q_t^0$ of $\mu$ individual $(1+1)$-CMA-ES is randomly initialized. In each iteration $t$, the population $Q_t^t$ generates $\lambda$ offspring $(1+1)$-CMA-ES. To generate $k$-th offspring a parent $a_{i_k}^t$ with index $i_k$ is selected according to ParentSelection scheme (line 5). There are three baseline versions of MO-CMA-ES:

- **Generational** $(\mu + \mu)$-MO-CMA-ES. If $\lambda = \mu$ and each parent $i_k$ generates exactly one offspring, i.e., $i_k = k$.

- **Steady-State** $(\mu + 1)$-MO-CMA-ES or SS-MO-CMA-ES. If $\lambda = 1$ and the parent $i_k$ is randomly uniformly drawn from $\mu$ parents, i.e., $i_k \in U(1, |Q_t^t|)$.

- **Steady-State** $(\mu \prec + 1)$-MO-CMA-ES. If $\lambda = 1$ and the parent $i_k$ is selected among non-dominated points, i.e., $i_k \in U(1, \text{ndom} |Q_t^t|)$.

All parameters of the $k$-th $(1+1)$-CMA-ES including its mean of the mutation distribution $m_k = x_k$, its objective vector $y_k = f(x_k)$, its step-size $\sigma_k$, computed probability of success $\bar{p}_{\text{succ},k}$ and evolution path of the step-size $p_k$ are encapsulated in an element $a_k$, which is called *individual*. When at iteration $t$ an individual $a_{i_k}^t$ is selected for mating it is just copied to an offspring individual $a_{i_k}^{n+1}$ (lines 5 - 6). The mating of $a_{i_k}^{n+1}$ consists of Gaussian mutation of its decision vector $x_{i_k}^t$ (line 7). Then the fitness of newly generated offspring is computed (line 8) and it is added to the population $Q_t^t$ (line 9). In the mating loop (4-9) $\lambda$ new offspring individuals are added to the population, where each offspring is a simple copy of its parent with the mutated decision vector $x_{i_k}^{n+1}$ and the corresponding objective vector $y_{i_k}^{n+1}$.

The augmented population of parents and offspring is truncated to size $\mu$ after sorting w.r.t. $\prec_{Q^t}$, defined in MO-CMA-ES as the non-dominated sorting + the hypervolume contribution indicator as the second sorting criterion (line 22). Before that, the CMA-ES parameters of both parent and offspring are updated depending on their success $\text{succ}_{Q^t}$ with respect to the parent and the population $Q_t^t$. While the success measure $\text{succ}_{Q^t}$ can
be defined in different ways, in original MO-CMA-ES it is defined as the $\prec_q$ preference. Parent’s and offspring’s success rates $p^{t+1}_{\text{succ},k}$ and $p^t_{\text{succ},k}$ are updated at each iteration using some decay factor $c_p$ (line 12). In a similar way to that of the 1/5-th rule, the step-sizes of the parent and offspring are updated in line 13. Thus, if it survives, parent $a^{n}_{k}$ will have an updated success rate and mutation step-size, while offspring $a^{n+1}_{k}$ will also have an updated evolution path $p^{n+1}_{c,k}$ and an updated covariance matrix $C^{n+1}_{k}$ (lines 15-19), which account the information from the last mutation step.

The relatively good convergence of the MO-CMA-ES is due to its (1+1)-CMA-ES part, while the diversity/spread preservation is due to its hypervolume-based environmental selection procedure. There is a less strict variant of $\text{succ}_q$ success measure than $\prec_q$ [Voß et al., 2010], where the offspring is successful if it is selected to the new population.
Chapter 2. Evolutionary Algorithms for Continuous Black-Box Optimization

\( Q_t \), more formally:

\[
\text{succ}_{Q^t} (a^t_i, a^{t+1}_k) = \begin{cases} 
1 & \text{if } a^{t+1}_k \in Q^{t+1} \\
0 & \text{otherwise} 
\end{cases} 
\] (2.11)

Another interesting extension of MO-CMA-ES is to use some recombination procedure for \((1+1)\)-CMA-ES algorithms [Voß et al., 2009]. The recombination consists of sharing the covariance matrix information between the neighborhood individuals, such that individuals with closer Mahalanobis distance in the decision space have larger weights of contribution in the recombination.

### 2.4.5 Benchmarking

Many ways of measuring the performance of multi-objective Evolutionary Algorithms have been proposed. In this study, we follow the guidelines of [Knowles et al., 2006] and use only Pareto-compliant quality indicators, more particularly the widely used hypervolume indicator \( H \). Let \( P \) be a \( \mu \)-size approximation of Pareto front and let \( P^* \) be the approximate \( \mu \)-optimal distribution of optimal Pareto points [Auger et al., 2009]. The approximation error of the Pareto front is defined by \( \Delta H(P^*, P) = H(P^*) - H(P) \). The optimization goal is to minimize \( \Delta H(P^*, P) \) to have a better approximation of the optimal Pareto front of the initial multi-objective function \( f(x) \).

The most popular sets of benchmark problems in the field are: the bi-objective ZDT problems [Zitzler et al., 2000], the scalable DTLZ problems [Deb et al., 2001] and the WFG problems with a wide variety of characteristics [Huband et al., 2005]. ZDT problems are often considered to be too easy, e.g., the Pareto front is located on a boundary of the box-constrained decision space, this could bias the optimization process (see Section 6.2.2.2). They are also separable, and this is more advantageous for MOEAs with separable variation operators (NSGA-II, SPEA2, etc.). To make these problems more reliable, the IHR problems were proposed [Igel et al., 2007b] as the shifted and rotated versions of the ZDT problems. Another interesting set of benchmark problems, inspired by the strategies for constructing test problems proposed in [Okabe et al., 2004], is the LZ problems [Li and Zhang, 2009] with complicated Pareto fronts in decision space. In this thesis, we will benchmark MOEAs on ZDT, IHR and LZ problems.

### 2.5 Discussion

The historical overview of Evolutionary Computation presented in this Chapter is itself an evolution of approaches to optimization. This evolution is a constant fight against the complexity of the real-world that we try to conquer, solving more and more complex problems every day. Since the diversity of these problems is growing, a source of robustness as an ability to adequately solve them with a smallest number of changes in the used optimization technique has become an attractive property. The CMA-ES algorithm clearly demonstrates such robustness, originated in invariance properties of the CMA-ES with respect to rank-preserving transformations of the objective function and with respect to
the rotation of the search space. The empirical observations confirms that CMA-ES overall outperforms other techniques on a wide set of black-box single-objective optimization problems. While we do not observe in the literature such a clear "domination" (and vice versa) of MO-CMA-ES on multi-objective problems, we suppose that its better adaptation to the multi-objective context is worth of scientific study, while its (1+1)-CMA-ES part is already quite powerful. This research might lead to a more general CMA-like approach, derived from the first principles, which would work equally good on single- and multi-objective problems.

In this Chapter, we analyzed Evolutionary Algorithms separately for single- and multi-objective optimization. While many multi-objective optimizers may be recovered by changing the environmental selection of their single-objective analogs, the fundamental difference of these two cases may be more questionable. It would be interesting to know if a nature-driven "evolution of everything" (in a very global scale) exists, then whether this evolution is multi-objective or single-objective, but passed through the multi-objectivization? The multi-objectivization [Knowles et al., 2001] consists in replacing the original objective by a set of objectives, or adding some new objectives to the original one. In certain cases multi-objectivization is shown to be favorable for the search of the global optimum [Knowles et al., 2001]. We also might ask oursewels whether, e.g., the company which produces and sells cars is a multi-objective problem driven unit or just single-objective profit-oriented one, passed through the multi-objectivization? We suppose that the answers depend very much on the scale with which we look at these questions. While the multi-objective case is a more general one and should be viewed as our final goal, we believe that most of colors to draw a picture of a feature general optimizer are already available in the single-objective case.
Chapter 3

Surrogate-assisted Evolutionary Optimization

Much learning does not teach understanding.
Heraclitus, On the Univers.

In a black-box optimization scenario, the only available information about the objective function $f$ comes from the evaluated candidate solutions. It is often assumed that the optimization algorithm estimates $\lambda$ new candidate solutions at each iteration, then, updates its own hyper-parameters $\theta$ (e.g., $m, \sigma$ and $C$ for CMA-ES) in such a way that the probability of sampling solutions with better fitness increases in a short and/or long term. One important thing here is that the algorithm often "forgets" previously evaluated solutions, assuming that all important information is already stored in the $\theta$ hyper-parameters. However, the latter assumption is usually too strong, and most algorithms actually lose very important information, taking into account only the recently evaluated candidate solutions. This is a fundamental basis for learning an approximation or surrogate model $\hat{f}$ of $f$ from a set of selected previously evaluated individuals, called training points/individuals. Thus, in principle all previously evaluated individuals can be checked and selected to build $\hat{f}$. The surrogate model $\hat{f}$ later can be used to give hints about where promising candidate solutions are located. An extreme case is when $\hat{f}$ replaces the original $f$ and is used for direct optimization.

In this Chapter, we review the state-of-the-art techniques for surrogate-assisted evolutionary optimization. First we analyze recent trends in Machine Learning (Section 3.1) and describe the most popular techniques used for surrogate model learning. We focus on Support Vector Machines (SVMs) in Section 3.2, because we will use SVMs in this thesis to build surrogate models with nice invariance properties, such as invariance to rank-preserving transformation of the objective function. After the surrogate model has been built, it can be exploited in many different ways: in Section 3.3, we review the state-of-the-art techniques used for surrogate model control and exploitation in single- and multi-objective scenarios, focusing on Evolutionary Strategies and CMA-ES. Finally, in Section 3.4, we conclude the Chapter and discuss some prospective directions of the surrogate-assisted search.

3.1 Surrogate Models

Various surrogate modeling techniques have been proposed to replace the actual expensive simulations or experiments by cheap surrogate models. Figure 3.1 illustrates the history of
publications reporting the use of the major surrogate techniques. Data was obtained using Google Scholar [Google, 2012] by a methodology proposed in [Viana and Haftka, 2008]. For each technique we searched articles with chosen keywords, e.g., for the response surface method (also referred to as Polynomial Regression (PR)) we used the following query: approximation OR metamodel OR regression OR prediction OR surrogate AND "response surface". Figure 3.1-Left illustrates the number of papers published in all fields. One can observe that the traditional polynomial response surface approach and Kriging approach grow every year in average by 10-15% in terms of number of publications per year. This can be viewed as a factor of growing of the field of theoretical and applied machine learning and optimization. One can also clearly observe the "boom" of artificial neural networks (ANN) in the late 1980s and the early 1990s, and a very similar "boom" of support vector machines (SVMs) 10 years later. The growth of popularity of ANN clearly stagnates for the last 5 years, while SVM might become the most popular approach in machine learning in the next few years.

To measure the popularity of surrogate techniques used with Evolutionary Algorithms (see Figure 3.1-Right) we searched using queries such as: "evolutionary algorithms" and "neural network". While the ranking of popularity of surrogate techniques is the same, the publications often correspond to the tuning of surrogate models (e.g., structures and weights of ANN, hyper-parameters of SVM) and not the surrogate-assisted optimization. Many publications only mention surrogate techniques but do not use them, other use them not for optimization but for, e.g., classification. Therefore, in order to estimate the popularity of techniques for surrogate-assisted optimization the absolute numbers should be divided by a factor from 10 to 50 for ANN and SVM and by some smaller factor for response surface and Kriging.

In this Section, we will describe only major techniques for surrogate model learning: Polynomial Regression in Section 3.1.1, Gaussian Processes in Section 3.1.2, Artificial Neural Networks in Section 3.1.3, Radial Basis Functions in Section 3.1.4 and Support Vector
Machines, which we will discuss in detail in Section 3.2. Many interesting approaches, such as Generalized Additive Models (GAMs [Hastie and Tibshirani, 1990]), Multivariate Adaptive Regression Splines (MARS [Friedman, 1991]) and Kernel Partial Least Squares (KPLS [Rosipal and Trejo, 2002]), are omitted for the sake of space.

### 3.1 Polynomial Regression (PR)

Polynomial regression is a form of linear regression which fits a non-linear model to a training set \( D = \{(x_i, y_i), i = 1, \ldots, \ell\} \), where \( x_i \in \mathbb{R}^n, y_i \in \mathbb{R} \). A PR model that is linear in \( \beta \) can be expressed as follows [Myers, 1990, Atkeson et al., 1997]:

\[
\hat{f} = \tilde{x}^T \beta, \tag{3.1}
\]

where \( \beta \) defines the complexity of the model. Linear (respectively, quadratic) polynomial regression models can be recovered by setting \( \tilde{x} = (x_1, \ldots, x_n, 1) \) (respectively, \( \tilde{x} = (x_1^2, \ldots, x_n^2, x_1 x_2, \ldots, x_{n-1} x_n, x_1, \ldots, x_n, 1) \)). The training points can be collected in a matrix equation

\[
\begin{pmatrix} (W \tilde{X})^T W \tilde{X} \end{pmatrix} \beta = (W \tilde{X})^T W y, \tag{3.2}
\]

where \( \tilde{X} = (\tilde{x}_1, \ldots, \tilde{x}_\ell)^T \) is a matrix of \( \tilde{x} \) parameters of \( \ell \) training points, \( y = (y_1, \ldots, y_\ell)^T \), and \( W = \text{diag}(1) \in \mathbb{R}^{\ell \times \ell} \) defines weights of training points. In global polynomial regression the weights of all training points are equal, the locally weighted regression with different weights is described in Section 3.3.2.2.

Estimating the parameters \( \beta \) using regression amounts to minimizing the least square loss criterion

\[
A(\mathbf{q}) = \sum_{i=1}^\ell \left[ (\hat{f}(x_i, \beta) - y_i)^2 \right] \tag{3.3}
\]

by solving (3.2) using least square or gradient-based methods.

The Response Surface Methodology (RSM) proposed by [Box and Wilson, 1951] suggested to use quadratic PR models to perform a sequence of designed experiments (DOE) to obtain an optimal response from an analyzed function \( f \). The terms PR and RSM are often used interchangeably.

If the target function \( f \) is smooth and its complexity is a priori known, then PR models may be a good choice. In practice, they are usually good for, e.g., quadratic or cubic landscapes, but often are not suitable for more complex multi-modal ones. An important drawback of PR models is that the computational complexity to find exact solution of the least square problems scales with \( O(n_{\text{param}}^3) \), where \( n_{\text{param}} \) is the number of parameters of the model that scales itself with \( O(n^2) \) for quadratic PR model. Therefore, the learning of quadratic PR model scales with \( O(n^6) \), limiting the range of application to problems with \( n \lesssim 15 \) [Kern et al., 2007].
3.1.2 Gaussian Processes (GP)

Kriging, also referred to as Gaussian Process (GP) regression, was originally developed in geostatistics by a South African mining engineer called Danie Krige in the early 1950s and was further developed by [Matheron, 1963]. Kriging models are rather global than local surrogate models, hence more suited for larger experimental areas than the areas typically used in low-order polynomial regression models [Kleijnen, 2009].

A Gaussian process is a collection of random variables, any finite number of which have (consistent) Gaussian distributions [Rasmussen, 2004]. Therefore, the vector of function values \( y_\ell \) of training points can be viewed as only one sample of a multivariate Gaussian distribution with joint probability density \( p(y_\ell | X_\ell) \), where \( X_\ell \) is the set of training points. Thus, the predictive distribution for a new test point \( x_{\ell+1} \) is obtained from \( \ell + 1 \)-dimensional joint Gaussian distribution for the outputs of the \( \ell \) training points and the test point, by conditioning on the observed targets in the training set:

\[
p(y_{\ell+1} | X_{\ell+1}, y_\ell) = \frac{p(y_{\ell+1} | X_{\ell+1})}{p(y_\ell | X_\ell)}, \tag{3.4}
\]

It should be noted that the dimensionality of each probability density here equals the number of data points. The (3.4) can be simplified to [Büche et al., 2004]

\[
p(y_{\ell+1} | X_{\ell+1}, y_\ell) \propto \exp \left( -\frac{1}{2} \frac{(y_{\ell+1} - \hat{y}_{\ell+1})^2}{\sigma_{\ell+1}^2} \right), \tag{3.5}
\]

a univariate Gaussian with mean and variance given by

\[
\hat{y}_{\ell+1} = k^T C_\ell^{-1} y_\ell, \tag{3.6}
\]

\[
\sigma_{\ell+1}^2 = \kappa - k^T C_\ell^{-1} k, \tag{3.7}
\]

where the smooth covariance function between two point \( x_i \) and \( x_j \) is defined as

\[
C(x_i, x_j) = \theta_1 \left( -\frac{1}{2} \sum_{i=1}^{n} \frac{(x_i,k - x_j,k)^2}{r_k^2} \right) + \theta_2 + \delta_{ij} \theta_3. \tag{3.8}
\]

The covariance vector \( k \) and variance \( \kappa \) can be expressed in terms of the covariance function as

\[
k_i = C(x_i, x_{\ell+1}), \quad i = 1, \ldots, \ell, \tag{3.9}
\]

\[
\kappa = C(x_{\ell+1}, x_{\ell+1}) = \theta_1 + \theta_2 + \theta_3. \tag{3.10}
\]

The optimization of hyper-parameters \( \theta = \{\theta_1, \theta_2, \theta_3, r_1, r_2, \ldots, r_\ell\} \) is a multi-modal function, therefore some combination of gradient-based (e.g., BFGS) and Evolutionary Algorithms (e.g., CMA-ES) can be used [Büche et al., 2004].
3.1 Surrogate Models

The computational complexity of model training scales with $O(N \ell^3 + n \ell^2)$, where $N \geq 1$ denotes the number of iterations spent on hyper-parameters optimization by a gradient-based algorithm [Emmerich et al., 2006b]. Each prediction of $\hat{f}$ scales with $O(\ell^2)$ in time and computation of $\sigma$ scales with $O(\ell)$. Therefore, if $\ell$ is chosen very conservatively to be linear in $n$, then the computational complexity scales at least with $O(n^3)$.

3.1.3 Artificial Neural Network (ANN)

The perceptron [Rosenblatt, 1958] is a linear classifier whose output is a function applied on the weighted sum of its $M$ inputs:

$$\hat{f}(x, w) = f_a \left( w_0 + \sum_{j=1}^{M} w_j x_j \right),$$  \hspace{1cm} (3.11)

where $x$ is the input with a corresponding weight vector $w$, $w_0$ is a constant bias and $f_a : \mathbb{R} \mapsto \mathbb{R}$ the non-linear activation function.

A basic artificial neural network in the form of a Multilayer Perceptron (MLP) can be described as a perceptron whose input is the output of other perceptrons. The overall model of an ANN with one hidden layer of $M$ perceptrons and an output layer with $K$ outputs for $k = 1, \ldots, K$ takes the form [Bishop, 2006]:

$$\hat{f}_k(x, w) = f_a \left( \sum_{j=1}^{M} w_{kj}^{(2)} f_a \left( \sum_{i=1}^{n} w_{ji}^{(1)} x_i + w_{j0}^{(1)} \right) + w_{k0}^{(2)} \right),$$  \hspace{1cm} (3.12)

where $w$ is the vector of weights and the superscripts (1) and (2) indicate that the corresponding parameters are in the first and the second layers of the network, respectively. The activation function is often defined by a logistic sigmoid function $f_a(a) = 1/(1 + \exp(-a))$.

Since it is supposed that each training point $x_i$ may have $K$ values $f_1(x_i), \ldots, f_K(x_i)$, after optimizing the neural network we want to find the weight vector $w$ that minimizes the mean square error on training points:

$$E = \frac{1}{\ell} \sum_{i=1}^{\ell} \sum_{k=1}^{K} (\hat{f}_k(x_i) - f_k(x_i))$$  \hspace{1cm} (3.13)

The main drawback of ANN is that the minimization of (3.13) during the model learning is a multi-modal problem, therefore popular approaches such as back-propagation [Bryson and Ho, 1975] and Newton’s method may prematurely converge to local optima. An open question is how to optimally choose the structure of ANN: the number of hidden layers and perceptrons in each layer, connections between perceptrons. A state-of-the-art Evolutionary Algorithm HyperNEAT for evolving large-scale neural networks (simultaneous optimization of weights and structure, see also NEAT [Stanley and Miikkulainen, 2002]) with millions of connections can be found in [Stanley et al., 2009].
3.1.4 Radial Basis Functions (RBFs)

The method of Radial Basis Functions (RBFs) [Orr, 1996] approximates $f$ by using a linear combination of $N_{RBF}$ (usually $N_{RBF} = \ell$) radially symmetric functions $h_i(x)$

$$\hat{f}(x) = \sum_{i=1}^{N_{RBF}} w_i h_i(x),$$

(3.14)

where $w$ is the weight vector. Similarly to the polynomial regression optimal weights can be found by solving

$$\hat{w} = A^{-1}H^T\hat{y}$$

(3.15)

where $A^{-1}$, the variance matrix is $A^{-1} = (H^TH)^{-1}$, where

$$H = \begin{bmatrix}
h_1(x_1) & \cdots & h_{N_{RBF}}(x_1) \\
\vdots & \ddots & \vdots \\
h_1(x_\ell) & \cdots & h_{N_{RBF}}(x_\ell)
\end{bmatrix}$$

(3.16)

The variance matrix can be used to estimate the variance $\sigma^2$ of an independent error $\epsilon_i$ w.r.t. $h_i(x)$. The kernel function $h_i(x)$ (see Section 3.2 for definition of kernels) can be defined by polynomial kernel of degree $d$ and linear splines, but usually is defined by Gaussian RBF [Bishop, 2006]. It is traditionally assumed that RBF networks have only one hidden layer [Orr, 1996], sharing some similarities with ANNs with one hidden layer.

As well as in GP, the computational complexity of RBG methods scales at least with $O(\ell^3)$ because of the inversion of $A$.

Because Support Vector Machines have been the preferred surrogate models in this thesis, the following Section details several variants of SVM.

3.2 Support Vector Machine (SVM)

Support Vector Machines (SVMs) and kernel-based learning methods have become popular since the 2000s for solving problems in classification, regression, and novelty detection. In this Section, we demonstrate the mathematical beauty of the SVM approach first for the case of simple linear classification and then show its extension to more complex tasks such as regression and ranking, which can be used for surrogate model learning.

3.2.1 Classification SVM

SVMs were initially proposed for linear classification by Vladimir Vapnik in [Vapnik, 1995], but have been developed by him together with Alexey Chervonenkis since the 1960s [Vapnik and Chervonenkis, 1971]. An important property of SVMs that made them popular is that the prediction model learning is a convex optimization problem, and so any local optimum is also a global optimum.
3.2 Support Vector Machine (SVM)

![Figure 3.2: Linear separating hyperplanes for separable case of classification of yellow (class '+1') and green (class '-1') points.](image)

3.2.1.1 Linear Separable Classification SVM

Considering a two-class training set $D = \{(x_i, y_i), i = 1, \ldots, \ell = N\}$, where $x_i \in \mathbb{R}^n$, $y_i \in \{-1, +1\}$, classification problem consists of finding $f_{\text{classSVM}}$ which can correctly classify all training points and predict classes of unseen test points.

**The Optimal Hyperplane** Suppose the training data $D$ can be separated by a hyperplane

$$\langle w, x \rangle - b = 0,$$  \hspace{1cm} (3.17)

where $w$ is the norm to the hyperplane. Then, there are infinitely many such hyperplanes (see, e.g., $W_1$, $W_2$ and $W_3$ in Figure 3.2-a), which separate the set of vectors $D$ such that all points of one class (e.g., class '+1' of yellow points) lie on one side of the hyperplane (e.g., on the left side), while all points of another class (class '-1' of green points) lie on the opposite side of the hyperplane (on the right side).

A hyperplane is called the optimal separating hyperplane or the maximum margin hyperplane if the set of vectors $D$ is separable without error and the distance between the closest vector to the hyperplane is maximal. More formally the separating hyperplane can be described as follows (after renormalization):

$$\begin{cases} 
\langle w, x_i \rangle - b \geq +1 & \text{if } y_i = +1 \\
\langle w, x_i \rangle - b \leq -1 & \text{if } y_i = -1,
\end{cases}$$  \hspace{1cm} (3.18)

or in a more compact notation as $y_i \left[\langle w, x_i \rangle - b\right] \geq 1$. It can be shown that the optimal hyperplane should satisfy the described above condition and minimize

$$\Phi(w) = \frac{1}{2} \langle w, w \rangle = \frac{1}{2}||w||^2,$$  \hspace{1cm} (3.19)
that corresponds to the maximization of the margin $2/\|w\|$ between two classes of points.

**Constructing The Optimal Hyperplane** To find the optimal hyperplane one has to solve the following *primal* quadratic programming problem:

$$\begin{align*}
\text{Minimize} \quad & \frac{1}{2} \|w\|^2 \\
\text{subject to} \quad & y_i [\langle w, x_i \rangle - b] \geq 1 \quad i = 1, \ldots, \ell
\end{align*}$$

(3.20)

Lagrange multipliers $\alpha_i, i = 1, \ldots, \ell$ can be introduced for each of the inequality constraints of (3.20) to i). replace the original constraints to constraints on the Lagrange multipliers themselves, which are much easier to handle; ii). formulate the problem only in terms of dot products between vectors, which will allow to generalize SVMs to non-linear classification and regression. The solution of (3.20) is given by the saddle point of the *primal* Lagrangian

$$L_P(w, b, \alpha) = \frac{1}{2} \langle w, w \rangle - \sum_{i=1}^\ell \alpha_i \{ y_i [\langle w, x_i \rangle - b] - 1 \},$$

(3.21)

The Lagrangian has to be minimized with respect to $w$ and $b$ and maximized with respect to $\alpha_i > 0$. At the saddle point, the solutions $w_0, b_0,$ and $\alpha^0$ should satisfy the Karush-Kuhn-Tucker (KKT) conditions [Fletcher, 1987]

$$\begin{align*}
\frac{\partial L_P}{\partial w}(w_0, b_0, \alpha^0) &= w_0 - \sum_{i=1}^\ell y_i \alpha^0_i x_i = 0 \\
\frac{\partial L_P}{\partial b}(w_0, b_0, \alpha^0) &= \sum_{i=1}^\ell \alpha^0_i y_i = 0
\end{align*}$$

(3.22) \quad (3.23)

From (3.22) and (3.23) we observe that the optimal hyperplane $w_0$ is a linear combination of the training vectors:

$$w_0 = \sum_{i=1}^\ell y_i \alpha^0_i x_i, \quad \alpha^0_i \geq 0, \quad i = 1, \ldots, \ell$$

(3.24)

All points $x_i$ with non-zero coefficients $\alpha^0_i$ are called *support vectors*. In Figure 3.2-b support vectors have additional black dots inside the circle.

We can substitute equalities (3.22) and (3.23) into the *primal* form of Lagrangian (3.21) and obtain a particular *dual* form, called the Wolfe dual [Fletcher, 1987]:

$$\begin{align*}
\text{Maximize} \quad & L_D(\alpha) = \sum_{i=1}^\ell \alpha_i - \frac{1}{2} \sum_{i=1}^\ell \alpha_i \alpha_j y_i y_j \langle x_i, x_j \rangle \\
\text{subject to} \quad & \alpha_i \geq 0, \quad i = 1, \ldots, \ell
\end{align*}$$

(3.25)
3.2 Support Vector Machine (SVM)

Figure 3.3: Linear separating hyperplanes for non-separable case of classification of yellow (class ’+1’) and green (class ’-1’) points.

The optimal solution $\alpha^0$ of (3.25) can be used to predict the class of a point $x$ computing the decision function

$$\hat{f}_{\text{classSVM}}(x) = \text{sign} \left( \sum_{i=1}^{\ell} y_i \alpha_i^0 (x, x_i) - b_0 \right),$$

(3.26)

where $b_0$ is the threshold

$$b_0 = \frac{1}{2} \left[ \langle w_0, x_{+1} \rangle + \langle w_0, x_{-1} \rangle \right],$$

(3.27)

where $x_{+1}$ (respectively, $x_{-1}$) is any support vector belonging to the class ’+1’ (respectively, to the class ’-1’).

3.2.1.2 Linear Soft Margin Classification SVM

The described above classification SVM is called hard margin SVM, because it assumes that the training data is separable by a linear classifier, and, therefore, the optimum solution leads to no error of classification in the training set. In the case where the training set is linearly non-separable, i.e., some of training points cannot be correctly classified (see, e.g., the yellow point in the left bottom corner in Figure 3.3), non-negative slack variables $\xi_i \geq 0$ are introduced to count and minimize the misclassification error. The soft margin SVM classification problem can be formulated as follows:

Minimize $\{w, \xi\} \frac{1}{2} ||w||^2 + C \sum_{i=1}^{\ell} \xi_i$

subject to

$$y_i \left[ \langle w, x_i \rangle - b \right] \geq 1 - \xi_i \quad i = 1, \ldots, \ell$$

$$\xi_i \geq 0, \quad i = 1, \ldots, \ell$$

(3.28)

The constant $C$ defines the trade-off and an aggregation weight of multi-objective problem of maximization of the margin and minimization of the error of misclassification. It also can be interpreted as an upper bound on the impact of the error of any given point.
Figure 3.4: Linear separating hyperplanes for separable (a) and non-separable (b) cases of non-linear classification of yellow (class '+1') and green (class '-1') points.

The dual form of (3.28) is formulated as follows:

\[
\text{Maximize } L_D(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} \alpha_i \alpha_j y_i y_j \langle \mathbf{x}_i, \mathbf{x}_j \rangle \\
\text{subject to } 0 \leq \alpha_i \leq C, \quad i = 1, \ldots, \ell
\]  

(3.29)

### 3.2.1.3 Non-linear Soft Margin Classification SVM

In case the training points cannot be successfully linearly classified in the original space \( X \in \mathbb{R}^n \) (Figure 3.4-a), they can be mapped using a mapping \( \Phi \) to some feature (possibly infinite dimensional) Euclidean space (to be an image \( \Phi(X) \)), where the classification error hopefully will be smaller (Figure 3.4-b), because the Vapnik-Chervonenkis (VC) dimension is higher [Vapnik and Chervonenkis, 1971].

**Definition 12. (The Vapnik-Chervonenkis dimension).** The Vapnik-Chervonenkis dimension, \( VC(H) \), of hypothesis space \( H \), defined over instance space \( X \) is the size of the largest finite subset of \( X \) shattered (isolated) by \( H \). If arbitrary large finite sets of \( X \) can be shattered by \( H \), then \( VC(H) \equiv \infty \).

VC-dimension defines the maximum number of training points which can be classified exactly for all possible labelings. For a linear classifier defined in \( n \)-dimensional space \( VC(H) = n + 1 \).

VC dimension gives an estimate of the upper bound on ”expected risk“ \( R(\alpha) \) (a measure of the error on unseen test data) of an \( \alpha \)-parametrized learning machine with probability \( 1 - \eta \):

\[
R(\alpha) \leq R_{\text{emp}}(\alpha) + \sqrt{\frac{VC(H)(\log(2\ell/h) + 1) - \log(\eta/4)}{\ell}},
\]

(3.30)

where \( R_{\text{emp}}(\alpha) \) is the ”empirical risk“, i.e., the measured error on the training data, and \( \ell \) is the size of the training set. The latter bound can be interpreted as follows: as the problem dimension increases, VC-dimension also increases that may decrease the
"empirical risk" since it becomes simpler to classify the training data, but also increase the upper bound of the "expected risk" on test data (in order to fix the bound, \(\ell\) should be increased).

The mapping of the training set and the whole algorithm to a higher dimensional space may improve the generalization error, but computing this mapping may become computationally intractable for some \(\Phi\) [Burges, 1998]. Fortunately, the formulation of SVM (3.29) does not require to explicitly know \(\Phi\), but only to know how to compute dot product of points. Hence if there exists a kernel function \(K\) defined on \(X\) such that \(K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle\) for all \((x_i, x_j)\) [Aizerman et al., 1964], the problem (3.29) can be solved in \(X\). One example is the Radial Basis Function (RBF):

\[
K_{RBF}(x_i, x_j) = \exp^{-\|x_i - x_j\|^2/2\sigma^2},
\]

where \(\sigma\) is a bandwidth parameter. RBF kernel maps any input point \(x_i\) onto a hyper-sphere of radius 1 since \(K(x_i, x_i) = 1\).

According to Hilbert-Schmidt theory, \(K(x_i, x_j)\) can be any symmetric function satisfying the following general Mercer’s conditions [Vapnik, 1995]:

**Theorem 1.** (Mercer) To guarantee that the symmetric function \(K(u,v)\) form \(L_2\) has an expansion

\[
K(u, v) = \sum_{k=1}^{\infty} a_k \langle \psi_k(u), \psi_k(v) \rangle
\]

with positive coefficients \(a_k > 0\) (i.e., \(K(u, v)\) describes a dot product in some feature space), it is necessary and sufficient that the condition

\[
\int \int K(u,v)g(u)g(v)du dv > 0
\]

be valid for all \(g \neq 0\) for which

\[
\int g^2(u)du < \infty
\]

To extend the linear classification SVM to the non-linear case (see Figure 3.4-a,b) the "only" thing to do is to replace the dot product of points defined in the original decision space \(X\) (3.29) by \(K(x_i, x_j)\) defined in the feature space \(\Phi(X)\) as follows:

Maximize \(L_D(\alpha) = \sum_{i=1}^{\ell} \alpha_i - \frac{1}{2} \sum_{i=1}^{\ell} \alpha_i \alpha_j y_i y_j K(x_i, x_j)\)

subject to \(0 \leq \alpha_i \leq C, \quad i = 1, \ldots, \ell\)

Thus, the decision function now is non-linear in the decision space:

\[
\hat{f}_{classSVM}(x) = \text{sign} \left( \sum_{i=1}^{\ell} y_i \alpha_i K(x_i, x) - b \right)
\]

An efficient algorithm to find optimal \(\alpha\) parameters of classification SVM can be found in [Platt, 1998].
3.2.2 \( \epsilon \)-Support Vector Regression (\( \epsilon \)-SVR)

Considering a training set \( D = \{(x_i, y_i), i = 1, \ldots, \ell\} \), where \( x_i \in \mathbb{R}^n, y_i \in \mathbb{R} \), \( \epsilon \)-SVR proposed by [Vapnik, 1995] consists in finding a function \( \hat{f}_{\epsilon \text{SVR}} \) that has at most \( \epsilon \) deviation from the actual targets \( y_i \) for all training points, and at the same time is as flat/regular as possible (see Figure 3.5-a). \( \epsilon \)-SVR does not take into account any approximation errors which are less than \( \epsilon \): this is defined by a so called \( \epsilon \)-insensitive loss function \( |\xi|_\epsilon \) [Smola and Schölkopf, 2004], illustrated in Figure 3.5-b and described as

\[
|\xi|_\epsilon = \begin{cases} 
0 & \text{if } |\xi| \leq \epsilon \\
|\xi| - \epsilon & \text{otherwise,}
\end{cases}
\]  

(3.37)

The \( \epsilon \)-SVR problem in primal form is defined as follows:

\[
\text{Minimize}_{\{w, \xi, \rho\}} \frac{1}{2} ||w||^2 + C \sum_{i=1}^{\ell} (\xi_{up}^i + \xi_{low}^i)
\]

subject to

\[
\begin{align*}
\langle w, x_i \rangle + b - y_i &\leq \epsilon + \xi_{up}^i, & i = 1, \ldots, \ell \\
y_i - \langle w, x_i \rangle - b &\leq \epsilon + \xi_{low}^i, & i = 1, \ldots, \ell \\
\xi_{up}^i, \xi_{low}^i &\geq 0, & i = 1, \ldots, \ell,
\end{align*}
\]  

(3.38)

The dual problem in feature space is

\[
\text{Maximize } L_D(\alpha) = \sum_{i=1}^{\ell} y_i (\alpha_{up}^i - \alpha_{low}^i) - \frac{1}{2} \sum_{i=1}^{\ell} \sum_{j=1}^{\ell} (\alpha_{up}^i - \alpha_{low}^i)(\alpha_{up}^j - \alpha_{low}^j)K(x_i, x_j)
\]

subject to

\[
\begin{align*}
0 \leq \alpha_{up}^i, \alpha_{low}^i &\leq C, & i = 1, \ldots, \ell \\
\sum_{i=1}^{\ell} (\alpha_{up}^i - \alpha_{low}^i) & = 0
\end{align*}
\]  

(3.39)

The prediction function of \( f(x) \) is

\[
\hat{f}_{\epsilon \text{SVR}}(x) = \sum_{i=1}^{\ell} (\alpha_{up}^i - \alpha_{low}^i)K(x_i, x) + b
\]  

(3.40)

The optimal \( \alpha \) parameters and corresponding \( b \) can be found as described in [Smola and Schölkopf, 2004]. [Chang and Lin, 2001] proved that \( \epsilon \)-SVR with pa-
3.2 Support Vector Machine (SVM)

3.2.3 One-Class SVM

One-Class SVM was proposed by [Schölkopf et al., 2001] for novelty and outliers detection. One-Class SVM learns a function $f_{OneClassSVM}$ that takes the value +1 in ”small” region capturing most of the training data and -1 elsewhere. The strategy is to separate the training data from the origin in the feature space by the hyperplane with maximum margin $\rho/||w||$ (see Figure 3.6), and this is equivalent to finding the hyper-sphere with minimum volume that includes most of the training points [Tax and Duin, 2004]. Given training vector $x_i \in \mathbb{R}^n, i = 1, \ldots, \ell$ without any class information, the primal problem of One-Class SVM is

$$
\text{Minimize}_{(w, \xi, \rho)} \left\{ \frac{1}{2} ||w||^2 - \rho + \frac{1}{\nu\ell} \sum_{i=1}^\ell \xi_i \right\}
$$
\begin{align*}
\text{subject to } & \langle w, x_i \rangle \geq \rho - \xi_i, \quad i = 1, \ldots, \ell \\
& \xi_i \geq 0, \quad i = 1, \ldots, \ell,
\end{align*}

(3.41)

where the parameter $\nu \in [0, 1]$ controls the trade-off between the radius of the hyper-sphere and the number of training points that it can hold, $\nu \in [0, 1]$ is also an upper bound on the fraction of outliers in the data set.

The dual problem in feature space is:

$$
\text{Minimize } L_D(\alpha) = \frac{1}{2} \sum_{i=1}^\ell \sum_{j=1}^\ell \alpha_i \alpha_j y_i y_j K(x_i, x_j)
$$
\begin{align*}
\text{subject to } & 0 \leq \alpha_i \leq \frac{1}{\nu\ell}, \quad i = 1, \ldots, \ell \\
& \sum_{i=1}^\ell \alpha_i = 1
\end{align*}

(3.42)
Figure 3.7: Three classes of points are illustrated: blue (rank 1), green (rank 2) and yellow (rank 3) points. The goal of ranking SVM is to find separated maximum-margin hyper-planes $W(r_1)$ and $W(r_2)$ with the norm $w$ such that all points are correctly ranked w.r.t their score on $w$.

The decision function is:

$$\hat{f}_{OneClassSVM}(x) = \text{sign} \left( \sum_{i=1}^{\ell} y_i \alpha_i K(x_i, x) - \rho \right)$$

(3.43)

The optimal $\alpha$ parameters and corresponding $\rho$ can be found as described in [Schölkopf et al., 2001].

### 3.2.4 Ranking SVM

A new learning setting aimed at preference learning, a.k.a. ordinal regression, has been addressed within the SVM framework [Herbrich et al., 1999, Joachims, 2005]. While preference learning can be cast as a classification problem on $X \times X$ (the class of $(x, x')$ is positive iff $x$ is to be preferred to $x'$), it may offer better generalization to formalize preference learning as an under-constrained regression problem, where the hypothesis $h$ mapping $X$ onto the real-valued space $\mathbb{R}$ is only required to satisfy $h(x) > h(x')$ whenever $x$ is preferred to $x'$. $\epsilon$-SVR also can be used for ordinal regression (see Section 4.1.4).

Let $\mathcal{D} = \{x_1, \ldots, x_\ell\}$ and let $\mathcal{P}$ denote the set of $m$ pairs $(i,j)$ such that $x_i$ is preferred to $x_j$: the original formulation of rank-based SVM, involving all preference constraints, is as follows:

$$\text{Minimize}_{\{w, \xi\}} \frac{1}{2} \|w\|^2 + C \sum_{(i,j) \in \mathcal{P}} \xi_{i,j}$$

subject to

$$\begin{cases}
\langle w, x_i \rangle - \langle w, x_j \rangle - 1 + \xi_{i,j} & \geq 0, \\
\xi_{i,j} \geq 0, & \forall (i,j) \in \mathcal{P}
\end{cases}$$

(3.44)

(3.45)
3.3 Surrogate-assisted Optimization

where $\xi_{i,j}$ stands for the slack variable associated to the violation of the preference constraint associated to $(x_i, x_j)$.

The dual form in feature space is

$$\text{Maximize}_{\alpha} L_D = \sum_{(i,j) \in P} \alpha_{ij} - \frac{1}{2} \sum_{(i,j) \in P} \sum_{(u,v) \in P} \alpha_{ij}\alpha_{uv} (K_{iu} - K_{iv} - K_{ju} + K_{jv})$$

subject to $0 \leq \alpha_{ij} \leq C, \forall (i,j) \in P,$

where $K_{ij} = K(x_i, x_j)$.

The choice of constraints may significantly affect the model quality as well as the model learning computational complexity. Usually two cases are considered:

**Linear in $\ell$ number of constraints**, e.g., with $\ell - 1$ defined preference relations and corresponding constraints, when $x_i$ is said to be preferred to $x_{i+1}$ for $i = 1, \ldots, \ell - 1$. This may be sufficient for model learning, since the final chain will correctly rank all training individuals.

**Super-linear/Quadratic in $\ell$ number of constraints**, e.g., with $1.2\ell$ (respectively, $(\ell - 1)^2/2$) constraints, when $x_i$ is preferred to some fraction of $x_j$ for $i + 1 \leq j \leq \ell$ (respectively, to all $x_j$ for $i + 1 \leq j \leq \ell$). This variant may be more robust if there is some uncertainty in preference relations.

We will discuss in detail both cases and the corresponding optimization procedures just before their application in Section 4.1.1 and Section 5.2.

When optimal $\alpha$ if found, the decision function that predicts the score of a test point $x$ is defined as follows:

$$\hat{f}_{\text{RankSVM}}(x) = \sum_{\forall (i,j) \in P} \alpha_{ij} (K(x_i, x) - K(x_j, x))$$

The evaluated score of $x$ allows to compute its rank with respect to the other evaluated points.

3.3 Surrogate-assisted Optimization

The basic idea of the surrogate-assisted optimization is to use one or several surrogate models $\hat{f}$ of the objective function $f$ to improve the quality of the search (e.g., in terms of number of functions evaluations required to reach the optimum). It is usually assumed that the evaluation of $f$ is expensive in terms of time or money and the learning of $\hat{f}$ is relatively cheap. $\hat{f}$ is learned/built from a set of pairs $(x, f(x))$ where actual $f$ has been computed. $\hat{f}$ should be learned optimally for an algorithm $A$ in the sense that there exists no $\hat{f}'$ such that $\hat{f}'$-assisted $A$ performs statistically better than $\hat{f}$-assisted $A$ (i.e., the optimum of $f$ can be found faster in terms of number of function evaluations). Of course one can suppose that the optimal $\hat{f}$ in the sense described above should be equal to $f$, surprisingly this is not always the case. An illustrative example would be $f(x) = \sum_{i=1}^n x_i^{10}$.
and $A$ is an algorithm which assumes a quadratic shape of $f$. In this case, $\hat{f}'(x) \approx \sum_{i=1}^{n} x_i^2$ might be a better approximation for this particular algorithm $A$ than $f(x) \approx \sum_{i=1}^{n} x_i^{10}$, because this $\hat{f}'$ corresponds to the class of functions the $A$ was designed for, while the optimum of $\hat{f}'$ and $f$ is the same. A similar effect was also investigated in the context of multi-modal optimization [Yew-Soon et al., 2006], where a smoother approximation $\hat{f}'$ sometimes is preferred to a more precise approximation $\hat{f}$ of the rugged $f$, and is able to prevent the search from getting stuck in local optima.

The optimal learning of $\hat{f}$ and optimal interaction between $\hat{f}$ and $A$ conceals many pitfalls and special cases in single- and multi-objective optimization, which we will discuss in this Section. A comprehensive survey of surrogate-assisted Evolutionary Algorithms can be also found in [Jin, 2005, Jin, 2011].

### 3.3.1 General Framework

A simple iteration of the surrogate-assisted algorithm is the following:

1. Learn $\hat{f}$ from training set of previously evaluated points $(x, f(x))$.
2. Use $\hat{f}$ in $A$ (either by replacing $f$ with $\hat{f}$, or by some other mechanism).
3. Compute additional points $(x, f(x))$.
4. Update training set.

In the following we will analyze in detail some steps of the described above procedure.

#### 3.3.1.1 Model Quality Assessment

Without any doubts, the Mean Square Error (MSE) is the most popular model quality indicator in the literature. It is defined as follows:

$$
\text{MSE} = \frac{1}{\ell} \sum_{i=1}^{\ell} w_i (f(x_i) - \hat{f}(x_i))^2,
$$

where $w_i = 1$, $i = 1, \ldots, \ell$ are the weights of the impact of each predicted $\hat{f}(x_i)$ to the final error estimation.

This indicator was investigated in [Jin, 2003], where one important aspect was observed: the Mean Square Error of the model only weakly correlates with the ability to select correct individuals. This means that what we usually do (minimization of the MSE of the model) only weakly correlates with what we really need in comparison-based optimizers (only the correct ranking of individuals). The problem can be partially solved if the measure uses some rank-dependent weighted error $w_i$. However, appropriate weights are usually unknown a priori.

[Jin, 2003] suggested to use the rank correlation coefficient [Spearman, 1904], given by

$$
\rho^{(\text{rank})} = 1 - \frac{6 \sum_{i=1}^{\lambda} d_i^2}{\lambda(\lambda^2 - 1)},
$$

where $\lambda$ is the number of values in the dataset, and $d_i$ is the difference in ranks for each pair of observations.


as a measure for the monotonic relation between the ranks of two variables, where $d_i$ is the difference between the ranks of the $i$-th offspring (among $\lambda$) individual based on the original fitness function and on the surrogate model. $\rho^{(\text{rank})} = 1$ (respectively, $\rho^{(\text{rank})} = -1$) when the model predicts correct (respectively, inverse) ranking of individuals. Again, weight coefficients can be defined to amplify the importance of better ranked individuals.

Rank-based quality indicators are invariant to rank-preserving transformations of the objective function, therefore, look quite promising for comparison-based Evolutionary Algorithms. However, they are rather unknown in the EC community, with a few exceptions [Ulmer et al., 2004, Runarsson, 2006, Kern et al., 2006, Ingimundardottir and Runarsson, 2011]. There is one simple reason for this: in most surrogate-assisted Evolutionary Algorithms there is no control of the model quality and the algorithm $A$ uses the surrogate model $\hat{f}$ without online checking whether the model is good or not. This checking is usually done offline, such that the surrogate-assisted algorithm $A$ does not fail, and this is usually achieved by setting quite conservative surrogate model hyper-parameters. The surrogate model quality indicators will attract much more attention when the comparison of different candidate surrogate models will become a common part of the learning procedure (see Section 4.3.2).

### 3.3.1.2 Training and Test Set Selection

*Training set selection* procedure should answer the question of which training points best represent the original function $f$ to be used to build the optimal model $\hat{f}$. *Test set selection* is responsible for identifying which set of test points is best suitable for the model quality assessment.

**Training Set Selection**

There are three popular ways to select representative points for model learning:

1. **Best Points** Select $\ell$ points with the best fitness among all evaluated points seems to be a reasonable approach for unimodal noise-less optimization problems. If $f$ is a noisy function, then this selection may become biased to noise-affected "lucky" points. If $f$ is a multi-modal or noisy function, then the algorithm also may search far away or at a different scale from the subspace where best points were observed, therefore, $\hat{f}$ will be learned for a subspace which is far away from the actual optimal search subspace of $A$.

2. **k-Nearest Neighbor Points** One can suppose that the $k$-nearest neighbor points ($k = \ell$) to some chosen point $m$ (e.g., mean of the mutation distribution in ESs) best represent the actual search space and are good candidates for model learning. The Euclidean or Mahalanobis distance can be chosen as well as any other suitable similarity measure metric (see also clustering method for local models learning in [Liu et al., 2009]). However, if $f$ is a multi-modal function, it may happen that $m$ is located around a local optimum visited by $A$ many generations ago, and many of $\ell$ closest points to $m$ may correspond to a small cluster of this local optimum, whose very precise learning may be undesirable.
3. Recently Evaluated Points One of the simplest strategy is to choose \( \ell \) the most recently evaluated points, which are very likely to be reasonably good on \( f \) and relatively close to the actual search subspace of \( A \). This approach works reasonably well on noisy problems, but may be insufficiently "greedy", because it "forgets" promising individuals generated more than \( \ell \) evaluations ago.

Optimal training and test set selection strategies depend on the surrogate modelling technique used and the optimization algorithm \( A \) (see Section 4.3.3.2), and probably should combine the described above approaches in some adaptive way.

Resampling Methods

Some approaches to model selection and validation such as resampling methods [Kohavi, 1995, Queipo et al., 2005, Bischl et al., 2012] can be applied almost independently on \( A \).

Split sample validation / Holdout With split sample validation, the whole dataset \( D \) of selected for model learning points is split into training set \( D_{\text{training}} \) and test set \( D_{\text{test}} \). The size of the test set \( \ell_{\text{test}} = |D_{\text{test}}| \) is usually smaller than the size of the training set \( \ell = |D_{\text{training}}| \), but still should be representative enough to estimate the generalization error and keep the confidence interval reasonably small. The test set is not used for model learning, because it would lead to overfitting. This limits the amount of information available for model learning and makes the estimate more biased. Another disadvantage is that the generalization error estimate is very sensitive to a particular split of training and test points.

Cross-validation In \( k \)-fold cross-validation [Stone, 1974] the dataset \( D \) is randomly split into \( k \) mutually exclusive subsets (the folds) \( D_1, D_2, \ldots, D_k \). The cross-validation process is repeated \( k \) times, each time \( i = 1, \ldots, k \), the model is trained and tested using training set \( D_i \) and test set \( D/D_i \), respectively. The cross-validation estimate of accuracy is averaged over \( k \) estimates of each subset. In leave-one-out cross-validation, \( k = \ell \) and the smallest possible test set \( D_i \) of only one test point (\( |D_i| = 1 \)) is considered. This procedure provides a good statistical estimate of accuracy, but may become computationally very expensive if \( D \) is large.

Bootstrap In contract to the cross-validation, where all subsets \( D_i \) are mutually exclusive, in bootstrap [Efron, 1979] the subsets are generated with replacement. Therefore, the probability of any point being chosen after \( \ell \) trials is \((1/\ell)^\ell \approx e^{-1} \approx 63.2\%\) (for large \( \ell \)), thus, only 63.2 % of \( D \) will be presented in the training set \( D_{\text{training}} \), while the remaining points form the test set \( D_{\text{test}} \). The final estimation of the error can be computed as the weighted sum of errors on the test and training sets. However, this still leads to a biased estimation of the generalization error, since some points are presented more than once in \( D_{\text{training}} \). An improvement version \( 0.632 + \) bootstrap was proposed in [Efron and Tibshirani, 1979], where the weights are corrected depending on a quantified relative overfitting rate.

A comprehensible analysis of successes and failures of the bootstrap and other resampling strategies is given in [Horowitz, 2001, Bickel et al., 1997], and specifically for Evolu-
3.3 Surrogate-assisted Optimization

Resampling approaches, while being powerful tools for accurate model quality estimation, still have only minor popularity in the EC community, mainly because of additional computations needed to re-learn the model several times. We envision that resampling approaches will become more relevant for complex multi-modal functions optimization and for surrogate-assisted algorithms, which tend to sample new candidate solution from mixed distributions, and, therefore, will need a careful estimation of the model error in different (disjoint) regions of the search space.

3.3.1.3 Selection of Surrogate Models and Hyper-parameters Adaptation

To build a surrogate model \( \hat{f} \) of \( f \) for the surrogate-assisted optimization algorithm \( A \), a common approach is to choose a surrogate learning approach with suggested \( \theta \) hyper-parameters from the literature. This may become a difficult task in the context of surrogate-assisted optimization, because the ideal \( \hat{f} \) depends on \( A \), the set of possible surrogate techniques to build this \( \hat{f} \) is quite large (see Section 3.1 and Figure 3.1), and, finally, the suggested hyper-parameters are actually suggested for benchmark problems and not for the real-world problem at hand. An obvious solution to this problem is to consider multiple surrogates and get the best estimate of \( f \). The two the most popular ways to aggregate the information from all surrogates in evolutionary and non-evolutionary optimization are the following [Viana and Haftka, 2008]:

1. **Best Surrogate** The idea to choose the best surrogate is obvious and probably was born on the same day when two surrogate models were built with different \( \theta \) hyper-parameters of the same surrogate technique. Since the earliest approaches [Zhang, 1993, Kohavi, 1995] the cross-validation has been intensively used for selecting surrogates. It is also a common practice in EC to build surrogate models using different techniques and then choose the best one for exploitation, for state-of-the-art approaches in this direction see [Viana, 2011].

2. **Weighted Average Surrogate (AWS)** [LeBlanc and Tibshirani, 1993] demonstrated that combining the estimators with some regularization can be useful for improving prediction performance. The weighted average surrogate \( \hat{f}_{AWS}(x) = \sum_{i=1}^{k} \hat{f}_i(x) \) is a weighted aggregation of \( k \) surrogate models \( \hat{f}_i(x) \), where \( \sum_{i=1}^{k} w_i = 1 \). The option 'best surrogate' is also recovered if all weights but one are set to 0, and the option 'average surrogate' if all weights are equal. The averaging may guarantee some robustness, but an adaptation of the weights is definitely a more general and perspective strategy. A recent analysis of adaptive weights selection depending on error correlation, prediction variance and cross validation error for PRS, RBF, Kriging, GP, SVR surrogate models is given in [Acar and Rais-Rohani, 2009].

First papers on selection of surrogates were published many years before Evolutionary Computation even first time touched this subject, in the early 1990s. However, in the 2000s the EC community made a step further and many adaptive and self-adaptive approaches have been proposed. [Zhou et al., 2007] shows that the combination of global and local
surrogate models performs better together than independently. [Shi and Rasheed, 2008] suggested to adaptively adjust the model complexity and the frequency of model usage according to time spent and model accuracy. An illustrative example of advances in surrogate model adaptation is the approach presented in [Gorissen et al., 2009], where SVR, Kriging, ANN and Ensemble of surrogate models evolve together with their $\theta$ hyperparameters in a parallel Genetic Algorithm, and more successful surrogate models are better represented in the population.

### 3.3.1.4 Strategies of Surrogate Model Exploitation

The key aspect of surrogate-assisted optimization is to get and exploit maximum information from the surrogate model $\hat{f}$ of $f$, and this is usually realized in one of the following ways:

1. **Pre-selection of promising solutions** Pre-selection has become especially popular for Evolution Strategies [Emmerich et al., 2002, Ulmer et al., 2004], where $\lambda$ individuals are generated at each iteration. In pre-selection strategy there are two popular options: i) generate $\lambda_{\text{Pre}} > \lambda$ individuals, then pre-select and evaluate only $\lambda' = \lambda$ (usually) best individuals according to $\hat{f}$; ii) generate $\lambda_{\text{Pre}} = \lambda$ individuals, pre-select and evaluate only $\lambda' < \lambda$ and use $\hat{f}$ estimate for ranking the other individuals. In both cases if $\hat{f}$ well approximates $f$, the overall number of function evaluations can be reduced thanks to a quicker (for the first case described above) or comparable (the second case with smaller than $\lambda$ evaluations per generation) progress per generation toward the optimum of $f$.

2. **Surrogate-informed operators** All operators of Evolutionary Algorithms in principle may benefit from knowing more information about $f$ or at least its good approximation $\hat{f}$. Informed operators [Rasheed and Hirsh, 2000] such as crossover [Anderson and Hsu, 1999] or mutation [Abboud and Schoenauer, 2002] provide the capability of quickly moving generated individuals toward regions with improved fitness.

3. **Direct optimization of the surrogate model** If $\hat{f}$ is a reasonably good approximation of $f$, then it can be used in lieu of $f$ for optimization [Jin et al., 2002]. A crucial point is to control the optimal time/number of generations that $\hat{f}$ should be used to obtain the maximum speedup and prevent the algorithm from divergence when $\hat{f}$ is imprecise.

The described above approaches are not that different from each other: the pre-selection when $\lambda_{\text{Pre}} > \lambda$ can be viewed as a Monte-Carlo search with the budget of $\lambda_{\text{Pre}}$ function evaluations, and the pre-selection is actually an informed operator. They are also similar in the sense that in both cases a model quality based control should be used to maximize the speedup and prevent divergence. In this thesis, we will use both the pre-selection in Sections 4.2, 5.3 and direct optimization in Section 4.3.
3.3 Surrogate-assisted Optimization

It should be noted that some surrogate models, such as Kriging, provide not only an estimation of the fitness of a tested point, but also the level of uncertainty of this estimation. There is a sub-field of optimization (see, e.g., Efficient Global Optimization (EGO) algorithm in [Jones et al., 1998]), which focuses on the expected improvement maximization $E[\max(f(\mathbf{x}) - Y, 0)]$, where $Y$ is a random variable that models the uncertainty of the estimation of $f(\mathbf{x})$.

### 3.3.2 Single-objective Case

In this Section, we will focus on state-of-the-art surrogate-assisted Evolution Strategies.

#### 3.3.2.1 Surrogate-assisted Evolution Strategies

Probably the first surrogate-assisted ($\mu + \lambda$)-ES and ($\mu, \lambda$)-ES were proposed in [Papadrakakis et al., 1998], where evaluations of an expensive structural optimization problem were replaced by a hidden-layer ANN trained by back-propagation. The authors suggested to re-learn the model at each iteration adding new training points randomly drawn from a Gaussian distribution with the mean located in the center of the decision space. They also reported that a speedup of a factor of 10 can nevertheless be obtained.

[Jin et al., 2001b] proposed individual-based and generation-based evolution control strategies for surrogate-assisted search with CMA-ES. Individual-based strategy corresponds to the pre-selection strategy, where random (random strategy) or best (best strategy) controlled individuals may be evaluated on the expensive function. In generation-based strategy the whole population will be evaluated on the expensive function for $\eta$ generations every $\kappa$ generations, where $\eta \leq \kappa$. The authors found that in both cases about 50\% of individuals should be evaluated in order to have a good final speedup, but employed only the generation-based strategy, because they found it more suitable for parallel implementation. It is suggested to set the fraction of generations $\frac{\eta}{\kappa}$ that $f$ is optimized to be proportional to $\frac{E(k)}{E_{\max}}$, where $E(k)$ and $E_{\max}$ are current and maximum model errors, respectively.\footnote{We found that we actually reinvented this simple idea in our surrogate-assisted CMA-ES, described in Section 4.3.1.3.}

The cost function of the ANN-based surrogate is defined by

$$
E = \frac{1}{\ell} \sum_{i=1}^{\ell} p(x_i) (f(x_i) - \hat{f}_{\text{ANN}}(x_i))^2,
$$

(3.50)

where $p(x_i)$ is the weight for a sample $x_i$, calculated using the covariance matrix $\mathbf{C}$ of CMA-ES:

$$
p(x_i) = \exp\left(-\frac{1}{2}(x_i - q)^T \mathbf{C}^{-1} (x_i - q)\right),
$$

(3.51)

where $q$ is an individual in the current population to be refereed. To reduce the complexity of training set selection, all individuals with a weight smaller than a given threshold are discarded in learning.
The experimental validation on 20-dimensional Ackley and Rosenbrock functions as well as on a real-world problem with computational fluid dynamics (CFD) simulations showed that the generation-based surrogate-assisted CMA-ES outperforms the original CMA-ES and that the weighted MSE estimation (3.50) is preferred to the original MSE estimation (3.48).

[Emmerich et al., 2002] studied Kriging model based pre-selection strategy within \((\mu, \kappa, \lambda)\)-ES [Schwefel, 1993] with \(\mu = 15\), \(\lambda = 100\) and \(\kappa = 5\), where the individuals that exceed the age of \(\kappa = 5\) generations are eliminated from the selection procedure. The authors suggested to evaluate the best \(\lambda' = 10\) individuals w.r.t. a criterion based on both the estimated value \(\hat{f}(\mathbf{x})\) and the estimated local standard deviation \(\hat{\sigma}(\mathbf{x})\) of the prediction RBF model:

\[
S_{c}(\mathbf{x}) = \hat{f}(\mathbf{x}) - \alpha \hat{\sigma}(\mathbf{x}),
\]

where \(\alpha\) defines the selection trade-off between the most promising solutions with \(\alpha = 0\) and promising solutions in still unexplored search areas with \(\alpha > 0\). The experimental validation of the proposed surrogate-assisted algorithm showed that on unimodal functions the exploration (\(\alpha = 1\)) does not harm, but leads to a better convergence on multi-modal functions than \(\hat{f}\)-based selection (\(\alpha = 0\)).

[Ulmer et al., 2003b] studied RBF networks based pre-selection strategy for \((2,8)\)-ES with \(\lambda_{Pre} = 30\) and \((1+1)\)-ES with \(\lambda_{Pre} = 10\) and concluded that the proposed meta-model assisted ES (MAES) performs better, especially on unimodal functions. Later [Ulmer et al., 2003a] analyzed GP based pre-selection strategy similar to the one of [Emmerich et al., 2002] for CMA-ES, where expected improvement was chosen as a selection criterion. The results confirmed the observations of [Emmerich et al., 2002] that on multi-modal functions the expected improvement-based pre-selection should be preferred to "greedy" \(\hat{f}\)-based pre-selection. [Ulmer et al., 2004] further studied SVR-based pre-selection strategy for \((\mu, \lambda)\) Main Vector Adaptation (MVA [Poland and Zell, 2001], a CMA-ES variant with linear time and space complexity). They proposed to adjust \(\lambda_{Pre}\), depending on a model quality measure similar to the one proposed by [Jin, 2003] and defined as summed rank of all correctly selected individuals. At each generation \(t\) the actual model quality \(Q^t\) is compared to a quality \(Q^{rand}\) measured for the random model. The update procedure is the following:

\[
\lambda_{Pre}^{t+1} = \begin{cases} 
\lambda_{Pre}^t + \frac{Q^{max} - Q^t}{Q^{rand} - Q^t} \delta_{Q^{Pre}} & \text{if } Q^t > Q^{rand} \\
\lambda_{Pre}^t - \frac{Q^{rand} - Q^t}{Q^{max} - Q^t} \delta_{Q^{Pre}} & \text{otherwise}
\end{cases}
\]

where \(Q^{max}\) is the maximum possible quality. Experimental results of the proposed C-MAES on unimodal and multi-modal functions confirmed that the adaptation works well even in dynamically changing noisy environment, where the speedup up to a factor of 2 also can be achieved.

[Runarsson, 2004] found that it is always desirable to evaluate on \(f\) at least one best point among \(\lambda\) evaluated on \(\hat{f}\), and proposed approximate ranking procedure which suggests to evaluate on \(f\) several additional points only if the model changes its ranking prediction of \(\lambda\) points (see the next Section for details).
3.3 Surrogate-assisted Optimization

[Poland, 2004] proposed an ES-like algorithm with quadratic meta-models, which showed the best results in the literature for an Evolutionary Algorithm on popular Rosenbrock benchmark problem, where it outperformed CMA-ES by a factor of 4. Unfortunately, the algorithm is not invariant to rank-preserving transformation and its performance probably will degrade significantly if \( f \) is scaled differently (also true for all regression to value based surrogate-assisted optimizers).

[Buche et al., 2005] proposed Gaussian Process Optimization Procedure (GPOP) which suggests to repeat the following procedure. First, build a GP based model and directly optimize Eq. (3.52) by CMA-ES. When a local optimum for a given \( \alpha \) is found, it can be evaluated on \( f \) and added to the training set. The training set consists of the union of the \( N_C \) closest points to the current best solution and the \( N_R \) most recently evaluated points. The search space for each local search is constrained by the hyper-rectangle around the current best solution. The GPOP outperforms CMA-ES on Sphere and Schwefel functions, but already for Rosenbrock the speedup is less than 2.0 for \( n = 8, 16 \) and less than 0.4 for \( n = 32 \). On multi-modal Rastrigin function GPOP is always outperformed by CMA-ES.

[Runarsson, 2006] first proposed to use ordinal regression and Ranking SVM as surrogate models in Evolutionary Computation. The author studied the performance of Ranking SVM based CMA-ES within the approximate ranking procedure and found that the surrogate-assisted version outperforms the original CMA-ES for \( n \leq 5 \), but shows no improvements for \( n > 5 \). The latter can be explained by the use of very small training set of only 60 training points. However, this paper has played an important role in the development of comparison-based surrogates that we will propose in Chapter 4.

[Hoffmann and Holemann, 2006] suggested to keep \( \lambda_{Pre} \) fixed, but adapt \( \lambda \) using the following formula, similar to (3.53):

\[
\lambda^{t+1} = \begin{cases} 
\max(\lambda^t - \frac{Q_{max}^t - Q_{rand}^t}{Q_{max}^t - Q_{rand}^t} \delta_{\lambda}, \mu) & \text{if } Q^t > Q_{rand}^t \\
\min(\lambda^t + \frac{Q_{max}^t - Q_{rand}^t}{Q_{max}^t - Q_{rand}^t} \delta_{\lambda}, \lambda_{Pre}) & \text{otherwise}
\end{cases}
\]

Thus, the better the model - the smaller \( \lambda^{t+1} \), and vice versa. The results of the proposed \( \lambda \)-controlled surrogate-assisted \( \lambda \)-CMA-ES are similar, but not directly comparable to [Ulmer et al., 2004], where MVA was used as a baseline algorithm. A comparison in [Hebbel and Nisticò, 2008] confirmed that \( \lambda \)-CMA-ES and C-MAES have comparable performance on a robot’s walking optimization problem. For a detailed analysis of different model quality indicators used to adapt \( \lambda \) the interested reader is refereed to [Gräning et al., 2007].

3.3.2.2 The Local Meta-model CMA-ES (lmm-CMA-ES)

The local meta-model CMA-ES was first proposed by [Kern et al., 2006] and later extended for large populations by [Bouzarkouna et al., 2010] and for partially-separable functions by [Bouzarkouna et al., 2011]. It is one of the most carefully designed surrogate-assisted CMA-ES algorithms.

Locally Weighted Regression In lmm-CMA-ES the locally weighted regression (LWR, [Atkeson et al., 1997]) surrogate model \( \hat{f} \) is build around each generated offspring
individual $q \in \mathbb{R}^n$ using a set of training points $(x_i, y_i), i = 1, \ldots, \ell$. The model error $A$ is minimized w.r.t. the parameters $\beta$ of the local model ($\hat{f} = \hat{x}^T \cdot \beta$) at query point $q$ and can be written as

$$A(q) = \ell \sum_{i=1}^{\ell} \left[ (\hat{f}(x_i, \beta) - y_i)^2 K \left( \frac{d(x_i, q)}{h} \right) \right], \quad (3.55)$$

where $K(\cdot)$ is the kernel function and $d(x_i, q)$ is the distance between data points $x_i$ and $q$, and $h$ is the local bandwidth. [Kern et al., 2006] investigated surrogates with different model complexity and reported that the best performance on test problems is achieved with the full quadratic surrogate models defined as

$$\hat{f}(x, \beta) = \beta^T (x_1^2, \ldots, x_n^2, x_1 x_2, \ldots, x_{n-1} x_n, x_1, \ldots, x_n, 1)^T, \quad (3.56)$$

where $\beta \in \mathbb{R}^{\frac{n(n+3)}{2}+1}$. The weights matrix $W$ of Eq. (3.2) was set by [Kern et al., 2006] to diag($\sqrt{K(d(x_i, q)/h)}$) $\in \mathbb{R}^{\ell \times \ell}$. For the calculation of $d(x_j, q)$ [Kern et al., 2006] proposed to use the covariance matrix $C$ adapted by CMA-ES to take into account correlations between the variables and compute the Mahalanobis distance:

$$d(x_j, q) = \sqrt{(x_j - q)^T C^{-1}(x_j - q)} \quad (3.57)$$

The kernel $K$ is chosen to be bi-quadratic:

$$K(\zeta) = \begin{cases} (1 - \zeta)^2 & \text{if } \zeta < 1, \\ 0 & \text{otherwise} \end{cases} \quad (3.58)$$

The bandwidth $h$ is defined by the distance to the $k$-th nearest neighbor point to $q$, where $k = n(n+3) + 2$ was chosen after some experimental analysis on Sphere, Ellipsoid and Rosenbrock functions.

**Approximate Ranking Procedure** In lmm-CMA-ES the surrogate model exploitation is controlled by the approximate ranking procedure [Runarsson, 2004], which suggests to evaluate with $f$ only a fraction of $\lambda$ new individuals if the ranking of some of $\lambda$ individuals on surrogate models of the current and previous generations remains unchanged. In lmm-CMA-ES the original fitness evaluation procedure of CMA-ES is replaced by the following procedure, where initially all $n_b = \lambda$ individuals are evaluated on $f$, then the number of evaluated individuals per iteration is increased (starting with $n_{init}$) by $n_b = \max \left(1, \left\lceil \frac{\lambda}{10} \right\rceil \right)$ as surrogate models show poor quality of prediction:

1. **build** $\lambda$ models $\hat{f}_i(x)$ for all $i = 1, \ldots, \lambda$ individuals of the current population $(x_i, f(x_i))_{i \leq \lambda}$.
2. **rank** individuals according to their approximated value $\hat{f}(x)$: $x_i, \lambda_{nc=0}$.
3. **evaluate** the best $n_{init}$ individuals with the true objective function $f$ and **add** their evaluations to the training set.
4. for \( n_{ic} = 1, \ldots, \left\lfloor \frac{\lambda - n_{init}}{n_b} \right\rfloor \) do

(a) build \( \lambda \) models \( \hat{f}_i(x) \) for all \( i = 1, \ldots, \lambda \) individuals of the current population \((x_i, f(x_i))_{i \leq \lambda}\).

(b) rank individuals according to their approximated value \( \hat{f}_i(x) \): \( x_i \).

(c-1) if \((ug_{\lambda n_{ic}} = x_{i:}\lambda_{n_{ic}}}, for \( i = 1, \ldots, \lambda \) \)// ranking of the \( \lambda \) best is the same then the meta-model is accepted, GO TO STEP 5.

(c-2) if \( (n_{init} + n_{ic}n_b < \frac{\lambda}{4}) \) and \((ug_{\lambda n_{ic}} = x_{i:}\lambda_{n_{ic}}}, for \( i = 1, \ldots, \mu \) \)// less than one-fourth of the population is evaluated and ranking of the \( \mu \) best is the same or \( (n_{init} + n_{ic}n_b \geq \frac{\lambda}{4}) \) and \((ug_{\lambda n_{ic}} = x_{i:}\lambda_{n_{ic}}}, for \( i = 1, \ldots, \mu \) \)// at least one-fourth of the population is evaluated and the best individual is the same then the meta-model is accepted, GO TO STEP 5.

(d) evaluate the \( n_b \) best unevaluated individuals with the true objective function \( f \) and add their evaluations to the training set.

5. if \( (n_{ic} > 2) \) then \( n_{init} = \min(n_{init} + n_b, \lambda - n_b) \).
else if \( (n_{ic} < 2) \) then \( n_{init} = \max(n_b, n_{init} - n_b) \).

In the original lmm-CMA-ES [Kern et al., 2006], the model is accepted if the surrogate-based ranking of all \( \lambda \) individuals remains unchanged after adding new \( n_b \) points to the training set, this model acceptance criterion corresponds to line 4.(c-1) of the procedure described above. A less conservative model acceptance criterion described in line 4.(c-2) was proposed in [Bouzarkouna et al., 2010], the corresponding algorithm is called nlmm-CMA-ES. When \( \lambda \) is large it becomes quite difficult to avoid small perturbations of ranking and keep it exactly the same after adding new batch of points. For this case if more than one-fourth of the population is already evaluated the new criterion suggests to accept the model if the ranking of the best individual does not change.

Performance of lmm-CMA-ES and nlmm-CMA-ES Performance of lmm-CMA-ES and different versions of nlmm-CMA-ES was tested on Sphere, Schwefel, Rosenbrock and Rastrigin functions in dimensions 2, 4, 8 and 16 [Kern et al., 2006, Bouzarkouna et al., 2011]. Both algorithms show a speedup between a factor of 1.1 and 6.7 (usually between 2 and 4) in comparison with the original CMA-ES. In contrast to the original lmm-CMA-ES, nlmm-CMA-ES is able to better keep this speedup for larger populations than the default \( \lambda_{de\text{fault}} = 4 + 3 \left\lfloor \log(n) \right\rfloor \). An important drawback of the algorithm was discussed already in [Kern et al., 2006] and further studied in [Kern et al., 2007]: the time complexity of the original algorithm scales with \( O(n^6) \), limiting the range of application to problems with \( n \lesssim 15 \). Several alternative approaches were proposed in [Kern et al., 2007] to reduce the time complexity to \( O(n^4) \), however, the observed cost in terms of number of floating points operations per saved function evaluation still scales with \( O(n^5) \) [Kern et al., 2007].

Besides from the time complexity of the algorithm (relevant for both versions), lmm-CMA-ES has two other important drawbacks: i). the evaluation of individuals cannot be
efficiently parallelized; ii). the algorithm loses the invariance of the original CMA-ES w.r.t. rank-preserving transformation of $f$. Suppose that there are $\lambda_{\text{default}}$ CPUs and the evaluation of $\lambda_{\text{default}}$ individuals on the expensive $f$ runs in parallel on each CPU independently, if the evaluations are not heterogeneous (see Yagoubi and Schoenauer, 2012 for an analysis of heterogeneous case), then there are might be a speedup of the original CMA-ES by a factor of $\lambda$. In lmm-CMA-ES all individuals are evaluated in the batch of maximum size $n_b = \lfloor \lambda/10 \rfloor$, i.e., the evaluation of $\lambda$ individuals will take $\frac{\lambda}{n_b} = 10$ times more time. Given that the speedup of lmm-CMA-ES is usually smaller than 10 in this scenario (in number of function evaluations), we will actually lose the time with comparison to the original CMA-ES. Therefore, the use of the nlmm-CMA-ES in the scenario of parallel evaluations does not make much sense for both default and large population sizes. The drawback of the lmm-CMA-ES is that it uses the metric regression (regression to value) in its quadratic meta-model, therefore scaling of $f$ will change the meta-model and the quality of the approximation. This effect is observed for the Schwefel function $f_{\text{Schwefel}}(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2$ and its scaled variant $f_{\text{Schwefel}}^{1/4} = (f_{\text{Schwefel}}(x))^{1/4}$, where the speedups for nlmm-CMA-ES in dimension 4 is 5.4 and 2.9, respectively [Bouzarkouna et al., 2010]. We suppose that the loss of invariance of CMA-ES w.r.t. rank-preserving transformation of $f$ may lead to no speedup at all of lmm-CMA-ES on the same test problem under certain rank-preserving transformations of $f$. It also should be noted that all original problems in [Kern et al., 2006, Bouzarkouna et al., 2010] have local or global quadratic landscape, and this is desirable property for quadratic meta-models.

### 3.3.3 Multi-objective Case

Surrogate-assisted multi-objective evolutionary optimization is a relatively new field, which was born in the early 2000s and the majority of the papers were published during the last three-four years.

One of the first attempts to extend the surrogate-assisted evolutionary search to the multi-objective case was made by [Yang et al., 2002], who proposed to build Kriging-based surrogates $\hat{f}_i$ for each objective $f_i$ and enrich the training set using few points (usually only two or three) from a set of non-dominated points found after multi-objective optimization (and niching) of surrogates $\hat{f}_i$. [Farina, 2002] proposed a very similar approach with ANN-based surrogate models (see also ANN-NSGA-II [Nain and Deb, 2003]), where in order to increase the diversity of the training set, a point from the most unexplored area of the search space is evaluated (to find such point is an additional single-objective optimization problem). The concept of the direct optimization of surrogates $\hat{f}_i$ is quite simple, but seems to be very reasonable if $f_i$ can be approximated efficiently and problem dimension is relatively small.

Later [Voutchkov and Keane, 2006] studied the same approach and compared different types of Kriging, RBF and Polynomial models. The authors reported very interesting results: on 10-dimensional ZDT2 and ZDT3 functions the proposed surrogate-assisted algorithms using about 100 function evaluations find a Pareto front approximation comparable to the one found by NSGA-II after about 10000 evaluations (our indirect estimate). This is probably one of the most important results in the field one should keep in mind,
3.4 Discussion

because it shows "the real" complexity of these ZDT functions.

Instead of learning $m$ different $f_i$ surrogate models, another approach is to learn and exploit only one global modal $\hat{f}_g$, which will aggregate the information about $f$. In ParEGO [Knowles, 2006], an extension of EGO algorithm for Pareto optimization, at each iteration the algorithm builds a randomly-weighted aggregated model $\hat{f}_g$. Then, the best solution found after optimization of $\hat{f}_g$ for a fixed number of generations is estimated and added to the training set. In contrast to the direct optimization of $\hat{f}_i$ surrogates that results to a set of non-dominated solutions, in ParEGO only one solution is obtained at each iteration, and, therefore, we should be able to properly formulate a priori our preferences (see the discussion of preference-based and ideal principles in Section 2.4.2).

[Emmerich and Naujoks, 2004] proposed CR-NSGA-II (NSGA-II with multiple Kriging-based surrogate models), where at each iteration only $\lambda'$ the most promising points are evaluated on $f_i$ (e.g., not dominated by $\mu$ parents) out of $\lambda$ offspring evaluated on $f_i$. [Emmerich et al., 2006a] formalized the notion of expected improvement in multi-objective optimization in terms of hypervolume indicator, such that pre-selection is seeking to maximize contributing hypervolume minimizing Eq. (3.52), where $\alpha$ controls the trade-off between exploration and exploitation. [Ponweiser et al., 2008] followed the idea of [Emmerich et al., 2006a] to calculate expected hypervolume contribution of a potential solution and proposed the hypervolume selection-based EGO (SMS-EGO). SMS-EGO outperforms Par-EGO [Ponweiser et al., 2008] and is comparable to MOEA/D-EGO [Zhang et al., 2010], which extends MOEA/D to multi-objective optimization and is based on a simple pre-selection with some niching in the decision space.

To summarize, both approaches of building individual $\hat{f}_i$ and global $\hat{f}_g$ surrogates are popular, there is, however, a trend to build individual surrogates for each objective and optimize or pre-select solutions w.r.t. expected hypervolume contribution improvement. However, since this approach usually assumes sequential evaluation of solutions, it might be less promising in the scenario with parallel evaluations of solutions.

### 3.4 Discussion

Most of the surrogate modeling methods surveyed in this Chapter build a surrogate model $\hat{f}$ of the actual expensive function $f$ which minimizes the mean square error of the prediction of exact values $f(x)$ of training points. However, there are some disadvantages of the use of MSE as a loss function for surrogate model learning and assessment:

1. MSE is only weakly correlated with the ability to correctly select best/prospective individuals [Jin, 2003], while the correct selection/ranking of individuals is the only thing that is needed from $f$ in comparison-based evolutionary optimizers.

2. MSE is not invariant w.r.t. rank-preserving transformations of $f$, and hence so are not all MSE-based surrogate-assisted optimization algorithms.

3. MSE is not interpretable for a black-box $f$ and can be used heavily as a quality indicator in surrogate model control procedures.

75
The disadvantages described above are also relevant for all quality indicators which use exact values of \( f \) and not only the information about comparison relations between the points, which is actually sufficient to build surrogate model. In this Chapter, we presented the Ranking SVM approach which learns the ranking of training points and predicts the score of test points. In contrast to PR, ANN, RBF, GP, Ranking SVM is invariant to rank-preserving transformations of \( f \), that we believe (and will demonstrate) is the key source of robustness.

Another important aspect of surrogate modeling is the computational complexity of model learning and testing. GP, RBF and PR models scale at least with \( O(\ell^3) \) and/or \( O(n^3) \), and this limits their application to low-dimensional problems. ANNs are cheaper to compute, but the model quality heavily depends on the ANN’s structure and the learning itself is a multi-modal problem. In contrast to ANNs, the learning of Ranking SVM model is a convex quadratic programming problem, whose optimum is unique and can be found by any quadratic programming solver. We will also show in Section 4.1 that the learning of Ranking SVM is much cheaper and typically scales with \( O(\ell^2) \).

The choice of "appropriate" hyper-parameters (e.g., number of training points) for surrogate learning procedure is the key problem of efficient surrogate-assisted optimization. It would be desirable to have a way to automatically detect optimal hyper-parameters which are, indeed, algorithm- and problem-dependent. In Section 4.3, we will show how to automatically tune/optimize surrogate model hyper-parameters during the optimization of the expensive function, thus, significantly reducing the problem of optimal model learning and hyper-parameters selection.
Part III

Contributions
Chapter 4

Single-Objective Surrogate-Assisted CMA-ES

From the practical point of view, it is so urgently desirable to obtain the smallest probable error with a given number of subjects, that the method of rank must often have the preference even when we are dealing with two series of measurements properly comparable with one another.


In this Chapter, we demonstrate the advantages of the use of ordinal regression over metric regression in surrogate-assisted optimization, and CMA-ES in particular. In Section 4.1, we analyze in detail the learning procedure of Ranking SVM, discuss the question of the choice of Kernel function and demonstrate comparative results of Ranking SVM and SVR. In Section 4.2, we present the surrogate-assisted rank-based CMA-ES (ACM-ES) algorithm, published in [Loshchilov et al., 2010c], which is based on pre-selection of promising offspring on \( \hat{f} \). In Section 4.3, we discuss the advantages and drawbacks of the original ACM-ES and propose the self-adaptive surrogate-assisted CMA-ES (**ACM-ES), published in [Loshchilov et al., 2012e, Loshchilov et al., 2012b, Loshchilov et al., 2012c]. The algorithm relies on iterative optimization of \( f \) for one generation and \( \hat{f} \) for \( \hat{n} \) generations, where \( \hat{n} \) depends on the prediction quality of \( \hat{f} \). We also show that hyper-parameters \( \alpha \) used to build \( \hat{f} \) can be adapted during the optimization, such that the user is only required to define the range of these parameters. The algorithm is experimentally validated on BBOB benchmark problems, where it shows best results for ill-conditioned problems, outperforming such algorithms as IPOP-aCMA-ES, BFGS and NEWUOA. In Section 4.4, we conclude the Chapter with a discussion and some perspectives for further research.

4.1 Ordinal Regression

The problem of predicting variables of ordinal scale is called ordinal regression or learning to rank, even though the latter is a broader term (see, e.g., [Li, 2011]). Ordinal regression can be viewed as a special case of preference learning, where the statement \( x \) is preferred to \( y \) can be simply expressed as an inequality \( f(x) > f(y) \) [Chu and Ghahramani, 2005b]. The goal of preference learning is to learn the underlying ordering \( \hat{f} \) over the instances from these pairwise preference relations. Preference learning may deal with contradicting preferences, while in ordinal regression the transitivity is
usually assumed. We will analyze preference learning in a context of surrogate-assisted multi-objective optimization in Section 5.2. In this Section, we will investigate ordinal regression within Ranking SVM, proposed by [Herbrich et al., 1999] and popularized by [Joachims, 2005, Chu and Keerthi, 2005].

4.1.1 Learning to Rank using Ranking SVM

One can rewrite the primal form of Ranking SVM Eq. (3.44) in a more general form with individual constraint violation weights $C_i$ and desirable difference $\epsilon_i$ on $\hat{f}$ for each pair of preference constraints as follows:

Minimize $\{w, \xi\} \frac{1}{2} ||w||^2 + \sum_{i=1}^{\lvert P \rvert} C_i \xi_i$ \hspace{1cm} (4.1)

subject to $\begin{cases} \langle w, x_a \rangle - \langle w, x_b \rangle \geq \epsilon_i - \xi_i, & i = 1, \ldots, \lvert P \rvert; (a, b) \leftarrow P_i \\ \xi_i \geq 0, & i = 1, \ldots, \lvert P \rvert, \end{cases}$ \hspace{1cm} (4.2)

where $P$ denote the set of pairs $(a, b)$ such that $x_a$ is preferred to $x_b$.

Then, the dual form of Ranking SVM will have the following form:

Maximize $\{\alpha\} L_D = \sum_{i=1}^{\lvert P \rvert} \alpha_i \epsilon_i - \frac{1}{2} \sum_{i=1}^{\lvert P \rvert} \sum_{j=1}^{\lvert P \rvert} \alpha_i \alpha_j dK_{ij}$ \hspace{1cm} (4.3)

subject to $0 \leq \alpha_i \leq C_i, i = 1, \ldots, \lvert P \rvert$, where $dK_{ij} = K(x_a, x_b) - K(x_a, x_d) - K(x_c, x_b) + K(x_c, x_d)$ for two pairs of constraints $P_i = (a, b)$ and $P_j = (c, d)$.

One can solve Eq. (4.3) using Sequential Minimal Optimization (SMO) method proposed in [Platt, 1998] for classification and later extended for Regression SVM [Shevade et al., 2000]. SMO suggests to sequentially optimize Eq. (4.3) by tuning minimum number of $\alpha$ parameters at a time (usually one or two). Then, Eq. (4.3) can be rewritten as a maximization w.r.t. some selected Lagrangian multiplier $\alpha_x$ as follows:

Maximize $\{\alpha_x\} L_D = \alpha_x \epsilon_x - \alpha_x \sum_{i \neq x} \alpha_i dK_{xi} - \frac{1}{2} \alpha_x^2 dK_{xx} + L_{D_{\text{const}}}$ \hspace{1cm} (4.4)

subject to $0 \leq \alpha_x \leq C_x$,

where $L_{D_{\text{const}}}$ is the part of Lagrangian which is independent on $\alpha_x$. If change $\alpha_x$ to $\alpha_x^* = \alpha_x + \Delta \alpha_x^*$, then Eq. (4.4) can be rewritten as follows:

Maximize $\{\Delta \alpha_x^*\} \Delta L_D = \Delta \alpha_x^* \epsilon_x - \Delta \alpha_x^* \sum_{i=1}^{\lvert P \rvert} \alpha_i dK_{xi} - \frac{1}{2} \Delta \alpha_x^* \Delta \alpha_x^* \sigma^2 dK_{xx}$ \hspace{1cm} (4.5)

subject to $0 \leq \alpha_x + \Delta \alpha_x^* \leq C_x$,

At the stationary point of the objective function (4.5) we have

$$\Delta \alpha_x^* = \frac{\epsilon_x - \sum_{i=1}^{\lvert P \rvert} \alpha_i dK_{xi}}{dK_{xx}},$$ \hspace{1cm} (4.6)

where $\epsilon_x$ will be set to 1 for all experiments in this Chapter.
For the sake of completeness and to demonstrate the simplicity of the learning to rank approach, we present an efficient implementation of SMO for Ranking SVM in Algorithm 4.1. To reduce the computational complexity we pre-compute $dK$ in lines 2-7, a temporary vector $\Delta \alpha^*_i, i = 1, \ldots, |P|$ and matrix $DivdK_{ij}, i, j = 1, \ldots, |P|$ in lines 12 and 14, respectively. The initial values of $\alpha$ parameters are set to constraint violation weights $C$ in line 9, any other non-negative initial $\alpha$ would lead to the optimum, but probably with a smaller convergence rate. For $n_iter$ iterations (lines 17-28) we select a Lagrangian multiplier $\alpha_{ix}$ (line 18) and update it (line 28) using Eq. (4.6) to maximize the Lagrangian $\Delta L$ (line 23). New value $\alpha_{new}$ should be checked to satisfy $0 \leq \alpha_{new} \leq C_{ix}$ (line 21), then for any non-zero $\Delta \alpha^*_i$, always have $\Delta L > 0$. The computational complexity of the algorithm scales with $O(n_{iter}n_{alpha})$, where $n_{iter}$ should be chosen in order of 1000$n_{alpha}$ (see Section 4.1.2) and $n_{alpha}$ will be set to $\ell - 1$ for all experiments in this Chapter. The computation of Kernel matrix $K \in \mathbb{R}^{\ell \times \ell}$ scales with $O(n\ell^2)$, i.e., $O(n^3)$ if $\ell = n$, but it usually has 10-100 times smaller constant factor than the one of the SMO optimization procedure. The memory complexity scales with $O(\ell^2)$, limiting the use of non-linear Kernels to $\ell \leq 1000$ on ordinary computers.

The simplest strategy to identify $\alpha_{ix}$ with the largest potential increase of $\Delta L$ is to compute $\Delta L$ for all $\alpha_{ix}$. This allows to decrease the number of updates of $\Delta \alpha^*$ (lines 26), but the procedure itself is linear in $\ell$, so there is a substantial gain in time only on first iterations of the algorithm before the distribution of improvements of Lagrangian become almost uniform for all $\alpha_{ix}$. We tried to apply gradient-based methods to optimize the dual form whose gradient information is available, but found no gains in terms of time. Finally, we use the simplest baseline procedure and iteratively choose all $\alpha_{ix}$ (line 18). We also found that random permutation of indexes of $\alpha$ does not lead to a faster convergence.

In a machine learning context, Quadratic Programming (QP) problems such as (4.3) are usually optimized up to a given accuracy $\epsilon_{QP}$. In this thesis, we adopt a different stopping criterion based on number of iterations $n_{iter}$ because in the context of surrogate-assisted optimization we need to control the computational time spent on surrogate model learning, as surrogate-assisted optimization involves many steps of surrogate learning. As the number $n_{iter}$ of iterations needed to reach precision $\epsilon_{QP}$ varies depending on the difficulty of the underlying QP, which itself varies along the search, we opted for specifying the number $n_{iter}$ of iterations (directly visible in the computational cost) rather than $\epsilon_{QP}$ (since we cannot bound a priori $n_{iter}$ needed to reach an accuracy $\epsilon_{QP}$). Since our way of defining the stopping criterion is not common in Machine Learning, we present an analysis of $n_{iter}$-based stopping criterion and its comparison with $\epsilon_{QP}$-based stopping criterion in Section 4.3.3.10.

1 In an early version of the manuscript, the formula in line 23 was incorrectly given as $\Delta L \leftarrow \Delta \alpha dK_{ix} (\Delta \alpha^*_i - 0.50, 0.58, 0\Delta \alpha + 1)$. The term ”+ 1” was introduced by a mistake both in the algorithm description and the source code, that led to a loss of performance of the learning algorithm in order of 1% in terms of time and number of iterations of SMO (while, the factor is problem-dependent), since the algorithm did not pass ”$\Delta L > 0$” when necessary and several iterations among $n_{iter}$ were lost. Thus, only about 0.99$n_{iter}$ were used instead of the declared 1.00$n_{iter}$. No significant difference of model quality is observed between the original and the corrected versions. However, the results presented in this Chapter are given for the original version except for Section 4.3.3.10.
Algorithm 4.1: Rank SVM Dual problem optimization

1: given training points \( \mathbf{x}_i, i = 1, \ldots, \ell \); Kernel matrix \( K \in \mathbb{R}^{\ell \times \ell} \); pool of constraints \( \mathcal{P} \); 
   \( n_{\text{alpha}} = |\mathcal{P}| \); \( C \in \mathbb{R}^{n_{\text{alpha}}} \); \( n_{\text{iter}} \)
2: for \( i = 1 \ldots n_{\text{alpha}} \) do
3:   for \( j = 1 \ldots n_{\text{alpha}} \) do
4:     \((a, b) \leftarrow \mathcal{P}_i; (c, d) \leftarrow \mathcal{P}_j\)
5:     \( dK_{ij} = K(x_a, x_b) - K(x_a, x_d) - K(x_c, x_b) + K(x_c, x_d) \)
6:   end for
7:   end for
8: for \( i = 1 \ldots n_{\text{alpha}} \) do
9:   \( \alpha_i \leftarrow C_i \)
10: end for
11: for \( i = 1 \ldots n_{\text{alpha}} \) do
12:   \( \Delta \alpha_i^* \leftarrow \frac{1 - \sum_{j=1}^{n_{\text{alpha}}} \alpha_j dK_{ij}}{dK_{ii}} \)
13:   for \( j = 1 \ldots n_{\text{alpha}} \) do
14:     \( DivdK_{ij} \leftarrow \frac{dK_{ij}}{dK_{jj}} \)
15:   end for
16: end for
17: for \( i = 1 \ldots n_{\text{iter}} \) do
18:   \( i_x \leftarrow i \ mod \ (n_{\text{alpha}} + 1) \)
19:   \( \alpha_{\text{old}} \leftarrow \alpha_{i_x} \)
20:   \( \alpha_{\text{new}} \leftarrow \alpha_{\text{old}} + \Delta \alpha_{i_x}^* \)
21:   \( \alpha_{\text{new}} \leftarrow \max(\min(\alpha_{\text{new}}, C_{i_x}), 0) \)
22:   \( \Delta \alpha \leftarrow \alpha_{\text{new}} - \alpha_{\text{old}} \)
23:   \( \Delta L \leftarrow \Delta \alpha dK_{i_xi_x}(\Delta \alpha_{i_x}^* - 0.5\Delta \alpha) \)
24:   if \( \Delta L > 0 \) then
25:     for \( j = 1 \ldots n_{\text{alpha}} \) do
26:       \( \Delta \alpha_j^* \leftarrow \Delta \alpha_j^* - \Delta \alpha DivdK_{i_xj} \)
27:     end for
28:   end if
29: end for
30: end for
31: output: optimal \( \alpha \)

4.1.2 Choice of Kernel Function

Performance of SVM algorithms may be significantly affected by a choice of kernel function. The most commonly used kernels are RBF kernel Eq. (3.31), Linear kernel defined as

\[
K_{\text{Linear}}(\mathbf{x}_i, \mathbf{x}_j) = \mathbf{x}_i \mathbf{x}_j, \quad (4.7)
\]
4.1 Ordinal Regression

Figure 4.1: Examples of non-linear SVM classification using (Left) Polynomial kernel of degree 2 [Ben-Hur et al., 2008] and (Right) RBF kernel.

and inhomogeneous Polynomial kernel defined as

\[ K_{\text{inPoly}}(x_i, x_j) = (x_i x_j + c)^d, \]  

(4.8)

where \( c > 0 \) and \( d \) is polynomial degree (for difference of inhomogeneous and homogeneous \((c = 0)\) kernels see [Scholkopf et al., 1999]).

Polynomial kernels allow to solve non-linear classification (see Figure 4.1-Left) and regression problems with a decision boundary, whose flexibility is growing with \( d \). The flexibility of RBF kernels allows to solve even so non-linear problems as the one shown in Figure 4.1-Right.

To choose an "appropriate" kernel for Ranking SVM we perform experiments on Sphere \( f_{\text{Sphere}}(x) = \sum_{i=1}^{n} x_i^2 \) and Rosenbrock \( f_{\text{Rosen}}(x) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2) \) functions. To build a Ranking SVM based model \( \hat{f} \) we first sort \( \ell \) training points in ascending order w.r.t. chosen \( f \) and fill a pool of \( \ell - 1 \) constraints \( \mathcal{P} \) such that each point \( x_i \) is preferred to points \( x_{i+1} \) for \( i = 1, \ldots, \ell - 1 \). For RBF kernel we set to \( \sigma = c_{\text{sigma}} \sigma_x \), where \( \sigma_x \) is the average distance between training points. The trade-off constant \( C_i \) was set to \( 10^{C_{\text{base}}} \) for \( i = 1, \ldots, \ell - 1 \). When the model is build after SMO optimization as described in the previous Section, we measure model quality on \( \ell_{\text{test}} = |\Lambda| \) points ordered after \( f \) as follows:

\[ \text{Err}(\hat{f}) = \frac{2}{|\Lambda|(|\Lambda| - 1)} \sum_{i=1}^{|\Lambda|-1} \sum_{j=i+1}^{|\Lambda|} \mathbb{I}_{\hat{f},i,j} \]  

(4.9)

where \( \mathbb{I}_{\hat{f},i,j} \) holds true iff \( \hat{f} \) violates the ordering constraint on pair \((i,j)\).

In order to investigate the performance of Ranking SVM on different learning problems and for different kernel functions, we set \( c_{\text{sigma}} \) and \( C_{\text{base}} \) to the values which lead to the best results of \( \text{Err}(\hat{f}) \) observed on test points for different settings of \( c_{\text{sigma}} \) and \( C_{\text{base}} \). For each pair of values \( c_{\text{sigma}} \) and \( C_{\text{base}} \), the training set was filled using 100 (respectively, 250) training points randomly uniformly drawn from \([-2, 2]^n\) and evaluated on 10-dimensional
Figure 4.2: Average surrogate model error in \((C_{\text{base}}, c_{\text{sigma}})\)-space (respectively, \((C_{\text{base}}, c)\)-space) measured in 10 runs of Ranking SVM with RBF kernel (respectively, Polynomial kernel with degree 2) on (Left) Sphere and (Right) Rosenbrock functions in 10-D after 1000\(\ell\) iterations of the SMO algorithm. The observed best pair of hyper-parameters is denoted as '*'.

Sphere (respectively, Rosenbrock) function. The test set was filled using \(\ell_{\text{test}} = 1000\) test points from the same region. The 31 values of \(C_{\text{base}}\) (respectively, \(c_{\text{sigma}}\)) are uniformly taken from \([-10, 30]\) (respectively, \([-5, 5]\)). The number of iterations of SMO \(n_{\text{iter}}\) was set to 1000\(\ell\) (see Section 4.3.3.10 for a discussion).

Figure 4.2 shows the \((C_{\text{base}}, c_{\text{sigma}})\)-space of Ranking SVM hyper-parameters for 10-dimensional Sphere and Rosenbrock functions. Indeed, the optimal hyper-parameters (with lowest \(Err(\hat{f})\) values) are different for these problems. A basin (a region where \(Err(\hat{f})\) is already small and tends to decrease slower) in \((C_{\text{base}}, c_{\text{sigma}})\)-space of a relatively good performance is observed for both problems. As a simple estimator of optimal \(C_{\text{base}}^*\) and \(c_{\text{sigma}}^*\), we originally tried to compute the mean/median of 1% (i.e., 9 out of
31×31=961) the best observed pairs of \( C_{\text{base}} \) and \( c_{\text{sigma}} \). While this kind of averaging might be reasonable given that the estimations of \( \text{Err}(\hat{f}) \) are noisy, it also might be very harmful when \( \text{Err}(\hat{f}) \) in \((C_{\text{base}}, c_{\text{sigma}})\)-space is multi-modal: the averaging of points from two good clusters may produce a point which is worse than the points of both clusters. The estimation of the global optimum of noisy multi-modal functions itself is an open scientific problem. Therefore, for the sake of simplicity, in this Section, the estimate of optimal hyper-parameters is set to the best pair of hyper-parameters \( C_{\text{base}}^*, c_{\text{sigma}}^* \), observed in 31×31=961 experiments. The estimated \( C_{\text{base}}^* = 6 \) and \( c_{\text{sigma}}^* = 0.4 \) (respectively, \( C_{\text{base}}^* = 26 \) and \( c_{\text{sigma}}^* = -0.2667 \)) for 10-dimensional Sphere function with 100 training points (respectively, Rosenbrock function with 250 training points).

For Polynomial kernel-based Ranking SVM a similar offline tuning procedure is applied, where instead of \( c_{\text{sigma}} \) the kernel parameter \( c \) (see Eq. 4.8) is varied in the range \([10^{-5}, 10^5]\). The estimated optimal values of \( C_{\text{base}}^* \) and \( c^* \) for Polynomial kernel-based Ranking SVM (see Figure 4.2) are used to plot the results shown in Figure 4.3 (see also Appendix A for several demonstrations of the \((C_{\text{base}}, c)\)-space). The kernel matrix is additionally normalized [Graf and Borer, 2001] such that \( \bar{K}_{ij} = K_{ij}/\sqrt{K_{ii}K_{jj}} \) (note that RBF-based kernel matrix is normalized by definition since \( K_{ii} = 1 \)).

Figure 4.3 shows the average model error \( \text{Err}(\hat{f}) \) estimated on 10-dimensional Sphere and Rosenbrock functions for 20 runs of Ranking SVM with different kernels: RBF ('RankSVM+RBF'), Linear ('RankSVM+Linear'), Polynomial with \( d = 2 \) ('RankSVM+Poly2'), Polynomial with \( d = 3 \) ('RankSVM+Poly3'), etc. For RBF kernel-based (respectively, Polynomial kernel-based) Ranking SVM the estimated \( C_{\text{base}}^* \) and \( c_{\text{sigma}}^* \) (respectively, \( C_{\text{base}}^* \) and \( c^* \)) values are used. It should be noted that these values are estimated for a given SMO stopping criterion of \( n_{\text{iter}} = k_{\text{iter}} 1000\ell \) iterations individually for each \( k_{\text{iter}} = 0.01, 0.1, 1, 10 \), and Figure 4.3 investigates a dependency of \( \text{Err}(\hat{f}) \) on \( k_{\text{iter}} \) for different kernel functions.
Chapter 4. Single-Objective Surrogate-Assisted CMA-ES

Ranking SVM with Linear kernel has $Err(\hat{f})$ slightly below 0.5, where $Err(\hat{f})$ can be interpreted as the fraction of incorrectly predicted comparison relations. A random 50% prediction ($Err(\hat{f}) = 0.5$) is due to the fact that the functions are not linear. Quadratic Sphere function has probably the most suitable topography for Polynomial kernel with degree 2, so its good results are not surprising. Despite implicitly defined high degree of non-linearity, Polynomial kernels of higher degree also catch the topography of Sphere function, but likely due to a relatively successful offline parameter tuning (see Figure 1 in Appendix A). On Rosenbrock function, on the contrary, the degree 2 is insufficient and higher degree of non-linearity ($d > 2$) is needed to catch the "banana" topography of this function. Ranking SVM with RBF kernel and Polynomial kernel with degree 10 and 30 show good results on both functions. While the computational complexity of model learning is growing linearly with $k_{\text{iter}}$, the decrease of model error $Err(\hat{f})$ usually becomes slower for $k_{\text{iter}} > 1$ on these particular problems. In some cases, $Err(\hat{f})$ even increases with $k_{\text{iter}}$ (see, e.g., 'RankSVM+Poly3' on 10-dimensional Sphere function). This might be due to one or a combination of the following effects:

- A larger number of iterations of SMO optimization leads to a better solution of the dual Ranking SVM problem, but the optimal solution of the dual problem not necessary leads to a smaller test error, while this is a goal of parametrization of the dual - to choose such parameters which will lead to a minimum test error.

- The estimated optimal hyper-parameters are located close to "bad" hyper-parameters in $(C_{\text{base}}, \sigma_{\text{sigma}})$-space, and due to the stochasticity of the process, this might lead to a certain increase of $Err(\hat{f})$.

Indeed, the described above effects are very likely to be also relevant for online tuning of model hyper-parameters.

4.1.3 Ordinal Regression vs Metric Regression

Most surrogate approaches described in Section 3.1 are based on standard regression also referred to as metric regression. To analyze how sensitive metric regression to scaling of $f$ we conducted several experiments with Support Vector Regression (see Section 3.2.2) and Ranking SVM.

The implementation of Support Vector Regression is derived from the Shark library [Igel et al., 2008], where the gradient-based working set selection of QP sub-problem is used in contrast to the uniform selection used in this thesis (see Section 4.1.1). The gradient-based selection usually leads to a faster QP problem optimization per iteration of the learning algorithm, while each iteration is computationally more expensive. In order to calibrate $n_{\text{iter}}$ iterations used to learn the model, we perfomed several experiments on 10-dimensional Sphere function for $\ell = 64, 100, 128, 256, 512, 1024, 2048$ and for 961 pairs of model hyper-parameters. We found that $n_{\text{iter}} = 1000\ell$ of Ranking SVM (including the cost of kernel matrix computation) corresponds to roughly $n_{\text{iter}} = 70\ell$ of SVR. Given that our Ranking SVM code was optimized to be computationally efficient, and in order to reduce a potential bias to Ranking SVM implementation, in the following we use $n_{\text{iter}} = 100\ell$.
for SVR as an equivalent of $n_{iter} = 1000\ell$ of Ranking SVM. For all experiments in this Section we set $\epsilon$ of SVR to 0, but other values of $\epsilon$ lead to very similar results of these noise-less functions.

Figure 4.4 shows the results of Ranking SVM with RBF ('RankSVM+RBF') and Polynomial with $d = 10$ ('RankSVM+Poly10) kernels, Support Vector Regression with RBF and Polynomial kernels on the original function $f$ ('SVR+RBF' and 'SVR+Poly10') and transformed functions $g = f^2$ ('SVR+RBF on $f^2(x)$' and 'SVR+Poly10 on $f^2(x)$') and $g = f^4$ ('SVR+RBF on $f^4(x)$' and 'SVR+Poly10 on $f^4(x)$'), where $f$ is Sphere and Rosenbrock functions for Figure 4.4-Left and Figure 4.4-Right, respectively. Note that Sphere and Rosenbrock functions are positive everywhere.

The results for Ranking SVM are the same for the original and transformed versions, because the transformations of $f$ do not change the ranking of training points. Our first observation is that 'SVR+Poly10' and 'SVR+RBF' demonstrate best performance on the original Sphere function $f$. However, the results of SVR on scaled functions $f^2$ and $f^4$ clearly show that the model accuracy quickly degrades as the transformed function less resembles the original quadratic function. For $f^6$ and $f^8$ SVR-based surrogate model tends to demonstrate even worse (comparable to random) prediction of test test points. This suggests that on one of the simplest optimization problems (e.g., a scaled version of Sphere function) metric regression may provide arbitrary bad results. On Rosenbrock function 'SVR+RBF' and 'SVR+Poly10' perform as well as 'RankSVM+RBF', but degrade quickly on $f^2$ and $f^4$.

The reasons of these results become clearer by looking at the space of model hyperparameters for $f$, $f^2$ and $f^4$ (see Figure 2 in Appendix A). The bassin of hyper-parameters which corresponds to good performance on $\text{Err}(\hat{f})$ becomes much smaller on $f^2$ and $f^4$ (for these particular problems it becomes more difficult to find ”good” mapping between the
Figure 4.5: Average surrogate model error measured in 20 runs of Ranking SVM and Support Vector Regression (SVR) with different rank-based transformations of fitness values on (Left) Sphere and (Right) Rosenbrock functions in 10-D after \(k_{iter}1000\ell\) and \(k_{iter}100\ell\) iterations of the SMO algorithm for Ranking SVM and SVR, respectively.

decision and the objective space), and this is especially the case for the hyper-parameter \(C\) and \(C_{base} = \log(C)\). At a first look, it seems reasonably to scale \(C\) according to the scaling of \(f\). However, in the black-box scenario we observe only \(f\), so it is unclear how to scale \(C\) since the notion of "scaling" can be defined only with respect to another function. As can been seen in Figure 2 of Appendix A, the basin of optimal model hyper-parameters is almost the same for \(f^2\) and \(f^4\) of Sphere function, but moves significantly for Rosenbrock function. Thus, even if the optimal basin is known for \(f\) for a given range of \(C\), it is still unclear how to scale the search range of \(C\) (to have a comparable chance to localize the optimal basin during the offline tuning) for \(f^2\) and \(f^4\), since this scaling is problem-dependent.

In black-box optimization, the scaling of \(f\) is unknown and may vary locally during the search. A sophisticated procedure can be proposed to learn "optimal" scaling of \(f\) and adapt hyper-parameters of SVM accordingly. However, the results on Rosenbrock function suggest that presumably even using the original "preferred scenario" of scaling (which might be preferred to, e.g., \(f^{10}\)), the accuracy of SVR may be at best comparable to the one of Ranking SVM.

### 4.1.4 Ordinal Regression using Metric Regression Approaches

One can use metric regression to learn ranking of training points and predict the score of test points. The original "raw" objective function value of each training point should be modified such that it will depend only on the rank of this point.

The simplest rank-based fitness assignment scheme, referred to as \'(1:N)\', sets the fitness of each training point to be equal to its rank among \(N = \ell\) training points. Since \(\ell\) is usually growing with \(n\), it might be reasonable to normalize new fitness values by \(\ell\) to
4.1 Ordinal Regression

Figure 4.6: Average surrogate model error measured in 20 runs of Ranking SVM and Support Vector Regression (SVR) with different number of training points on (Left) Sphere and (Right) Rosenbrock functions in 10-D after 1000 $\ell$ and 100 $\ell$ iterations of the SMO algorithm for Ranking SVM and SVR, respectively.

the range $[1/N, 1]$: we call this scheme $'(1:N)/N'$. As was shown in the previous Section, quadratic shapes of fitness function may be beneficial for SVR. To check this hypothesis for rank-based SVR, we set the fitness of each training points to be equal to its squared and normalized squared rank value in $'(1:N)^2/2$' and $'(1:N)^2/(N^2)^2$' schemes, respectively.

It can be seen from Figure 4.5 and Figure 4.4 that rank-based SVR performs worse than metric-based SVR in its "preferred scenario" of scaling of $f$, but does not fail like metric-based SVR in its "worst scenario", e.g., on $f^4$, $f^6$, etc., because rank-based SVR is invariant to rank-preserving transformations of $f$. Rank-based SVR performs worse on 10-dimensional Sphere function, while on Rosenbrock function the difference is not significant. The normalization of rank-based fitness given to SVR sometimes even worsen the results, but usually has minor impact. The scaling of the rank-based fitness seems to be more important than the normalization, but quadratic scaling worsen the results on Sphere function and is not better on Rosenbrock function. We also tried to use other scaling schemes such as square root of rank $'(1:N)^{0.5}$', which slightly improves the results on Sphere, but worsen on Rosenbrock. Finally, the simplest $'(1:N)/N'$ scheme probably should be taken as a baseline for a black-box function $f$, but its optimal parametrization in principle may be adapted during the optimization of $f$.

4.1.5 Ranking SVM vs Rank-based SVR

It was shown in the previous Sections that ordinal regression usually should be preferred to metric-regression, since the latter is sensitive to rank-preserving transformations of $f$. One should keep in mind that all conjectures and conclusions drawn from experimental results are supposed to be valid for a given parametrization of algorithms and on a given set of test functions. However, the results obtained on some functions, such as Sphere
function, can be generalized for a larger class of problems which resemble Sphere-like function at least around local and global optima, i.e., on later stages of the optimization of these problems we actually optimize Sphere-like function.

Figure 4.6 demonstrates the results of Ranking SVM and rank-based SVR for different number of training points $\ell = 2, 4, \ldots, 2048$ used to build $\hat{f}$ on (Left) Sphere and (Right) Rosenbrock functions in dimensions 2, 10 and 40 (see also Figure 3 and 4 of Appendix A). It can be seen that Ranking SVM outperforms SVR on Sphere function for $\ell > 10n$: this is very important result which among other reasons has determined our preferences to Ranking SVM. While on Rosenbrock function Ranking SVM outperforms SVR only for $n = 2$, for $n = 10, 40$ both algorithms have very similar performance.

The main difference between Ranking SVM and Rank-based SVR is that the latter is more constrained since target values of $\hat{f}$ are pre-defined for all training points, while in Ranking SVM only the minimum difference of target values is defined. Thus, Ranking SVM may find solutions of SVR constrained to have specific $\hat{f}$ values, but it is also open to a larger set of optimal solutions, which are unavailable for SVR. An important remark is that we analyzed the cheapest variant of Ranking SVM with only $\ell - 1$ constraints, a slightly larger number of constraints, say $1.2\ell$, may improve the robustness of the model learning on multi-modal and/or noisy functions.

The simple offline tuning procedure used in this Section allows to investigate the surrogate models hyper-parameters and to estimate their nearly-optimal values w.r.t. model error. However, the search range of these hyper-parameters still should be defined by the user, the estimation itself is computationally expensive and requires a quite large test set to be available, that is usually not the case (instead, a cross-validation procedure might be considered). Moreover, the adapted grid search procedure badly scales w.r.t. the number of hyper-parameters. In preliminary experiments of Section 4.2.2, for the sake of simplicity we use some fixed values of hyper-parameters found after some offline tuning. This is also the case for multi-objective surrogate-assisted search (see Chapter 5). However, an important contribution of this thesis is a procedure of surrogate model hyper-parameters adaptation by CMA-ES (see Section 4.3.2), which replaces the grid search and automates the resolution of the problem of model selection for the learning problem at hand.

### 4.2 ACM with Pre-selection

To the best of our knowledge, the only work investigating the use of Ranking SVM as surrogate model within a meta-model assisted EA published before [Loshchilov et al., 2010c] was [Runarsson, 2006]; while the approach proposed in [Runarsson, 2006] was reported to bring small improvements over a CMA-ES baseline, a major issue regards the choice of the kernel, the Achilles heel of all SVM-based methods. Following the path opened by [Runarsson, 2006], in [Loshchilov et al., 2010c] we investigated the use of Ranking SVM surrogate models within CMA-ES.

In this Section, we describe Ranking SVM-based CMA-ES, referred to as ACM-ES [Loshchilov et al., 2010c] for (alphabetically) ranked CMA-ES. In Section 4.2.1, we discuss the advantages of the use of the Covariance Matrix adapted by CMA-ES, viewed as the
4.2 ACM with Pre-selection

proportional metric to look at the region of the fitness landscape currently explored, and which renders surrogate learning techniques invariant w.r.t. orthogonal transformations of $f$. In Section 4.2.2, we present pre-selection strategy used in ACM-ES and the algorithm itself. The experimental validation of the approach is reported in Section 4.2.3, and directions for further work are discussed in Section 4.2.4.

4.2.1 Toward Optimal Kernel Function: Invariance w.r.t. Orthogonal Transformations

In Section 3.3.2.2, we have described lmm-CMA-ES algorithm proposed by [Kern et al., 2006], which predicts fitness of an offspring $q$ by building a weighted quadratic meta-model $\hat{f}$ around $q$. An important contribution made by [Kern et al., 2006] was to use the covariance matrix adapted by CMA-ES to compute the weight of each training point $x_j$, such that the weight decreases as the Mahalanobis distance $\sqrt{(x_j - q)^T C^{-1} (x_j - q)}$ to the query point $q$ increases. This covariance matrix-based Mahalanobis distance takes into account the correlations of the data (the Euclidean distance can be obtained by setting the covariance matrix to identity). Later, in order to decrease the computational complexity of lmm-CMA-ES [Kern et al., 2007] studied different versions of lmm-CMA-ES. They pointed out more explicitly that

$$\sqrt{(x_j - q)^T C^{-1} (x_j - q)} = \sqrt{x_j^T x_j},$$

where $x_j'$ is an image of $x_j$ in some transformed space, defined as

$$x_j' = C^{-1/2} (x_j - q) \quad (4.11)$$

Later [Hansen, 2008] proposed Adaptive Encoding procedure which gradually learns an appropriate coordinate system and can be used in principle within any continuous optimizer. The author showed that CMA-ES can be viewed as Cumulative Step-size Adaptation algorithm performed in the transformed coordinate system, adapted by Adaptive Encoding procedure. The change of coordinates, defined from the current covariance matrix $C$ and the current mean value $m$ (this was also proposed before in [Kern et al., 2007]), reads:

$$x_j' = C^{-1/2} (x_j - m) \quad (4.12)$$

We hence propose to define the kernel function (e.g., RBF kernel) directly to the covariance matrix (the idea was proposed in [Loshchilov et al., 2010c] independently from a similar idea for Gaussian Processes proposed in [Kruisselbrink et al., 2010]), with $\sigma > 0$:

$$K_C(x_i, x_j) = e^{-\frac{(x_i - x_j)^T C^{-1} (x_i - x_j)}{2\sigma^2}} \quad (4.13)$$

However, the computation of the kernel matrix with $\ell \times \ell$ elements may become computationally expensive if we need to compute Eq. (4.13) for each element. Instead, we can...
translate all training points to the feature space using Eq. (4.12), where all variables are
as decorrelated as possible and the objective function more resembles the Sphere function,
which is relatively easy to learn (see Figure 4.6). Then, in this feature space, we may
use the Euclidean distance and compute \( \sigma \), which we set to the average distance between
training points.

Figure 4.7 displays surrogate models of 2-dimensional rotated Ellipsoid function \( f \) built
on 15 the same training points using (Left) standard RBF kernel and (Right) transformed
RBF kernel with Eq. (4.13), where \( \mathbf{C} \) is set to the inverse of the Hessian of \( f \). It can
be seen that the use of the transformed RBF allows to obtain more satisfactory (bold)
contour lines of the surrogate model to the (dotted) contour lines of the original Ellipsoid
function, i.e., we may obtain a much better prediction of \( f \) if covariance matrix \( \mathbf{C} \) is known.
Fortunately, CMA-ES adapts \( \mathbf{C} \) during the search, therefore, we may transform training
and test points using Eq. (4.12) to make our surrogate modeling approach invariant w.r.t.
orthogonal transformation of the search space. An important detail is that \( \mathbf{C}^{-1/2} \) is already
computed within CMA-ES, therefore the transformation comes at almost no additional
cost.

We believe that the use of the transformation (4.12) may improve the performance of
most of the surrogate techniques proposed for CMA-ES. For other EAs which do not adapt
\( \mathbf{C} \), the estimation of covariance matrix can be obtained performing Principal Component
Analysis (PCA) of the training data. One should mention that the transformed kernel
(4.13) shares certain similarities with the Fisher Kernel, relatively recently proposed by
[Jaakkola et al., 1999] and briefly reviewed in [Sewell, 2011].
4.2 ACM with Pre-selection

4.2.2 Overview of ACM-ES

Having chosen the transformation of training and test points after Eq. (4.12), the integration of Ranking SVM as surrogate model within CMA-ES raises three main issues: i). how to train the surrogate model, i.e., how to select the current training points in the set of all points evaluated with the true objective function; ii). how to use the model within CMA-ES, without perturbing the delicate adaptive mechanism thereof, and iii). how to select the new points which will be evaluated with the true objective function.

Regarding the first issue, i.e., the selection of the training sample, several requirements have been identified. Firstly, the number \( \ell \) of training samples must increase with the dimension \( n \) of the search space. Using statistical learning arguments, \( \ell \) should be of the order of the VC dimension of the model space. Note that after transformation (4.12) the decision space is a variant of the Sphere function, in the best case, or a noisy multi-modal variant thereof in the worst case. A second requirement is that the training samples should not lie "too far" from the current mean \( m \) of the mutation distribution used by CMA-ES to generate its offspring, since the transformation defined by the current covariance matrix only aims at the local structure of the fitness landscape around \( m \). Finally, the analysis of preliminary experiments on the \( n \)-dimensional Sphere function shows that \( \ell \) should increase proportionally to \( \sqrt{n} \); the proportionality constant however remains problem-dependent as will be seen in Section 4.2.3. These \( \ell \) selected points are the most recently evaluated points with the true fitness function (see Section 3.3.1.2 for a detailed discussion of different selection schemes). Figure 4.8 illustrates the whole optimization loop of ACM-ES, the training points selection corresponds to part A. The surrogate model learning using transformed after Eq. (4.12) training set corresponds to part B.

The second issue regards how to use the rank-based surrogate within CMA-ES. Using the surrogate model \textit{in lieu} of the true fitness is a risky option due to the lack of guarantees about errors in regions outside the training sample (i.e., without model error-based control). A more conservative option thus is to use the surrogate model to pre-screen the offspring, generating many more \textit{pre-children} than required, and keeping the best ones according the surrogate model. Such an approach however rapidly looses the offspring diversity, hindering the CMA-ES adaptive mechanism used to adapt the covariance matrix. Some trade-off between the optimization of the objective and the adaptation of the covariance matrix must thus be found.

The approach finally proposed is a two-step process. In order to prevent premature convergence, and interfere as little as possible with CMA-ES cumulative step-size adaptation, a large number \( \lambda_{pre} \) of pre-children is drawn using the standard CMA Gaussian distribution; let them be noted \( x_1, \ldots, x_{\lambda_{pre}} \), assuming with no loss of generality them to be ranked after the surrogate model (see Figure 4.8 part C). The \( \lambda \) offspring are obtained by iteratively drawing a real number \( a < \lambda_{pre} \) from distribution \( \mathcal{N}(0, \sigma_{sel0}^2) \) (where \( \sigma_{sel0} \) is a parameter of the algorithm), and retaining the pre-child with rank \( \lfloor a \rfloor \) (see Figure 4.8 part D). The same procedure is followed to select the points to be evaluated according to the true objective function, with the same rationale: on the one hand, one should select the best points according to the current surrogate model; on the other hand, some diversity must be preserved. Finally, i). the point with top rank is selected and always
Chapter 4. Single-Objective Surrogate-Assisted CMA-ES

4. Generate = 500 pre-children and rank them according to surrogate fitness function.

A. Select training points
1. Select \( l = k \sqrt{n} \) recently evaluated points.

B. Build a surrogate model
2. The change of coordinates defined from the current covariance matrix \( C \) and the current mean value \( m \):
\[
x'_j = C^{-1/2}(x_j - m)
\]

D. Select the most promising children
5. Prescreen \((\lambda, O)\)
   \[
   \text{Retain } \text{O with rank } \sigma < \lambda_{\text{Pre}}, \sigma \sim \lambda_{\text{Pre}} N(0, \sigma^2_{\text{sel0}})
   \]
6. Evaluate \((\lambda')\)
   \[
   \text{Retain } \text{with rank } \sigma < \lambda, \sigma \sim \lambda N(0, \sigma^2_{\text{sel1}})
   \]

C. Generate pre-children
4. Generate \( \lambda_{\text{Pre}} = 500 \) pre-children and rank them according to surrogate fitness function.

Figure 4.8: Optimization loop of the ACM-ES algorithm with pre-selection.

evaluated (as in the approximate ranking approach [Runarsson, 2004]); ii). other \((\lambda' - 1)\) points selected among the pre-children using a rank distribution \( N(0, \sigma^2_{\text{sel1}}) \) are evaluated, using the same process as for the offspring selection albeit with a larger standard deviation \((\sigma_{\text{sel1}} > \sigma_{\text{sel0}})\). A typical distribution of the ranks of the \( \lambda \) offspring is depicted in Figure 4.8 Part D, where \( \lambda = 12 \) white circles are pre-selected among \( \lambda_{\text{Pre}} = 500 \) pre-children with \( \sigma^2_{\text{sel0}} = 0.4 \), while points that will be evaluated with the true fitness are represented by green circles \((\lambda' = 4 \text{ and } \sigma^2_{\text{sel1}} = 0.8)\).

In ACM-ES, a fixed number \( \lambda' \) of points is evaluated in each generation, thus bounding the complexity in terms of true fitness evaluation. The choice of the ratio \( \lambda/\lambda' \) thus controls the efficiency of the approach and the speedup w.r.t. the standard CMA-ES (where \( \lambda \) offspring are evaluated in each generation).

4.2.3 Experimental Validation

The experimental validation of ACM-ES investigates the performance of the approach comparatively to CMA-ES and nlmm-CMA, focussing on its scalability w.r.t. the problem dimension \( n \), the robustness with respect to multi-modality, and with respect to the calibration of the surrogate training.

Seven uni- and multi-modal benchmark functions have been considered (see Table 4.1,
Table 4.1: Test functions, initialization intervals and initial std. dev. (from [Kern et al., 2006; Bouzarkouna et al., 2010]).

<table>
<thead>
<tr>
<th>Name</th>
<th>Function</th>
<th>Init</th>
<th>sel²</th>
</tr>
</thead>
<tbody>
<tr>
<td>Noisy Sphere</td>
<td>(f_{\text{NoisySphere}}(x) = (\sum_{i=1}^{n} x_i^2) \exp(\epsilon \Lambda(0, 1)))</td>
<td>([-3, 3]^n]</td>
<td>5</td>
</tr>
<tr>
<td>Ellipsoid</td>
<td>(f_{\text{Ellip}}(x) = \sum_{i=1}^{n} 10^i \frac{x_i}{\sqrt{i}})</td>
<td>([1, 5]^n]</td>
<td>2</td>
</tr>
<tr>
<td>Schwefel</td>
<td>(f_{\text{Schwefel}}(x) = \sum_{i=1}^{n} (\sum_{j=1}^{i} x_j)^2)</td>
<td>([-10, 10]^n]</td>
<td>10</td>
</tr>
<tr>
<td>Schwefel(^{1/4})</td>
<td>(f_{\text{Schwefel}^{1/4}}(x) = (f_{\text{Schwefel}}(x))^{1/4})</td>
<td>([-10, 10]^n]</td>
<td>10</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>(f_{\text{Rosenbrock}}(x) = \sum_{i=1}^{n-1} (100(x_i^2 - x_{i+1}^2) + (x_i - 1)^2))</td>
<td>([-5, 5]^n]</td>
<td>0.5</td>
</tr>
<tr>
<td>Ackley</td>
<td>(f_{\text{Ackley}}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2}\right) + \exp\left(\frac{1}{n} \sum_{i=1}^{n} \cos(2\pi x_i)\right))</td>
<td>([1, 30]^n]</td>
<td>14.5</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>(f_{\text{Rastrigin}}(x) = 10d + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i)))</td>
<td>([1, 5]^n]</td>
<td>2</td>
</tr>
</tbody>
</table>

definitions in [Kern et al., 2006] and [Bouzarkouna et al., 2010]), with dimension \(n\) ranging in \([2, 40]\) except for the Rastrigin function. Within ACM-ES, CMA-ES is used with its default parameters [Hansen et al., 2003]. Reported results are based on 20 independent runs. The stopping criterion is reaching target value \(10^{-10}\), with a maximum of \(1000n^2\) function evaluations.

After preliminary experiments, the rank-based surrogate was trained using \(\ell = \lceil 30\sqrt{n} \rceil\) samples for all functions, except for Ellipsoid and Rosenbrock where it was set to \(\lceil 70\sqrt{n} \rceil\) (see Section 4.3.3.2). The maximum number of iterations of the SVM learning algorithm was arbitrarily set to \([50000\sqrt{n}]\). The constraint weights \(C_i\) in Eq. (4.2) were set to \(10^6(\ell - i)^2\), implying that the cost of rank-constraints violation quadratically increases for top-ranked samples. For all functions except Rastrigin, \(\lambda' = \frac{\lambda}{2}, \sigma_{sel0}^2 = 0.4, \sigma_{sel1}^2 = 2\sigma_{sel0}^2 = 0.8, \ell_{test} = 500\). For Rastrigin function \(\sigma_{sel0}^2 = \sigma_{sel1}^2 = 0.6\).

4.2.3.1 Results and Discussion

Firstly, experiments are conducted to estimate the empirical complexity of the surrogate training and using, with \([100\sqrt{n}]\) training points, stopping after \([50000\sqrt{n}]\) iterations and assessing the surrogated model on 500 test points. The empirical complexity with respect to dimension \(n\) (Figure 4.9-Left in log scale) is 1.13, thus, slightly super-linear, contrasting with lmm-CMA complexity of \(O(n^6)\) (see Section 3.3.2.2 for a detailed discussion).

Secondly, the comparative validation of ACM-ES, lmm-CMA and standard CMA-ES on all benchmark functions is reported in Table 4.2; lmm-CMA and nlmm-CMA results have been taken from original papers [Kern et al., 2006] and [Bouzarkouna et al., 2010] when available; those of CMA-ES have been recomputed\(^2\). Overall, ACM-ES outperforms lmm-CMA and nlmm-CMA algorithms on most problems, particularly so for problems with dimension \(n > 4\). The invariance of ACM-ES w.r.t. monotonous transformations of the fitness is witnessed by its almost identical results on \(f_{\text{Schwefel}}\) and \(f_{\text{Schwefel}^{1/4}}\) functions, when the stopping criterion is adjusted accordingly (which is not the case for the results of Table 4.2). Likewise, the results on \(f_{\text{Ellip}}\) confirm that ACM-ES also retains the good behavioral properties of CMA-ES with respect to the ill-conditioning of the fitness.

\(^2\)using the source code available at the website of Nikolaus Hansen: http://www.lri.fr/~hansen/cmaes_inmatlab.html/
Table 4.2: Computational effort SP1 (i.e., the average number of function evaluations of successful runs divided by proportion of successful runs), standard deviations and speedup performance (spu) of ACM-ES, (n)lmm-CMA-ES and CMA-ES. Results in the (n)lmm-CMA column are the best of those in [Kern et al., 2006] and [Bouzarkouna et al., 2010] (marked with leading “n:” for the latter). Successful runs are those who reached the target fitness value of $10^{-10}$. The proportion of successful runs is given in parentheses if less than 100%. $\epsilon$ is the noise level (when relevant).

<table>
<thead>
<tr>
<th>Function</th>
<th>n</th>
<th>$\lambda$</th>
<th>$\lambda$</th>
<th>$\epsilon$</th>
<th>ACM-ES</th>
<th>spu (n)</th>
<th>lmm-CMA</th>
<th>spu (n)</th>
<th>CMA-ES</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{\text{Schwefel}}$</td>
<td>2 6 3</td>
<td>186±5</td>
<td>2.0</td>
<td>$81\pm5$</td>
<td>4.5</td>
<td>370±32</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 8 3</td>
<td>289±9</td>
<td>3.0</td>
<td>$145\pm7$</td>
<td>6.0</td>
<td>879±60</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 8 3</td>
<td>344±9</td>
<td>3.2</td>
<td>1112±72</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 10 3</td>
<td>558±18</td>
<td>3.6</td>
<td>$282\pm11$</td>
<td>7.1</td>
<td>2010±82</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 10 3</td>
<td>801±36</td>
<td>3.3</td>
<td>2667±87</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 11 3</td>
<td>2204±74</td>
<td>2.3</td>
<td>$626\pm17$</td>
<td>8.2</td>
<td>5156±101</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 12 4</td>
<td>3531±179</td>
<td>2.0</td>
<td></td>
<td>7042±172</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 14 4</td>
<td>8933±337</td>
<td>1.7</td>
<td></td>
<td>15072±377</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40 15 5</td>
<td>13404±281</td>
<td>1.7</td>
<td></td>
<td>22400±289</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{\text{Schwefel}/4}$</td>
<td>2 6 3</td>
<td>551±12</td>
<td>2.8</td>
<td>n:413±25</td>
<td>3.7</td>
<td>1527±76</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 8 3</td>
<td>783±8</td>
<td>3.6</td>
<td>n:971±36</td>
<td>2.9</td>
<td>2847±109</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 8 3</td>
<td>914±15</td>
<td>3.8</td>
<td>n:1302±31</td>
<td>2.7</td>
<td>3505±114</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 10 3</td>
<td>1366±35</td>
<td>4.3</td>
<td></td>
<td>5892±116</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 10 3</td>
<td>1774±37</td>
<td>4.1</td>
<td></td>
<td>7220±206</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 11 3</td>
<td>4193±88</td>
<td>3.0</td>
<td></td>
<td>12411±198</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 12 4</td>
<td>6138±82</td>
<td>2.5</td>
<td></td>
<td>15600±294</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 14 4</td>
<td>14796±310</td>
<td>2.0</td>
<td></td>
<td>29378±330</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40 15 5</td>
<td>22658±390</td>
<td>1.8</td>
<td></td>
<td>41534±466</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{\text{Rosenbrock}}$</td>
<td>2 6 3</td>
<td>511±84</td>
<td>1.4</td>
<td>n:252±52</td>
<td>2.8</td>
<td>700±194</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 8 3</td>
<td>775±108</td>
<td>2.8</td>
<td>n:719±54</td>
<td>(0.85)</td>
<td>3.0</td>
<td>2187±376</td>
<td>(0.85)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 8 3</td>
<td>914±15</td>
<td>3.8</td>
<td>n:1302±31</td>
<td>2.7</td>
<td>3505±114</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 10 3</td>
<td>1366±35</td>
<td>4.3</td>
<td></td>
<td>5892±116</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 10 3</td>
<td>1774±37</td>
<td>4.1</td>
<td></td>
<td>7220±206</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 11 3</td>
<td>4193±88</td>
<td>3.0</td>
<td></td>
<td>12411±198</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 12 4</td>
<td>6138±82</td>
<td>2.5</td>
<td></td>
<td>15600±294</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 14 4</td>
<td>14796±310</td>
<td>2.0</td>
<td></td>
<td>29378±330</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40 15 5</td>
<td>22658±390</td>
<td>1.8</td>
<td></td>
<td>41534±466</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{\text{NoisySphere}}$</td>
<td>2 6 3</td>
<td>0.35</td>
<td>413±14</td>
<td>1.0</td>
<td>n:109±12</td>
<td>3.7</td>
<td>407±61</td>
<td>(0.95)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 8 3</td>
<td>0.25</td>
<td>428±46</td>
<td>2.0</td>
<td>n:236±19</td>
<td>3.6</td>
<td>844±141</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 8 3</td>
<td>0.22</td>
<td>480±66</td>
<td>2.1</td>
<td></td>
<td>1014±68</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 10 3</td>
<td>0.18</td>
<td>630±76</td>
<td>2.6</td>
<td>n:636±33</td>
<td>2.6</td>
<td>1663±140</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 10 3</td>
<td>0.15</td>
<td>766±90</td>
<td>(0.95)</td>
<td>2.7</td>
<td></td>
<td>2058±148</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 11 3</td>
<td>0.13</td>
<td>1119±115</td>
<td>2.8</td>
<td>n:2156±216</td>
<td>1.4</td>
<td>3120±168</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 12 4</td>
<td>0.11</td>
<td>1361±212</td>
<td>2.8</td>
<td></td>
<td>3777±127</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 14 4</td>
<td>0.09</td>
<td>1997±247</td>
<td>2.9</td>
<td></td>
<td>5767±162</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40 15 5</td>
<td>0.08</td>
<td>2409±120</td>
<td>2.9</td>
<td></td>
<td>7032±173</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{\text{Ackley}}$</td>
<td>2 6 3</td>
<td>352±39</td>
<td>2.1</td>
<td>n:227±23</td>
<td>3.2</td>
<td>735±55</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 8 3</td>
<td>540±29</td>
<td>(0.95)</td>
<td>2.9</td>
<td></td>
<td>1577±83</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 8 3</td>
<td>566±33</td>
<td>3.4</td>
<td>n:704±24</td>
<td>(0.90)</td>
<td>2.2</td>
<td>1904±122</td>
<td>(0.95)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 10 3</td>
<td>800±22</td>
<td>(0.95)</td>
<td>3.8</td>
<td></td>
<td>3056±114</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 10 3</td>
<td>892±28</td>
<td>4.1</td>
<td>n:2066±119</td>
<td>(0.95)</td>
<td>1.8</td>
<td>3641±154</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 11 3</td>
<td>1530±39</td>
<td>3.7</td>
<td></td>
<td>5672±151</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 12 4</td>
<td>1884±50</td>
<td>3.5</td>
<td>8150±196</td>
<td>0.8</td>
<td>6641±108</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 14 4</td>
<td>2747±62</td>
<td>3.7</td>
<td></td>
<td>10663±203</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>40 15 5</td>
<td>3690±80</td>
<td>3.3</td>
<td></td>
<td>12084±247</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{\text{Ellis}}$</td>
<td>2 6 3</td>
<td>393±19</td>
<td>2.0</td>
<td></td>
<td>774±73</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>4 8 3</td>
<td>582±24</td>
<td>2.9</td>
<td></td>
<td>1688±11</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>5 8 3</td>
<td>683±33</td>
<td>3.4</td>
<td></td>
<td>2342±102</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>8 10 3</td>
<td>1142±53</td>
<td>4.0</td>
<td></td>
<td>4542±155</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>10 10 3</td>
<td>1628±95</td>
<td>3.8</td>
<td></td>
<td>6211±264</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>16 11 3</td>
<td>4706±148</td>
<td>2.8</td>
<td></td>
<td>13177±341</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>20 12 4</td>
<td>8250±393</td>
<td>2.3</td>
<td></td>
<td>19060±501</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>32 14 4</td>
<td>27281±753</td>
<td>1.6</td>
<td>32 14</td>
<td>14796±539</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>96 40 15 5</td>
<td>33602±548</td>
<td>2.1</td>
<td></td>
<td>69642±644</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$f_{\text{Rastrigin}}$</td>
<td>2 50 25</td>
<td>164±242</td>
<td>(0.6)</td>
<td>1.2</td>
<td>n:528±48</td>
<td>(0.95)</td>
<td>3.6</td>
<td>1970±418</td>
<td>(0.85)</td>
</tr>
<tr>
<td></td>
<td>5 140 70</td>
<td>23293±1374</td>
<td>(0.3)</td>
<td>0.5</td>
<td>n:4037±209</td>
<td>(0.60)</td>
<td>3.0</td>
<td>12310±1098</td>
<td>(0.75)</td>
</tr>
</tbody>
</table>
4.2 ACM with Pre-selection

![Graph 1](image1.png)

![Graph 2](image2.png)

Figure 4.9: **Left:** the cost of model learning/testing increases quasi-linearly with $n$. **Right:** the average speedup and speedup for all problems except Rastrigin.

function. The speedup w.r.t. CMA-ES is depicted in Figure 4.9-**Right** versus the problem dimension $n$. Interestingly, the speedup reaches its peak for $n$ ranging in 8..10, then it decreases – except on the Noisy Sphere function. A possible explanation is that the noise level is comparatively less when the dimension increases (as in [Bouzarkouna et al., 2010]), enabling the regularization involved in the model optimization to counteract the noise effects.

On the negative side, ACM-ES performs poorly on $f_{Rastrigin}$ function, and only solves it marginally for dimensions $n > 8$. This failure is attributed to the fact that ACM-ES does not handle well multi-modal diversity; it tends to accelerate the premature convergence to a local optimum, thus amplifying the weakness of CMA-ES without restarts on this benchmark problem: the best-performing versions of CMA-ES require an increasing population size [Hansen, 2009]. Further work will consider the use of niching techniques as well as model quality control strategies to overcome this weakness.

### 4.2.4 Preliminary Conclusion

The main novelty of the proposed ACM-ES is that all steps of the algorithm preserve invariance with respect to *both* monotonous transformations of the fitness function and orthogonal transformations of the search space. Comparison-based invariance is enforced by using Ranking Support Vector Machines to learn the surrogate model; coordinate invariance is enforced through using the covariance matrix adapted by CMA-ES as SVM kernel. Experimental validation confirms both invariance claims, and demonstrates the merits of the approach in terms of fitness evaluations and scalability w.r.t. the space dimension.

The main weakness of the approach is due to the failure of the surrogate model to account for multi-modal landscapes, as shown on the Rastrigin function; some improvements,
Figure 4.10: Rank-based surrogate error vs number of function evaluations, during a representative run of CMA-ES on 10-dimensional Sphere (Left) and Rosenbrock (Right) functions. See text for details.

e.g., related to model control, will be discussed in the next Section. Another issue regards the surrogate model hyper-parameters, which have been calibrated after preliminary experiments on the Sphere function conditionally to the carefully tuned hyper-parameters of CMA-ES [Hansen and Ostermeier, 2001]. A global approach, considering both sets of hyper-parameters in an integrated way, would be appropriate. Another perspective, pointed out in [Runarsson, 2004], is to extend the approach to constrained optimization.

4.3 Self-adaptive Surrogate-assisted CMA-ES

In this Section, we present a surrogate-adaptation mechanism (***) which can be used on top of any surrogate optimization approach. *** adapts online the number of generations after which the surrogate is re-trained, referred to as the surrogate lifelength; further, it adaptively optimizes during the search the surrogate hyper-parameters using an embedded CMA-ES module. A proof of principle of the approach is given by implementing *** on top of ACM-ES, a surrogate-assisted variant of CMA-ES, yielding the ACM-ES algorithm. To our best knowledge, the self-adaptation of the surrogate model within CMA-ES and by CMA-ES is a new contribution. The merits of the approach are shown as ACM-ES shows significant improvements compared to CMA-ES and ACM-ES on the BBOB-2012 noiseless and noisy testbeds.

The results of this Section have been published as [Loshchilov et al., 2012e, Loshchilov et al., 2012b, Loshchilov et al., 2012c].

4.3.1 Preliminary Analysis

Following the characterization proposed in [Jin, 2005], ACM-ES, discussed in detail in the previous Section, is a surrogate-assisted optimizer with an individual-based evolution con-
4.3 Self-adaptive Surrogate-assisted CMA-ES

trol. As in many other pre-selection methods (see Section 3.3.2.1 for a detailed overview), at each generation ACM-ES generates $\lambda_{pre}$ individuals, where $\lambda_{pre}$ is much larger than population size $\lambda$. Then $\lambda_{pre}$ pre-children are evaluated and ranked using surrogate model $\hat{f}$. The most promising $\lambda'$ (not always the best) pre-children are selected and evaluated using the true expensive function, yielding new points ($x, f(x)$). When the true objective function of $\lambda'$ individuals is known, the ranking of other $\lambda - \lambda'$ points can be approximated.

While the experimental results demonstrate the improvements brought by ACM-ES on some functions (about 2-4 times faster than CMA-ES on Rosenbrock, Ellipsoid, Schwefel, Noisy Sphere and Ackley functions up to dimension 20, see Figure 4.9), they also show a loss of performance on the multi-modal Rastrigin function. Complementary experiments suggest that:

1. on highly multi-modal functions, the surrogate model happens to suffer from a loss of accuracy; in such cases some control is required to prevent the surrogate model from misleading the search;

2. surrogate-assisted algorithms may require a larger population size for multi-modal problems.

The lack of surrogate control appears to be an important drawback in ACM-ES. This control should naturally reflect the current surrogate accuracy. A standard measure of the rank-based surrogate error is given as the fraction of violated ranking constraints on the test set (see Section 3.3.1.1 and Eq. 4.9). Error value 0 (respectively, 0.5) corresponds to a perfect surrogate (respectively, random guessing).

4.3.1.1 Model Quality Estimation

Before doing surrogate-assisted optimization one should be sure that the model provides a reasonable prediction of the objective function. Figure 4.10 illustrates the surrogate model error during a representative run of CMA-ES (with re-training at each iteration, but without any exploitation of the model) on 10-dimensional Sphere (Left) and Rosenbrock (Right) functions using 100 and 250 training points, respectively. Thin blue curves correspond to the model error $Err(\hat{f})$ measured with Eq. (4.9) on $\lambda$ offspring (test) solutions, which were not used to build $\hat{f}$. Dotted curves correspond to fitness of the mean $m$ of the mutation distribution of CMA-ES. A fluctuation of $Err(\hat{f})$ is clearly observable because i). the model quality changes depending on used training points, hyper-parameters, etc.; ii). the estimation of $Err(\hat{f})$ is imprecise because $\lambda$ is too small ($\lambda = 10$ for 10-dimensional problems). To make this estimation more robust one of the following approaches can be used:

1. Use larger $\lambda$. This may increase the overall runtime of CMA-ES.

2. Exclude some training points and use them as test points. These points should be located in the actual search region of CMA-ES. Exclusion of such points may worsen the model.
3. Use cross-validation, e.g., leave-one-out cross-validation (see Section 3.3.1.2). This may significantly increase the computational complexity of the algorithm.

4. Use some relaxation of $\text{Err}(\hat{f})$. The results may be sensitive to the parametrization of the relaxation (also true for all approaches described above).

For a better numerical stability without additional computational cost, we suggest to update the surrogate error using additive relaxation with relaxation constant $\beta_{\text{Err}}$:

$$
\text{Err}(\hat{f}) = (1 - \beta_{\text{Err}})\text{Err}(\hat{f}) + \beta_{\text{Err}}\text{Err}(\hat{f})_{\text{loc}},
$$

(4.14)

where $\text{Err}(\hat{f})_{\text{loc}}$ is the local model error estimated on $\lambda$ points of the current generation using Eq. (4.9). We experimentally found that $\beta_{\text{Err}} = 0.2$ represents a reasonable trade-off between the use of local and global information. This parameter indeed is problem- and model control approach-dependent. It can be adjusted if some relevant information about the problem at hand is available. As the initial setting of $\text{Err}(\hat{f})$ we choose a pessimistic value 0.5 (random prediction) to avoid a potential overestimation of $\text{Err}(\hat{f})$ in the first generations of CMA-ES. As can be seen in Figure 4.10, the relaxed model error (bold blue lines) is quite robust w.r.t. outliers, and at the same time is sensitive to local information.

### 4.3.1.2 Model Exploitation

As can be seen in Figure 4.10, after the first generations of CMA-ES, the surrogate error decreases to approximately 10%. This better than random prediction can be viewed as a source of information about the function which can be used to improve the search.

Let $\hat{n}$ denote the number of generations a surrogate model is used, referred to as surrogate lifespan. In so-called generation-based evolution control methods [Jin, 2005], the surrogate $\hat{f}$ is directly optimized for $\hat{n}$ generations, without requiring any true (expensive) objective computations. Then, the following generation considers the objective function $f$, and yields instances to enrich the training set, relearn or refresh the surrogate and adjust some parameters of the algorithm. The surrogate lifespan $\hat{n}$ can be fixed or adapted.

The above-described procedure, as well as several approaches which we will present in this Section, can be used in principle with any optimization algorithm. In this work, however, we consider CMA-ES and its active and restart versions by the reasons discussed in Section 2.3.3.

An experimental study of the impact of $\hat{n}$ is displayed in Figure 4.11, showing the speedup reached by direct surrogate optimization on several 10-dimensional benchmark problems vs the number of generations $\hat{n}$ the surrogate is used. A factor of speedup 1.7 is obtained for $\hat{n} = 1$ on the Rotated Ellipsoid function, close to the optimal speedup 2.0. A speedup ranging from 2 to 4 is obtained for IPOP-aACM-ES with surrogates for $\hat{n}$ in $[5, 15]$. As could have been expected again, the optimal value of $\hat{n}$ is problem-dependent and widely varies. In the case of the Attractive Sector problem for instance, the surrogate model is imprecise and not useful, so $\hat{n} = 0$ should be used (thus falling back to the original IPOP-aCMA-ES with no surrogate) to prevent the surrogate from misleading the search.
4.3 Self-adaptive Surrogate-assisted CMA-ES

Figure 4.11: The speedup of IPOP-aACM-ES over IPOP-aCMA-ES, where speedup = 2.0 means that IPOP-aACM-ES with a given lifelength $\hat{n}$ of the surrogate model, requires 2.0 times less computational effort SP1 (i.e., the average number of function evaluations of successful runs divided by proportion of successful runs) than IPOP-aCMA-ES to reach the target objective value of $f_t = f_{opt} + 10^{-8}$. The results of the IPOP-aCMA-ES are given when number of generations is zero.

4.3.1.3 Adjusting Surrogate Lifelength

A natural idea is then to adjust lifelength $\hat{n}$ can be adjusted depending on the error made by the surrogate $\hat{f}$ on the new $\lambda$ points evaluated on $f$. If this error is 0, then $\hat{f}$ is perfectly accurate and could have been used for some more generations before learning a new $\hat{f}$. In this case, lifelength $\hat{n}$ is set to the maximum value $\hat{n}_{\text{max}}$, which corresponds to the maximum theoretical speedup of the ACM-ES. If the error is circa 0.5, surrogate $\hat{f}$ provides no better indications than random guessing and thus misleads the optimization: $\hat{n}$ is set to 0. More generally, considering an error threshold $\tau_{err}$, we decided to adjust $\hat{n}$ between $\hat{n}_{\text{max}}$ and 0, proportionally to the ratio between the actual error and the error threshold $\tau_{err}$ (bold curve in Figure 4.12):

$$\hat{n} \leftarrow \left[ \frac{\tau_{err} - \text{Err}(\hat{f})}{\tau_{err} \hat{n}_{\text{max}}} \right].$$  \hfill (4.15)

4.3.1.4 Alternative Approaches to Surrogate Model Exploitation

To try to improve the results of ACM, we also tested a trust-region-based algorithm, where $\hat{f}$ is optimized to find a candidate solution $\mathbf{x}^*$ to be evaluated on $f$. The trust-region is bounded using the Mahalanobis distance defined by the covariance matrix of CMA-ES. We found that the results of ACM on Ellipsoid function can be improved, but

---

3Complementary experiments show that the best adjustment of $\hat{n}$ depending on the surrogate error is again problem-dependent, but that the linear adjustment proposed here is a good compromise.
in larger dimensions \((n > 10)\) this strategy becomes too "greedy", the training set is not sufficiently diverse to build an accurate \(\hat{f}\). Thus, some diversity preservation procedure must be considered. The most important drawback of the trust-region-based search is the evaluation of (usually) only one solution on \(f\) per iteration. This makes the algorithm not suitable for parallel evaluations of solutions, which is the main source of speedup in real-world expensive optimization. Another important detail is that the algorithm actually not resembles anymore CMA-ES and its performance is not stable on noisy functions.

Instead of evaluating \(\lambda\) individuals after \(\hat{n}\) generations on \(\hat{f}\) we may use alternative strategies, e.g., evaluate on \(f\) one best or random individual per generation. We found that this may improve the results, however, there are at least two issues: i). again, the algorithm cannot be efficiently parallelized; ii). CMA-ES never observes the correct ranking of all individuals, thus, a potential misleading introduced by the surrogate model may be heavily corrected.

We also tested several strategies to control \(\hat{n}\), including the following ones:

1. Optimal control of \(\hat{n}\) can be viewed as an operator selection problem with a fixed number of arms/operators, e.g., 21 arms for \(\hat{n} = 0, \ldots, 20\) (see Figure 4.11). To find an optimal \(\hat{n}\), Multi-Armed Bandits paradigm can be applied (see, e.g., [Fialho et al., 2010]). We also tested an Elo-based (chess rating system named after Arpad Elo, a master-level chess player) system, where in each game we check two arms \(i\) and \(j\) and the one whose \(\lfloor \lambda/2 \rfloor\) offspring are statistically better wins the game. Using Elo-based approach we may find the rating for different \(\hat{n}\), an important detail is that even if some arm is rarely used it may quickly reach high Elo rating if wins against another high-rated arm. The approach can be extended to the competition of teams of arms using generalized Elo-system proposed in [Dangauthier et al., 2007, Coulom, 2007].

2. The search for an optimal \(\hat{n}\) can be viewed as a one-dimensional noisy optimization problem. We may directly apply ESs to solve this problem. For each generated offspring defined by a decision variable \(\hat{n}\) we run CMA-ES on \(\hat{f}\) for \(\hat{n}\) generations and evaluate one best, mean or random individual of the \(\hat{n}\)-th population on \(f\).
4.3 Self-adaptive Surrogate-assisted CMA-ES

Then, we may compare different alternatives (solutions) of \( \hat{n} \), choose new mean of distribution, adapts parameters of ES and repeat the procedure again. It should be noted that the minimum variance must be fixed to prevent the algorithm from premature convergence on an integer value of \( \hat{n} \), given that the problem is dynamic and noisy.

3. An alternative to the previously described strategy would be to consider only two candidate generations at a time, generations \( \hat{n} \) and \( \hat{n} + 1 \). We may run CMA-ES on \( \hat{f} \) for \( \hat{n} \) generation and evaluate \( \lfloor \lambda/2 \rfloor \) individuals on \( f \) (without updating \( \hat{f} \)), then run for one more generation and evaluate the remaining \( \lambda - \lfloor \lambda/2 \rfloor \) individuals. We may check whether \( \hat{n} + 1 \)-th individuals are statistically significantly better or not, if this is the case, we may increment \( \hat{n} \) and vise versa. If there is no difference, then we may suppose that we are around the maximum of curves illustrated in Figure 4.11. However, it may happen that the one-dimensional curve is flat and we will (almost randomly) walk unreasonably far away. The latter becomes a problem if the model quality drops quickly, then we should quickly decrease \( \hat{n} \), but doing so decrementally may take a long time so that the algorithm finally diverges. As an alternative we used Wilcoxon rank-sum test to estimate the significance of difference between \( \hat{n} \)-th and \( \hat{n} + 1 \)-th generations to set the scale of the increasing/decreasing factor of \( \hat{n} \) correspondingly.

The described above approaches have at least three important drawbacks:

1. When \( \lambda \) points from different generations are evaluated, we have a dilemma to choose which state of CMA-ES will be used as the actual one for the next generation. We may compute this state as a weighted sum of all states, which were used to generate \( \lambda \) points. However, this may potentially introduce an instability to CMA-ES.

2. If the model quality drops relatively quickly, it may take some time to adjust \( \hat{n} \) accordingly.

3. If the model is random guessing (\( Err(\hat{f})=0.5 \)), then we will still need to check \( \hat{n} = 0 \) (no surrogate mode used = no loss) and \( \hat{n} = 1 \) (one generation of divergence). Thus, when the model badly predicts \( f \), which is often the case on multi-modal and/or noisy functions, the algorithm will often diverge because of \( \hat{n} = 1 \).

To summarize, we choose the approach described in Section 4.3.1.3 as a baseline for our experiments because it allows to prevent divergence if \( \hat{f} \) is random guessing, the evaluation of all \( \lambda \) individuals can be easily parallelized, and the implementation of the approach is straightforward. However, we strongly believe that the other approaches described above are prospective and can be superior to the error model-based one under certain conditions.

4.3.2 Self-adaptive Surrogate-assisted CMA-ES

In this Section, we describe the novel surrogate adaptation mechanism resulting from previous discussions, which can be used in principle on top of any iterative population-based
optimizer without requiring any significant modifications thereof. The approach is illustrated on top of CMA-ES and ACM-ES. The resulting algorithm, \(*ACM-ES\), maintains a global hyper-parameter vector \(\theta = (\theta_{\text{opt}}, \theta_{\text{sur}}, \hat{n}, \mathcal{A}, \alpha)\), where:

- \(\theta_{\text{opt}}\) stands for the optimization parameters of the outer CMA-ES used for expensive function optimization (see Figure 4.13);
- \(\theta_{\text{sur}}\) stands for the optimization parameters of the internal CMA-ES used for surrogate model hyper-parameters optimization;
- \(\hat{n}\) is the number of internal generations during which the current surrogate model is used;
- \(\mathcal{A}\) is the archive of all points \((x_i, f(x_i))\) for which the true objective function has been computed, exploited to train the surrogate function;
- \(\alpha\) stands for the surrogate hyper-parameters.

All hyper-parameters are indexed by the current outer generation \(g\); by abuse of notations, the subscript \(g\) is omitted when clear from the context.

### 4.3.2.1 Overview of \(*ACM-ES\)

Let \(\text{GenCMA}(h, \theta_h, \mathcal{A})\) denote the elementary optimization module (here one generation of CMA-ES) where \(h\) is the function to be optimized (the true objective \(f\) or the surrogate \(\hat{f}\)), \(\theta_h\) denotes the current optimization parameters (e.g., CMA-ES step-size and covariance matrix, etc.) associated to \(h\), and \(\mathcal{A}\) is the archive of \(f\). After each call of \(\text{GenCMA}\), optimization parameters \(\theta_h\) are updated; and if \(\text{GenCMA}\) was called with the true objective function \(f\), archive \(\mathcal{A}\) is updated and augmented with the new points \((x_i, f(x_i))\). Note that \(\text{GenCMA}\) can be replaced by any black-box optimization procedure, as long as it is able to update its own optimization parameters and the archive.

\(*ACM-ES\) starts by calling \(\text{GenCMA}\) for \(g_{\text{start}}\) number of generations with the true objective \(f\), where \(\theta_{\text{opt}}\) and \(\mathcal{A}\) are respectively initialized to the default parameter of CMA-ES and the empty set (Algorithm 4.2, lines 4-7). In this starting phase, optimization parameter \(\theta_{\text{opt}}\) and archive \(\mathcal{A}\) are updated in each generation.

Then \(*ACM-ES\) iterates a six-step process (Algorithm 4.2, lines 9 - 22, illustrated in Figure 4.13):

1. Learning surrogate \(\hat{f}\) using surrogate model hyper-parameters \(\alpha\) (e.g., number of training points, parameters of kernel functions) and the current optimization parameters \(\theta_{\text{opt}}\) such as covariance matrix \(C\) (procedure BuildSurrogateModel, line 9; Section 4.1.1).

2. Optimizing the surrogate \(\hat{f}\) for \(\hat{n}\) generations (lines 11-13). This step classically calls \(\text{GenCMA}(\hat{f}, \theta_{\text{opt}}, \mathcal{A})\) for \(\hat{n}\) consecutive generations; \(\theta_{\text{opt}}\) is updated accordingly while \(\mathcal{A}\) is unchanged since this step does not involve any computation of the expensive \(f\).

3. Optimizing the expensive function \(f\) for one generation (line 15). In this step, we sample and evaluate \(\lambda\) individuals on \(f\) and update \(\theta_{\text{opt}}\) parameters of CMA-ES accordingly, updating and augmenting archive \(\mathcal{A}\) with new \((x, f(x))\) points.
4 Estimating model error $Err(\hat{f})$ using Eq. (4.9) on $\lambda$ recently evaluated points (line 17).

5 Adjusting the surrogate lifelength $\hat{n}$ using Eq. (4.15) (line 19, Section 4.3.1.3).

6 Optimizing surrogate model hyper-parameters $\alpha$ (line 21).

The contribution regards the adjustment of the surrogate hyper-parameters $\alpha$ (e.g., the number of the training points in $A$; the weights of the constraint violations in Ranking SVM, see Table 4.3), which are adjusted to optimize the quality of the surrogate $Err$ (Eq. 4.9). Formally, to each surrogate hyper-parameter vector $\alpha$ is associated a surrogate error $Err(\alpha)$ defined as follows (see Algorithm 4.3): hyper-parameter $\alpha$ is used to learn surrogate $\hat{f}_\alpha$ using $A_{g-1}$ as training set, and $Err(\alpha)$ is set to the ranking error of $\hat{f}_\alpha$ (Eq. 4.9), using the most recent $\lambda$ points from $A_g - A_{g-1}$ as test set.

The elementary optimization module $GenCMA(Err, \theta_{sur})$ (in this study, we do not use the archive parameter here) is launched for one generation (line 21), and the mean of the CMA-ES mutation distribution is used (line 22) as surrogate hyper-parameter vector in the next surrogate building phase (line 9).

Hyper-parameters might be optimized for more than one generation, however, this would increase the overall computational complexity of the algorithm, because $\lambda_{hyp}$ surrogates must be built at each generation. Since the hyper-parameters optimization problem is noisy, we suggest to perform one generation with a relatively large $\lambda_{hyp}$ instead of few generations with a small $\lambda_{hyp}$.
Algorithm 4.2: *ACM-ES

1: given $g \leftarrow 0; \text{Err} \leftarrow 0.5; A_g \leftarrow \emptyset$;
2: $\theta_{\text{opt}} \leftarrow \text{InitializationCMA}();$ // to optimize $f(x)$
3: $\theta_{\text{sur}} \leftarrow \text{InitializationCMA}();$ // to optimize $\text{Err}(\alpha)$
4: repeat
5:   $\{\theta_{\text{opt}}, A_g + 1\} \leftarrow \text{GenCMA}(f, \theta_{\text{opt}}, A_g)$;
6:   $g \leftarrow g + 1$;
7: until $g = g_{\text{start}}$ ;
8: repeat
9:   $\tilde{f}(x) \leftarrow \text{BuildSurrogateModel}(\alpha, A_g, \theta_{\text{opt}})$;
10:  $g_{\text{prev}} \leftarrow g$;
11: for $i = 1, \ldots, \tilde{n}$ do
12:   $\{\theta_{\text{opt}}, A_g + 1 = A_g\} \leftarrow \text{GenCMA}(\tilde{f}, \theta_{\text{opt}}, A_g)$;
13:   $g \leftarrow g + 1$;
14: end for
15: $\{\theta_{\text{opt}}, A_g + 1\} \leftarrow \text{GenCMA}(f, \theta_{\text{opt}}, A_g)$;
16: $g \leftarrow g + 1$;
17: $\text{Err}(\alpha) \leftarrow \text{MeasureSurrogateError}(\tilde{f}, \theta_{\text{opt}})$;
18: $\text{Err} \leftarrow (1 - \beta_{\text{Err}})\text{Err} + \beta_{\text{Err}}\text{Err}(\alpha)$;
19: $\tilde{n} \leftarrow \left\lfloor \frac{\tau_{\text{err}} - \text{Err}}{\tilde{n}_{max}} \right\rfloor$;
20: // adjust surrogate hyper-parameters
21: $\theta_{\text{sur}} \leftarrow \text{GenCMA}(\text{Err}, \theta_{\text{sur}})$;
22: $\alpha \leftarrow \theta_{\text{sur}, \text{m}}$; // detach the mean $m$ of mutation distribution stored in $\theta_{\text{sur}}$
23: until stopping criterion is met ;

Algorithm 4.3: Objective function $\text{Err}(\alpha)$ of surrogate model

1: given $\alpha$
2: $\tilde{f}(x) \leftarrow \text{BuildSurrogateModel}(\alpha, A_g_{\text{prev}}, \theta_{\text{sur,prev}})$;
3: $\text{Err}(\alpha) \leftarrow \text{MeasureSurrogateError}(\tilde{f}, \theta_{\text{opt,prev}})$;

4.3.3 Experimental Validation

The experimental validation of the approach proceeds by comparing the performance of *ACM-ES to the original CMA-ES and the active CMA-ES (aCMA-ES), considering IPOP and BIPOP restart scenarios with increasing population size (see Section 2.3.2 for a detailed description of CMA-ES variants).

The active IPOP-aCMA-ES [Hansen and Ros, 2010a] with the weighted negative covariance matrix update is found to perform equally well or better than IPOP-CMA-ES, which is explained as it more efficiently exploits the information of the worst $\lambda/2$ points. We use IPOP-aCMA-ES as challenging baseline, as it is a priori more difficult to speed up than the original IPOP-CMA-ES.
4.3 Self-adaptive Surrogate-assisted CMA-ES

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range for online tuning</th>
<th>Offline tuned value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\ell = N_{\text{training}}$</td>
<td>$[4n, 2(40 + \lfloor 4n^{1/7} \rfloor)]$</td>
<td>$40 + \lfloor 4n^{1/7} \rfloor$</td>
</tr>
<tr>
<td>$C_{\text{base}}$</td>
<td>$[0, 10]$</td>
<td>6</td>
</tr>
<tr>
<td>$C_{\text{pow}}$</td>
<td>$[0, 6]$</td>
<td>3</td>
</tr>
<tr>
<td>$c_{\sigma}$</td>
<td>$[0.5, 2]$</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 4.3: Surrogate hyper-parameters, default value and range of variation

Specifically, **ACM-ES is validated on the noiseless BBOB testbed by comparing IPOP-aACM-ES with fixed hyper-parameters, and IPOP-**aACM-ES (also called IPOP-saACM-ES) with online adaptation of hyper-parameters of the surrogate model⁴.

After detailing the experimental setting, this Section reports on the offline tuning of the number $\ell$ of points used to learn the surrogate model, and the online tuning of the surrogate hyper-parameters (line 21 of Algorithm 4.2).

4.3.3.1 Experimental Setting

The default BBOB stopping criterion is reaching target function value $f_t = f_{\text{opt}} + 10^{-8}$. Ranking SVM is trained using the $\ell$ most recently evaluated points; its stopping criterion is arbitrarily set to a maximum number of $1000\ell$ iterations of the SMO optimization procedure (see Section 4.1.1).

After a few preliminary experiments, the Ranking SVM constraint violation weights (Eq. 4.1) are set to

$$C_i = 10^{C_{\text{base}}(\ell - i)}C_{\text{pow}}$$

with $C_{\text{base}} = 6$ and $C_{\text{pow}} = 3$ by default; the cost of constraint violation is thus cubically higher for top-ranked samples. The $\sigma$ parameter of the RBF kernel is set to $\sigma = c_{\sigma}\sigma_x$, where $\sigma_x$ is the dispersion of the training points (their average distance after translation, Eq. 4.12) and $c_{\sigma}$ is set to 1 by default. The number $g_{\text{start}}$ of CMA-ES calls in the initial phase is set to 10, the maximum lifelength $\hat{n}_{\text{max}}$ of a surrogate model is set to 20. The error threshold $\tau_{\text{err}}$ is set to 0.45 and the error relaxation factor $\beta_{\text{Err}}$ is set to 0.2.

The surrogate hyper-parameters $\theta_{\text{sur}}$ are summarized in Table 4.3, with offline tuned value (default for IPOP-aACM-ES) and their range of variation for online tuning, (where $n$ stands for the problem dimension). Surrogate hyper-parameters are optimized with a population size 20 ($\lambda_{\text{hyp}} = 20$ surrogate models), where the Err function associated to a hyper-parameter vector is measured on the $\lambda$ most recently evaluated points in archive $A$, $\lambda$ being the current optimization population size.

⁴ For the sake of reproducibility we used the Octave/MatLab source code of IPOP-CMA-ES with default parameters, available from its author’s page, with the active flag set to 1. The **ACM-ES source code is available at https://sites.google.com/site/acmesgecco/.
Chapter 4. Single-Objective Surrogate-Assisted CMA-ES

Figure 4.14: The speedup of IPOP-aACM-ES over IPOP-aCMA-ES w.r.t. fixed number of training points on (Left) 10-dimensional and (Right) 20-dimensional BBOB benchmark problems.

4.3.3.2 Offline Tuning: Number of Training Points

It is widely acknowledged that the selection of the training set is an essential ingredient of surrogate learning [Jin, 2005, Ingimundardottir and Runarsson, 2011]. After some alternative experiments the training set includes the most recent $\ell$ points in the archive (see Section 4.3.3.2 for a discussion of different training set selection strategies). The study thus focuses on the tuning of $\ell$. Its optimal tuning is of course problem- and surrogate learning algorithm-dependent. Several tunings have been considered in the literature, for instance for 10-dimensional problems: $3\lambda$ for SVR [Ulmer et al., 2004]; 30 for RBF [Ulmer et al., 2003a]; $2n = 20$ for Kriging [Kruisselbrink et al., 2010]; 50 for ANN [Jin, 2005]; $2n = 20$ for Ranking SVM [Runarsson, 2006, Ingimundardottir and Runarsson, 2011]; $\frac{n(n+1)}{2} + 1 = 66$ for LWR in the lmm-CMA-ES [Bouzarkouna et al., 2010]; $\left\lfloor 30\sqrt{n} \right\rfloor = 95$ and $\left\lfloor 70\sqrt{n} \right\rfloor = 221$ for Ranking SVM in the ACM-ES (see Section 4.2.3).

In all above cases but ACM-ES, the surrogate models aim at local approximation. These approaches might thus be biased toward small $\ell$ values, as a small number of training points are required to yield good local models (e.g., in the case of the Sphere function), and small $\ell$ values positively contribute to the speedup. It is suggested however that the Sphere function might be misleading regarding the optimal adjustment of $\ell$.

Let us consider the surrogate speedup of IPOP-aACM-ES w.r.t. IPOP-aCMA-ES depending on (fixed) $\ell$, on unimodal benchmark problems from the BBOB noiseless testbed (Figure 4.14 for $n = 10, 20$). While the optimal speedup varies from 2 to 4, the actual speedup strongly depends on the number $\ell$ of training points.

Complementary experiments on $n$-dimensional problems with $n = 2, 5, 10, 20, 40$ yield to propose an average best tuning of $\ell$ depending on dimension $n$:

$$\ell = \left\lfloor 40 + 4n^{1.7} \right\rfloor$$

Eq. (4.16) is found to empirically outperform the one proposed for the original ACM-ES
4.3 Self-adaptive Surrogate-assisted CMA-ES

in Section 4.2.3 ($\ell = \lfloor 70\sqrt{n} \rfloor$), which appears to be biased to 10-dimensional problems, and underestimates the number of training points required in higher dimensions. Experimentally however, $\ell$ must increase super-linearly with $n$; Eq. (4.16) states that for large $n$ the number of training points should triple when $n$ doubles.

Further, Figure 4.14 shows that the optimal $\ell$ value is significantly smaller for the Sphere function than for other functions, which experimentally supports our conjecture that the Sphere function might be misleading with regard to the tuning of surrogate hyper-parameters.

4.3.3.3 Online Tuning: Surrogate Hyper-parameters

The IPOP-**aACM-ES achieves the online adaptation of the surrogate hyper-parameters within a specified range (Table 4.3), yielding the surrogate hyper-parameter values to be used in the next surrogate learning step.

Note that a surrogate hyper-parameter individual might be non-viable, i.e., if it does not enable to learn a surrogate model (e.g., the number of training points is negative).
Chapter 4. Single-Objective Surrogate-Assisted CMA-ES

Such non-viable individual is heavily penalized ($\text{Err}(\alpha) > 1$). In case no usable hyper-parameter individual is found (which might happen in the very early generations as it is shown in Figure 4.15), $\theta_{\text{sur}}$ is set to its default value.

The online adaptation of surrogate hyper-parameters however soon reaches usable hyper-parameter values. The trajectory of the surrogate hyper-parameter values vs the number of generations is depicted in Figure 4.15, normalized in $[0, 1]$ and considering the median out of 15 runs optimizing 20-dimensional Sphere, Rosenbrock, Rotated Ellipsoid and Discus functions.

On Ellipsoid, Discus and Rosenbrock functions the trajectory of $\ell$ displays three stages. In a first stage, $\ell$ increases as the overall number of evaluated points (all points are required to build a good surrogate). In a second stage, $\ell$ reaches a plateau; its value is close to the one found by offline tuning (see Section 4.3.3.2). In a third stage, $\ell$ steadily decreases. This last stage is explained as CMA-ES approaches the optimum of $f$ and gets a good estimate of the covariance matrix of the objective function. At this point the optimization problem is close to the Sphere function, and as can be seen from the results for the Sphere function, a good surrogate can be learned from comparatively few training points.

The trajectories of other surrogate hyper-parameters are more difficult to interpret, although they clearly show non-random patterns (e.g., $C_{\text{pow}}$ on Ellipsoid and Discus functions).

4.3.3.4 Comparative Performances

The comparative performance of $^{**}$ACM-ES combined with the original and the active variants of IPOP-CMA-ES is depicted in Figure 4.16, on the 10-dimensional Rotated Ellipsoid (Left) and Rosenbrock (Right) functions. In both cases, the online adaptation of the surrogate hyper-parameters yields a quasi constant speedup, witnessing the robustness of $^{**}$ACM-ES. On the Ellipsoid function, the adaptation of the covariance matrix is much faster than for the baseline, yielding same convergence speed as for the original CMA-
ES on Sphere function. On the Rosenbrock function the adaptation is also much faster, although there is clearly room for improvements.

The performance gain of $s^* ACM-ES$, explained from the online adjustment of the surrogate hyper-parameters, in particular $\ell$, confirms the fact that the appropriate surrogate hyper-parameters vary along search, and can be adjusted based on the accuracy of the current surrogate model. Notably, IPOP-$s^* ACM-ES$ almost always outperforms IPOP-ACM-ES, especially for $n > 10$ (in some cases, offline well-tuned hyper-parameters, indeed, may lead to better results).

### 4.3.3.5 Scalability w.r.t. Population Size

The default population size $\lambda_{default} = 4 + \lfloor 3\log n \rfloor$ is suggested to be the only CMA-ES parameter to possibly require manual tuning. Actually, $\lambda_{default}$ is well tuned for unimodal problems and only depends on the problem dimension. Increasing the population size usually does not decrease the overall number of function evaluations needed to reach an optimum. However, it allows one to reach the optimum faster in terms of number of generations. Increasing the population size and running the objective function computations in parallel is a source of speedup, which raises the question of $s^* ACM-ES$ scalability w.r.t. the population size.

Figure 4.17 shows the speedup of the IPOP-$s^* a ACM-ES$ compared to IPOP-aCMA-ES for unimodal 10-dimensional problems, when the population size $\lambda$ is set to $\gamma$ times the default population size $\lambda_{default}$ (the original IPOP-$s^* a ACM-ES$ is shown for $\gamma = 1$). In all experiments $g_{start}$ was set to 2, because using large populations we may quickly store sufficiently many points in the training set. For F8 Rosenbrock, F12 Cigar and F14 Sum of Different Powers the speedup remains almost constant and independent of $\gamma$, while for F10 Rotated Ellipsoid, F11 Discus and F13 Sharp Ridge, it even increases with $\gamma$. With a larger population size, “younger” points are used to build the surrogate model, that is hence more accurate.

The experimental evidence suggests that $s^* ACM-ES$ can be applied on top of parallelized versions of IPOP-$s^* a ACM-ES$, while preserving or even improving its speedup. Note that the same does not hold true for many surrogate-assisted methods; for instance in trust region methods, one needs to sequentially evaluate the points.

It is thus conjectured that further improvements of CMA-ES (e.g., refined parameter tuning, noise handling) will simultaneously be translated to $s^* ACM-ES$, without degrading its speedup.

### 4.3.3.6 Ranking SVM vs Rank-based SVR

We also tested IPOP-$s^* a ACM-ES$ with Rank-based SVR surrogate models on 20-dimensional Sphere, Rosenbrock, Rotated Ellipsoid and Discus functions. In all experiments we used so-called ’(1:N)’ scheme, where fitness of each individual corresponds to its rank in the training set (see Section 4.1.4). While it was expected that adaptive scaling of rank-based fitness assignment should lead to better results (i.e., adding the scaling factor to the set of adapted model hyper-parameters), our preliminary experiments do
not confirm this hypothesis. The adaptation of \( \epsilon \) of SVR also does not improve the results, therefore we set \( \epsilon = 0 \) for all experiments, that probably is a reasonable setting for noise-less problems (for a discussion of a dependency between the noise level and \( \epsilon \), see [Smola et al., 1998]). The lack of improvements from the use of more hyper-parameters may be due to a larger space of hyper-parameters, which requires larger \( \lambda_{hyp} \) as well as \( \lambda \) for a better model quality estimation.

The results are shown in Figure 4.18 and can be directly comparable with the results of Ranking SVM, presented in Figure 4.15. It can be seen that SVR-based IPOP-*aACM-ES is about 3.2, 2.6, 2.8 and 2.6 times slower than Ranking SVM-based IPOP-*aACM-ES on Sphere, Rosenbrock, Rotated Ellipsoid and Discus functions, respectively. Thus, the use of SVR surrogate models insignificantly improves the original aCMA-ES. We suppose that the main reason for these results is that SVR learning is more constrained than Ranking SVM learning in the sense that \( f \) values in SVR learning are pre-defined, while in Ranking SVM they are variables under constraints that the ranking is correct. Thus, the space of satisfactory solutions of Ranking SVM model is larger than of SVR, and, moreover, includes the latter. Finally, the bad results of SVR are not that surprising if one closely looks at Figure 4.4, where it is shown that Rank-based SVR relatively badly approximates the Sphere function.

The number of iterations of SMO optimization was set to \( 1000\ell \) for Ranking SVM and \( 10\ell \) for SVR. These numbers are not directly comparable, because one iteration of SVR learning includes more simple numerical operations, and, therefore, computationally more expensive. If the number of training points is fixed, then the learning of SVR with the given number of iterations is faster than the one of Ranking SVM. However, in our experiments with hyper-parameters adaptation we found that the learning and the full run of SVR actually becomes more expensive than the one of Ranking SVM, because SVR usually requires (as suggested by the adaptation) more training points that quickly increases the computational complexity. Finally, the full run of SVR takes about 60,
4.3 Self-adaptive Surrogate-assisted CMA-ES

Figure 4.18: Median trajectories of normalized surrogate hyper-parameters estimated on 15 runs of the IPOP-\textsuperscript{*}aACM-ES with Rank-based SVR on Sphere, Rotated Ellipsoid, Discus and Rosenbrock 20-dimensional BBOB benchmark problems. See Figure 4.15 for the results using Ranking SVM.

10, 10 and 15 times longer than the of Ranking SVM for Sphere, Rosenbrock, Rotated Ellipsoid and Discus functions, respectively. We also found that the use of larger than default population sizes improves the results for SVR, but it is still worse than Ranking SVM by a factor of about 2-3.

4.3.3.7 Benchmarking on Noiseless BBOB-2012 Testbed

For benchmarking on BBOB noiseless testbed (see Section 2.3.3 for a detailed description of BBOB framework) we consider four CMA-ES algorithms with restart scenario: IPOP-aCMA-ES [Hansen and Ros, 2010a], BIPOP-CMA-ES [Hansen, 2009], IPOP-\textsuperscript{*}aACM-ES and BIPOP-\textsuperscript{*}aACM-ES.

Results from experiments according to [Hansen \textit{et al.}, 2012] on the benchmark functions given in [Hansen \textit{et al.}, 2009a] are presented in Figures 4.20 and 4.21 and in Table 4.4. The expected running time (ERT), used in the figures and table, depends on a given target function value, $f_t = f_{opt} + \Delta f$, and is computed over all relevant trials.
(on the first 15 instances) as the number of function evaluations executed during each trial while the best function value did not reach $f_t$, summed over all trials and divided by the number of trials that actually reached $f_t$ [Hansen et al., 2012, Storn and Price, 1997].

**Statistical significance** is tested with the rank-sum test for a given target $\Delta f_t$ ($10^{-8}$ as in Figure 4.20) using, for each trial, either the number of needed function evaluations to reach $\Delta f_t$ (inverted and multiplied by $-1$), or, if the target was not reached, the best $\Delta f$-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

IPOP-*aACM-ES and BIPOP-*aACM-ES are the same algorithm (*aACM-ES) before the first restart occurs, therefore, the results are very similar for the unimodal functions, where the optimum can be found without restarts. The *aACM-ES outperforms aCMA-ES usually by a factor from 2 to 4 on $f_1$, $f_2$, $f_8$, $f_9$, $f_{10}$, $f_{11}$, $f_{12}$, $f_{13}$ and $f_{14}$ for dimensions between 5 and 20. The speedup in dimension 2 is less pronounced for problems, where the running time is too short to improve the search. This is the case for $f_5$ Linear Slope function, where the speedup can be observed only for dimension 20, because the optimum can be found after about 200 function evaluations. To improve the search on functions with small budgets it would make sense to use the surrogate model right after the first ($g_{\text{start}} = 1$) generation of the CMA-ES, while in this study, this parameter $g_{\text{start}}$ is conservatively set to 10 generations.

The good results on unimodal functions can be explained by the fact that, while using the same amount of information (all previously evaluated points), *aACM-ES processes this information in a more efficient way by constructing the approximation model of the function. Similar effect of more efficient exploitation of the available information can be observed for aCMA-ES in comparison to CMA-ES.

The speedup on multi-modal functions is less pronounced, because they are more difficult to approximate and the final surrogate model often has a bad precision. In this case, the adaptation of the number of generations leads to an oscillation of $\hat{n}$ close to 0, such that the surrogate model is not used for optimization or used for small number of generations.

The BIPOP versions of CMA-ES usually perform better than IPOP on $f_{23}$ and $f_{24}$, where the optimum is more likely to be found using small initial step-size. This leads to overall better performance of the BIPOP versions, and BIPOP-*aACM-ES in particular. The better performance of the latter in comparison with BIPOP-CMA-ES can be partially explained by the use of the active covariance matrix update. However, this is not the case for $f_{20} - f_{24}$ functions in 5-D and $f_{15} - 19$ in 20-D (see Figure 4.20).

The *aACM-ES algorithms improve BBOB-2010 records in dimension 10 and 20 on $f_7$, $f_{10}$, $f_{11}$, $f_{12}$, $f_{13}$, $f_{14}$, $f_{15}$, $f_{16}$, $f_{20}$.

We also compared the proposed algorithms with 43 optimization algorithms tested during the BBOB-2009 and BBOB-2010. Figure 4.19 shows empirical runtime distributions of these algorithms on all noise-less functions ($f_1 - f_{24}$) with target values in $\{10^2, \ldots, 10^{-8}\}$ in dimension 20. The results for dimension 40 are not presented here, because they are incomplete due to extremely long runtime of surrogate-assisted versions on multi-modal functions, where generally millions of function evaluations are needed to reach the optimum.
4.3 Self-adaptive Surrogate-assisted CMA-ES

Figure 4.19: Performance of 45 optimization algorithms, including IPOP-^aACM-ES and BIPOP-^aACM-ES on 20-dimensional noiseless BBOB benchmark problems. For the interpretation of axis-x and axis-y see Section 2.3.3.

It may be difficult to recognize curves and corresponding algorithms, however, it can be clearly seen that IPOP-^aACM-ES and BIPOP-^aACM-ES dominate all other optimization algorithms in the range [100n, 1000n] of functions evaluations used for optimization. In this range our surrogate-assisted algorithms outperform the closest competitor, IPOP-^aCMA-ES, by a factor between 2 and 3. This speedup is due to the good performances on non-separable ill-conditioned problems (see Figure 4.20), where IPOP-^aACM-ES outperforms all tested on BBOB algorithms, including BFGS and NEWUOA. On original f^8 and rotated f^9 Rosenbrock functions IPOP-^aACM-ES is as fast as derivative-based BFGS and at most 2 times slower than NEWUOA algorithm with quadratic meta-models, which are indeed well suitable for the quadratic form of the Rosenbrock function, but may fail under certain transformations of f.

For a budget smaller than 100n function evaluations, the proposed algorithms are
among the best performing ones, while the latter often benefit from exploiting functions separability or very simple landscape (see, e.g., \( f_1 \) Sphere, \( f_2 \) Separable Ellipsoid, \( f_5 \) Linear slope).

For a budget larger than 5000\( n \) function evaluations, BIPOP-*ACM-ES outperforms all tested algorithm except MOS (a hybrid of DE and CMA-ES, [LaTorre et al., 2010]) which exploits the separability and therefore performs well on \( f_3 \) Rastrigin and \( f_4 \) Skew-Rastrigin-Bueche functions. On multi-modal functions the advantages of using surrogate models are less significant because of difficult functions landscapes and limited number of training points together with high computational cost of surrogate learning. In this case, it might be more advantageous to use alternative restart strategies for CMA-ES which will lead to a comparable speedup but with a lower computational cost. We will investigate this question in the exploration part of this thesis, see Section 6.3.3.
4.3 Self-adaptive Surrogate-assisted CMA-ES

Figure 4.20: Expected running time (ERT in number of $f$-evaluations) divided by dimension for target function value $10^{-8}$ as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in the legend of $f_1$ and $f_{24}$. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Horizontal lines give linear scaling, slanted dotted lines give quadratic scaling. Black stars indicate statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ◦: BIPOP-CMA, ▽: BIPOP-saACM, ⋆: IPOP-aCMA, □: IPOP-saACM.
Figure 4.21: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{-8.2}$ for all functions and subgroups in 20-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target (generally with a different algorithm for different targets).
### Table 4.4: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 (given in the respective first row) for different $\Delta f$ values of *noise-less* functions in dimension 20. The central 80% range divided by two is given in braces. The median number of conducted function evaluations is additionally given in *italics*, if ERT($10^{-7}$) = $\infty$. #succ is the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. Best results are printed in bold.

<table>
<thead>
<tr>
<th>$\Delta f_{\text{opt}}$</th>
<th>IPOP-sa</th>
<th>BIPOP-s</th>
<th>BIPOP-C</th>
<th>IPOP-aC</th>
<th>BIPOP-C</th>
<th>BIPOP-s</th>
<th>BIPOP-aC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0e-0</td>
<td>$2\times 10^{4}$</td>
<td>$3\times 10^{4}$</td>
<td>$4\times 10^{4}$</td>
<td>$5\times 10^{4}$</td>
<td>$6\times 10^{4}$</td>
<td>$7\times 10^{4}$</td>
<td>$8\times 10^{4}$</td>
</tr>
<tr>
<td>2.0e-0</td>
<td>$3\times 10^{4}$</td>
<td>$4\times 10^{4}$</td>
<td>$5\times 10^{4}$</td>
<td>$6\times 10^{4}$</td>
<td>$7\times 10^{4}$</td>
<td>$8\times 10^{4}$</td>
<td>$9\times 10^{4}$</td>
</tr>
<tr>
<td>3.0e-0</td>
<td>$4\times 10^{4}$</td>
<td>$5\times 10^{4}$</td>
<td>$6\times 10^{4}$</td>
<td>$7\times 10^{4}$</td>
<td>$8\times 10^{4}$</td>
<td>$9\times 10^{4}$</td>
<td>$10\times 10^{4}$</td>
</tr>
<tr>
<td>4.0e-0</td>
<td>$5\times 10^{4}$</td>
<td>$6\times 10^{4}$</td>
<td>$7\times 10^{4}$</td>
<td>$8\times 10^{4}$</td>
<td>$9\times 10^{4}$</td>
<td>$10\times 10^{4}$</td>
<td>$11\times 10^{4}$</td>
</tr>
<tr>
<td>5.0e-0</td>
<td>$6\times 10^{4}$</td>
<td>$7\times 10^{4}$</td>
<td>$8\times 10^{4}$</td>
<td>$9\times 10^{4}$</td>
<td>$10\times 10^{4}$</td>
<td>$11\times 10^{4}$</td>
<td>$12\times 10^{4}$</td>
</tr>
<tr>
<td>6.0e-0</td>
<td>$7\times 10^{4}$</td>
<td>$8\times 10^{4}$</td>
<td>$9\times 10^{4}$</td>
<td>$10\times 10^{4}$</td>
<td>$11\times 10^{4}$</td>
<td>$12\times 10^{4}$</td>
<td>$13\times 10^{4}$</td>
</tr>
<tr>
<td>7.0e-0</td>
<td>$8\times 10^{4}$</td>
<td>$9\times 10^{4}$</td>
<td>$10\times 10^{4}$</td>
<td>$11\times 10^{4}$</td>
<td>$12\times 10^{4}$</td>
<td>$13\times 10^{4}$</td>
<td>$14\times 10^{4}$</td>
</tr>
<tr>
<td>8.0e-0</td>
<td>$9\times 10^{4}$</td>
<td>$10\times 10^{4}$</td>
<td>$11\times 10^{4}$</td>
<td>$12\times 10^{4}$</td>
<td>$13\times 10^{4}$</td>
<td>$14\times 10^{4}$</td>
<td>$15\times 10^{4}$</td>
</tr>
<tr>
<td>9.0e-0</td>
<td>$10\times 10^{4}$</td>
<td>$11\times 10^{4}$</td>
<td>$12\times 10^{4}$</td>
<td>$13\times 10^{4}$</td>
<td>$14\times 10^{4}$</td>
<td>$15\times 10^{4}$</td>
<td>$16\times 10^{4}$</td>
</tr>
<tr>
<td>1.0e-0</td>
<td>$11\times 10^{4}$</td>
<td>$12\times 10^{4}$</td>
<td>$13\times 10^{4}$</td>
<td>$14\times 10^{4}$</td>
<td>$15\times 10^{4}$</td>
<td>$16\times 10^{4}$</td>
<td>$17\times 10^{4}$</td>
</tr>
</tbody>
</table>

*Note: #succ indicates the number of trials that reached the final target $f_{\text{opt}} + 10^{-8}$. Best results are printed in bold.*
Benchmarking on Noisy BBOB-2012 Testbed

For benchmarking we consider $^{*}\text{ACM-ES}$ in IPOP restart scenario ($^{*}\text{aACM-ES}$) using default parameters and termination criteria as given in [Hansen and Ros, 2010b]. The only one parameter of the surrogate part of $^{*}\text{aACM-ES}$ different from the default one is the index of generation $g_{\text{start}}$ when we start to use the surrogate model. For noisy case we set $g_{\text{start}} = 5(i_{\text{restart}} + 1)$ instead of the default $g_{\text{start}} = 10$. We found that sometimes it makes sense to postpone the surrogate-assisted search if several restarts ($i_{\text{restart}}$) were performed.

Results from experiments according to [Hansen et al., 2012] on the benchmark functions given in [Hansen et al., 2009b] are presented in Figures 4.22, 4.23. Table 4.5 gives the Expected Running Time (ERT) for targets $10^1, -1, -3, -5, -7$ divided by the best ERT obtained during BBOB-2009 (given in the ERT$_{\text{best}}$ row) in 20-D.

The $^{*}\text{aACM-ES}$ outperforms $^{*}\text{aACM-ES}$ usually by a factor from 2 to 3 on functions with moderate noisy. This is the case for Sphere ($f_{101}$, $f_{102}$, $f_{103}$) and Rosenbrock ($f_{104}$, $f_{105}$, $f_{106}$) functions with Gaussian, Uniform and Cauchy noise models. It seems that the moderate noise only slightly affects the quality of the surrogate model and allows to have a speedup comparable to the one of the noise-less case.

On most functions with severe noise the surrogate model usually is not used ($\hat{n}$ oscillates around zero), because it gives a very imprecise prediction of the objective function. An exception is the 20-dimensional $f_{124}$ Schaffer function with Cauchy noise, where $^{*}\text{aACM-ES}$ requires about 7 times more functions evaluations to reach the optimum with $\Delta f_{\text{opt}} = 10^{-7}$ than $^{*}\text{aCMA-ES}$ (see Figure 4.22). However, according to Table 4.5, the performance for $\Delta f_{\text{opt}} = 10^{-5}$ is exactly the same for both algorithms, therefore, we suppose that the loss of performance for $\Delta f_{\text{opt}} = 10^{-7}$ can be explained by some influence of surrogate-assisted search on restart conditions. We also found that if default coefficients $c_1$ and $c_\mu$ for the covariance matrix update are used instead of noisy settings of $c_1/5$ and $c_\mu/5$, then $^{*}\text{aACM-ES}$ performs as well as $^{*}\text{aCMA-ES}$ on this function. However, the use of default coefficients worsen the results of $^{*}\text{aCMA-ES}$ and $^{*}\text{aACM-ES}$ on other functions.

We also observe some loss in performance of $f_{125}$ Griewank-Rosenbrock and $f_{128}$ Gallagher functions for $n \leq 5$. For $f_{115}$ we observe the speedup for $n = 5$ and loss for $n = 20$, probably because of the same reasons as for $f_{124}$.

The $^{*}\text{aACM-ES}$ improves (sometimes insignificantly) the BBOB-2010 records in dimension 20 on $f_{101}$, $f_{102}$, $f_{103}$, $f_{104}$, $f_{105}$, $f_{106}$, $f_{107}$, $f_{109}$, $f_{112}$, $f_{114}$, $f_{117}$, $f_{118}$, $f_{120}$, $f_{122}$, $f_{123}$, $f_{127}$, $f_{129}$. 

120
Figure 4.22: Expected running time (ERT) divided by dimension for target function value $10^{-8}$ as $\log_{10}$ values versus dimension. Different symbols correspond to different algorithms given in legend of $f_{101}$ and $f_{130}$. Light symbols give the maximum number of function evaluations from all trials divided by the dimension. Horizontal lines give linear scaling, the slanted dotted lines give quadratic scaling. Legend: o: IPOP-aCMA, ▼: IPOP-saACM
Figure 4.23: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{-8..2}$ for all functions and subgroups in 20-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.
Table 4.5: Expected running time (ERT) in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 (given in the respective first row) for different Δf values of noisy functions in dimension 20. The central 80% range divided by two is given in italics. The median number of conducted function evaluations is additionally given in italics, if ERT(10⁻⁷) = ∞. #succ is the number of trials that reached the final target fopt + 10⁻⁸. Best results are printed in bold.
4.3.3.9 CPU Timing Experiment

For the timing experiment the IPOP-**aACM-ES was run on noiseless $f_1$, $f_8$, $f_{10}$ and $f_{15}$ functions without self-adaptation of surrogate model hyper-parameters. The crucial hyper-parameter for CPU time measurements, the number of training points, was set to $\ell = \left\lfloor 40 + 4n^{1.7} \right\rfloor$ as a function of dimension $n$.

These experiments have been conducted on a single core with 2.4 GHz under Windows XP using Matlab R2006a.

On unimodal functions the time complexity of surrogate model learning usually increases cubically in the search space dimension (see Figure 4.24) and quadratically in the number of training points. For small dimensions ($n < 10$) the overall time complexity increases super-linearly in the dimension. The time complexity per function evaluation depends on the population size, because one model is used to estimate the ranking of all points of the population. This leads to a smaller computational complexity on multimodal functions, e.g., $f_{15}$ Rastrigin function, where the population becomes much larger after several restarts. The time complexity on noisy functions is more similar to the one of Rastrigin function, because in both cases the large populations are used.

The results presented here do not take into account the optimization of model hyper-parameters, where $\lambda_{hyp}$ surrogate models should be build at each iteration, which leads to an increase of CPU time per function evaluation by a factor of $\lambda_{hyp}$. For IPOP-**aACM-ES $\lambda_{hyp}$ was set to 20.

4.3.3.10 Quadratic Programming Problem Optimization

The number of iterations $n_{iter}$ used to optimized the QP problem 4.3 is left fixed to 1000$\ell$ in the hyper-parameters optimization procedure. However, it can also be adapted by setting $\text{Min}(n_{iter}) = 100\ell$ and $\text{Max}(n_{iter}) = 1500\ell$ (this would complement Table 4.3). Thus, the procedure of hyper-parameters adaptation could, if necessary, reduce the CPU
4.3 Self-adaptive Surrogate-assisted CMA-ES

![Figure 4.25:](image)

Figure 4.25: Median trajectories of normalized surrogate hyper-parameters (including $n_{\text{iter}}$) from 15 runs of the IPOP-* $\alpha$ ACM-ES with Ranking SVM on Sphere, Rotated Ellipsoid and Rosenbrock 20-dimensional BBOB benchmark problems.

The surrogate learning step involves a trade-off between the computational effort and the accuracy of the QP resolution. This trade-off can be controlled through adapting the number $n_{\text{iter}}$ of iterations, and/or selecting the $\alpha_i$s (see Section 4.1.1), e.g., selecting a working set selection based on gradient information of the Lagrangian $L$. In practice, the selection proceeds by computing $\Delta L$ for all $\alpha_i$s at each iteration of the algorithm, to
update the $\alpha_i$ which leads to the largest increase of the Lagrangian and $\Delta L$.

This trade-off has been empirically investigated, comparing the uniform and gradient-based selection strategies on 10-dimensional Sphere and Rosenbrock functions with 100 training points and 40-dimensional Sphere and Rosenbrock functions with 2000 training points. The training points, uniformly randomly drawn from $[-0.5, 0.5]^n$, are used to build Ranking SVM surrogate model with RBF kernel, where $\sigma$ parameter is defined as the average distance between the points. The constant $C$ of Ranking SVM is set to $10^6$ in all experiments. For both selection strategies the maximum number of iterations is $1000\ell$, for gradient-based selection an additional stopping criterion is used: the algorithm also stops if $\frac{\text{Max}(\Delta L)}{C^{2\ell}} < 10^{-20}$.

The results are displayed in Figure 4.26, showing the cumulative increase of Lagrangian $L$ and observed $\text{Max}(\Delta L)$ for uniform and gradient-based selection on 10-dimensional Sphere function. As expected, gradient-based selection converges faster to the optimum of the QP problem by selecting at each iteration the most promising $\alpha_i$ Lagrangian multiplier. The speedup w.r.t. uniform selection is of order 100 for small number of iterations (e.g., 10) and of order smaller than 10 for relatively large number of iterations (e.g., $10^4 - 10^5$). Indeed, these speedup factors are problem- and SVM hyper-parameters dependent. However, an important aspect that should be taken into account is that one operation of gradient-based selection is numerically more expensive than random or periodic selection. Therefore, the overall CPU time of the gradient selection-based surrogate learning might be longer than the one of the uniform selection-based learning. In order to investigate the time complexity of both approaches, we re-plotted the results with, on the x-axis, the "time" metric, where "time" is equal to the number of iterations for uniform selection and the number of iterations times a factor of how much one gradient
selection-based iteration is more expensive than uniform selection-based iteration.

Figure 4.27 shows the convergence to the optimum of the QP problem versus "time" for uniform selection and gradient selection-based strategies. As can be seen, the advantages of gradient-based selection becomes less obvious by taking into account the CPU time instead of the number of iterations. Moreover, uniform selection outperforms gradient-based selection on 10-dimensional Sphere and Rosenbrock functions in the sense of the cumulative increase of the Lagrangian at the typical stopping point of $n_{\text{iter}} = 1000\ell$. This was observed in our earlier experiments and has been discussed in Section 4.1.1. The results for 2000 training points suggest that both approaches are quite comparable at $500\ell \ldots 1000\ell$ iterations of uniform selection-based learning. However, at early stages of QP problem optimization, gradient-based selection clearly outperforms uniform selection. Thus, if the accuracy of QP problem solution at these early stages is sufficient for accurate
predictions of $f(x)$, then the CPU time complexity of surrogate model learning can be reduced by a factor of 10-100 by considering a gradient-based selection. Unfortunately, our preliminary experiments, as well as the results shown in Figure 4.25 (the adaptation of $n_{iter}$), suggest that a much lower accuracy of QP problem solutions is not sufficient to build accurate surrogate models. These preliminary findings, however, should be very carefully verified considering more objective functions and larger dimensions, since the gradient-based selection is one of the most straightforward ways to reduce the time complexity of surrogate learning procedure and scale the algorithm to larger problem dimensions. Another possibility, to be considered in further work, might be to leave the decision of gradient-based versus uniform selection, to the hyper-parameter optimization step.

4.3.4 Conclusion and Perspectives

In this Section, we have presented a generic framework for adaptive surrogate-assisted optimization, which can in principle be combined with any iterative population-based optimization and surrogate learning algorithms. This framework has been instantiated on top of surrogate-assisted ACM-ES, using CMA-ES as optimization algorithm and Ranking SVM as surrogate learning algorithm. The resulting algorithm, $^*\text{ACM-ES}$, inherits from CMA-ES and ACM-ES the property of invariance w.r.t. monotonous transformations of the objective function and orthogonal transformations of the search space.

The main contribution of the $^*\text{ACM-ES}$ regards the online adjustment of i). the number $\hat{n}$ of generations a surrogate model is used, called surrogate lifelength; ii). the surrogate hyper-parameters controlling the surrogate learning phase. The surrogate life-length is adapted depending on the quality of the current surrogate model; the higher the quality, the longer the next surrogate model will be used. The adjustment of the surrogate hyper-parameters is likewise handled by optimizing them w.r.t. the quality of the surrogate model, without requiring any prior knowledge on the optimization problem at hand.

IPOP-$^*\text{aACM-ES}$ (respectively, BIPOP-$^*\text{aACM-ES}$) was found to improve on IPOP-aCMA-ES (respectively, BIPOP-CMA-ES) with a speedup ranging from 2 to 3 on unimodal $n$-dimensional functions from the BBOB-2012 noiseless testbed, with dimension $n$ ranging from 2 to 20. On multi-modal functions, IPOP-$^*\text{aACM-ES}$ is equally good or sometimes better than IPOP-aCMA-ES, although the speedup is less significant than for unimodal problems. Further, IPOP-$^*\text{aACM-ES}$ also improves on IPOP-aCMA-ES on problems with moderate noise from BBOB-2012 noisy testbed. BIPOP-$^*\text{aACM-ES}$ usually outperforms IPOP-$^*\text{aACM-ES}$ and BIPOP-CMA-ES on multi-modal functions (note that an active version of BIPOP-CMA-ES, BIPOP-aCMA-ES, will be presented in Section 6.3.3).

A long term perspective for further research is to better handle multi-modal (see Section 6.3.3) and noisy functions. A shorter-term perspective is to consider a more comprehensive surrogate learning phase, involving a portfolio of learning algorithms and using the surrogate hyper-parameter optimization phase to achieve portfolio selection. Another perspective is to design a tighter coupling of the surrogate learning phase, and the CMA-ES optimization, e.g., using the surrogate model $\hat{f}$ to adapt the CMA-ES hyper-parameters...
during the optimization of the expensive objective $f$.

### 4.4 Discussion

In this Chapter, we focused on single-objective surrogate-assisted CMA-ES algorithms for optimization of expensive problems of small to medium dimension. In the following, we analyze alternatives which may be considered for further research:

**Core Optimization Algorithm** CMA-ES seems to be a good candidate baseline optimizer, the only competitive algorithms are i) BFGS and NEWUOA algorithms which now perform not better than ACM-ES even on well suitable (for them) functions such as Rosenbrock, being sensitive to rank-preserving transformations of $f$; ii) Differential Evolution algorithms which are based on alternative strategies of optimization and have attracted a lot of attention in the last years (see, e.g., several DE algorithms in BBOB-2012), but are about one order of magnitude slower than ACM-ES.

**Surrogate Modeling Approach** We strongly believe that the ordinal regression should be preferred to the metric regression, because the invariance of the former allows to reduce the problem of hyper-parameters tuning. We also suppose that Ranking SVM definitely should be used as a baseline surrogate technique for comparison-based optimization algorithms, at least on unimodal optimization problems. For multi-modal and/or noisy problems the use of constraints seems to be too optimistic and additional selection of constraints (e.g., the most violated ones) should be considered. As an alternative to Ranking SVM, Gaussian Processes for ordinal regression can be used [Chu and Ghahramani, 2005a], however, our preliminary experiments showed that the surrogate learning is computationally more expensive and numerically unstable.

**Surrogate Model Control** Alternative approaches to the simple surrogate model based control used in this Chapter were discussed in Section 4.3.1.4. The lifelength-based control strategy is used because it can be parallelized on $\lambda$ CPUs and is simple to interpret. This strategy may be improved by using more sophisticated $\hat{n}(Err(\hat{f}))$ functions (dotted curves in Figure 4.12) instead of the linear one (bold curve). A more meaningful estimation of the model error can be obtained if use weights for different points in Eq. (4.9). We suppose that the two ideas described above may significantly improve the results. However, we intentionally avoided their further analysis to prevent the algorithm from a potential overfitting to the testbed and to keep the strategy in the simple and interpretable form.

**Surrogate Model Hyper-parameters Adaptation** In addition to already adapted hyper-parameters, we may also learn which Kernel function is the most suitable in the current state of the search [Tenne and Armfield, 2007]. Apart from the computational complexity, there is no reason not to use different surrogate modeling techniques with hyper-parameters adaptation for each technique independently, then
use the best model or an aggregation of the models for the exploitation phase. It is also suggested to use larger $\lambda_{hyp}$ if the computational of the model is relatively cheap.

The main limitations of the $^*$ACM-ES which should be addressed in further research:

**Space complexity** Ranking SVM with non-linear Kernels needs to store Kernel matrix of size $\ell^2$. The number of training points usually scales super-linearly with $n$ such that for $n > 100$ it is very likely that all required training points cannot be stored in memory. However, $^*$ACM-ES still can be used for say $n = 1000$ by setting upper bound for $\ell$, but this will lead to a smaller speedup (see Figure 4.9 for a similar effect for the original ACM). To reduce the space complexity, ensembles of surrogate models can be used, such that each surrogate model is responsible for $\ell' < \ell$ training points. In this case, an aggregation procedure of different surrogates should be used.

**Time complexity** The algorithm scales with $O(\ell^2)$. To drastically reduce the time complexity, one should not use non-linear Kernels. Recent results show that large-scale linear classification can be extremely fast (see LIBLINEAR [Fan et al., 2008]). We suppose that the use of ensembles of linear Ranking SVM is an attractive long term direction of research.

**Small Speedup on Noisy and/or Multi-modal Functions** The simple reason why the speedup may be relatively small on noisy and multi-modal functions is that these functions are difficult to approximate and usually more training points are required to build their accurate model. To better deal with this problem, one should closely look at the model learning on simplest 2-dimensional instances of the multi-modal and noisy problems at hand.

We suppose that the performance of $^*$ACM-ES on unimodal functions can be further improved by more carefully developing surrogate model control and adaptation procedures, taking inspiration from the prospective directions of research that have been discussed. On multi-modal functions, however, it seems more prospective to look at alternative restart strategies for CMA-ES which potentially may bring a comparable speedup at a lower computational cost. We will address this issue in detail in Section 6.3.3. Another interesting observation that we made during the development of $^*$ACM-ES is that if transformation (4.11) allows to decorrelate all the variables as well as possible such that the transformed space resembles the Sphere function, then why not use some very simple optimization algorithm which exploits the separability of this space? To answer this question we propose and investigate in Section 6.1 Adaptive Coordinate Descent method, a non-evolutionary and non-stochastic algorithm that has a comparable performance with CMA-ES on unimodal functions and may have linear time complexity.
Chapter 5

Multi-Objective Surrogate-Assisted CMA-ES

Multi-objective optimization problems are usually difficult to solve, because they inherit properties of individual objectives and a set of Pareto optimal solutions is the primal goal, rather than one single solution (see Section 2.4 for a review of multi-objective optimization). Mainstream surrogate approaches for multi-objective problems, with the notable exception of [Yun et al., 2004], often extend surrogate-based standard EAs, building one surrogate for each objective function and replacing the objective by its surrogate (see Section 3.3.3). The main limitation of such approaches is due to the approximation noise as the number of objectives increases. The learning cost indeed increases linearly with the number of objectives; but the Pareto dominance test, checking whether one individual is dominated by another one, requires comparing their surrogate values over all objectives; the error thus exponentially increases in the worst case with the number of objectives. Alternative single-model approaches based on the notion of expected improvement in multi-objective optimization in terms of hypervolume indicator have been proposed by [Emmerich et al., 2006a, Ponweiser et al., 2008] (see Section 3.3.3). These approaches, however, usually assume the evaluation of only one candidate solution per iteration that may lead to a loss of efficiently in the scenario with parallel evaluations.

Addressing the limitations described above, we propose a mono surrogate approach based on Aggregated Surrogate Models (ASM) aiming at building a global surrogate model in decision space, characterizing whether an individual belongs to i). the current Pareto set, or ii). the dominated region, or iii). the rest of the decision space (not yet visited, and containing the true Pareto set). This surrogate model, providing an aggregated perspective on all objective functions simultaneously, is then used to guide the search in the vicinity of the current Pareto set, and speed up the population move toward the true Pareto set. ASM is constructed by combining ideas from Regression and One-class SVMs and is described in detail in Section 5.1.

The formulation of ASM has been found to be over-constrained in Section 5.4.3.1, and, similarly to Chapter 4, we propose a relaxed version inspired from Ranking SVM, rank-based ASM model (RASM). RASM only requires to locally approximate the Pareto dominance relation, enabling to rank neighbor points within the objective space. Thanks to the flexibility of Ranking SVM, RASM is able to learn any kind of preference relations defined for solutions of multi-objective problem, e.g., based on Pareto dominance, Quality Indicator and Decision Maker preferences. The description of RASM is presented in Section 5.2.

As for surrogate model exploitation, we employ a simple pre-selection scheme in order
to investigate basic properties of ASM models (see Section 5.3). In Section 5.4, we experimentally validate both ASM and RASM models on a relatively simple two-objective benchmark testbed, demonstrating a significant reduction of the number of evaluations of the actual objective functions both for ASM-assisted NSGA-II and MO-CMA-ES algorithms. Finally, Section 5.5 concludes the Chapter and presents some perspectives for further research.

The results presented in this Chapter have been published as [Loshchilov et al., 2010a, Loshchilov et al., 2010b, Loshchilov et al., 2010d].

5.1 Aggregated Surrogate Model

In mono surrogate approach, the goal is to build an Aggregated Surrogate Model which aggregates the informations about preferences among training points in the objective space, usable to drive the population toward the true Pareto set. In this Section, a surrogate model \( \hat{f} \) of a multi-objective problem \( f \) is learned from i) points belonging to the current Pareto set, and ii) dominated points.

5.1.1 Rationale and Assumption

At any given time during the EMOA run, the relative position of the Pareto set and the dominated points can be schematically depicted as follows. The situation might be simple in the objective space (Figure 5.1-a), with the true Pareto front and the dominated region located on the two opposite sides of the current Pareto front. It can be much more intricate in the decision space; Figure 5.1-b illustrates the case where the true Pareto set (respectively, the dominated region) lies within (respectively, outside) the convex hull of the current Pareto set. Furthermore, the Pareto set can include many disjoint regions in the decision space. The assumption made is that the Pareto region includes a small number of connected components; note that this assumption holds for most classical multi-objective optimization benchmarks (e.g., IHR1, see Figure 5.5-c and -d).

While ASM expectedly discriminates the Pareto set and the dominated region, a binary classification approach is ill-suited, as it would not give any precise indication about where the true Pareto set is located. More generally, the Pareto set (true or current) and the dominated points cannot be handled in a symmetrical way: dominated points span over a subspace whereas the Pareto set should better be viewed as a manifold.

It thus comes to map all Pareto points onto a single value \( \rho \) (up to some tolerance \( \epsilon \)); meanwhile, the dominated points would be mapped onto the half space \( ]- \infty, \rho - \epsilon[ \). Such a mapping might actually provide useful indications: expectedly, points mapped onto the half space \( [\rho + \epsilon, + \infty[ \) would belong to the yet unexplored region, which is bound to contain the true Pareto set, and these points could thus be considered promising.

The above constraints on the ASM mapping can be expressed by combining the Support Vector Regression formulation ([Smola and Schölkopf, 2004], see Section 3.2.2), mapping each point \( \mathbf{x} \) onto some target value \( f(\mathbf{x}) \) up to some tolerance \( \epsilon \), and the One-class SVM ([Schölkopf et al., 2001], see Section 3.2.3), mapping a set of points onto a connected interval and thus characterizing the support of the underlying sample distribution. The
main difference is that the target value $\rho$ associated to the Pareto points is free in the ASM problem.

### 5.1.2 Lagrangian formulation

Let the training set be defined as $\mathcal{D} = \{x_1, \ldots, x_\ell, x_{\ell+1}, \ldots, x_m\}$ where the first $\ell$ points belong to the current Pareto front and the following points $x_{\ell+1}, \ldots, x_m$ are dominated ones. As discussed above, the ASM is obtained from the following learning constraints:

- All Pareto points $x_1, \ldots, x_\ell$ are mapped on some value $\rho$ up to tolerance $\epsilon$ (regression constraints);
- All dominated points are mapped onto $(-\infty, \rho - \epsilon]$ (one-class constraints).

Let $C$ and $\epsilon$ be two positive constants, and $\xi_{i}^{\text{up}}, \xi_{i}^{\text{low}}$ are slack variables measuring constraints violations, the primal ASM learning problem is:

\[
\text{Minimize } \frac{1}{2} \|w\|^2 + C \sum_{i=1}^{\ell} (\xi_{i}^{\text{up}} + \xi_{i}^{\text{low}}) + C \sum_{i=\ell+1}^{m} \xi_{i}^{\text{up}} + \rho \\
\text{subject to}
\]

\[
< w, \Phi(x_i) > \leq \rho + \epsilon + \xi_{i}^{\text{up}} \quad (i = 1 \ldots \ell) \\
< w, \Phi(x_i) > \geq \rho - \epsilon - \xi_{i}^{\text{low}} \quad (i = 1 \ldots \ell) \\
< w, \Phi(x_i) > \leq \rho - \epsilon + \xi_{i}^{\text{up}} \quad (i = \ell + 1 \ldots m) \\
\xi_{i}^{\text{up}} \geq 0 \quad (i = 1 \ldots \ell) \\
\xi_{i}^{\text{low}} \geq 0 \quad (i = 1 \ldots \ell) \\
\xi_{i}^{\text{up}} \geq 0 \quad (i = \ell + 1 \ldots m)
\]
The formulation of dual form of ASM learning problem and its resolution using SMO method is described in Appendix B.

In some cases the set of non-dominated points is too small to build accurate surrogate models or the preference relations between training points are only partially known, then it would be reasonable to directly learn preference relations as will be discussed in the next Section.

5.2 Rank-based Aggregate Surrogate Model

This Section gives an overview of the Rank-based Aggregate Surrogate Model (RASM), a Ranking SVM-based approach for learning preference information with a relatively small number of constraints.

5.2.1 Learning Preference Information in Multi-objective Space

To build a mono surrogate model \( \hat{f} \) of a multi-objective problem \( f \), one can use preference information defined on a set of training points \( \mathcal{D} = \{x_1, \ldots, x_\ell \} \) to satisfy \( \hat{f}(x_a) > \hat{f}(x_b) \) whenever \( x_a \) is preferred to \( x_b \). All preference pairs \((a, b)\) can be stored in a set of preference constraints \( \mathcal{P} \). Thus, one can formalize the learning of \( \hat{f} \) using Ranking SVM approach (see Sections 3.2.4) as follows:

\[
\begin{align*}
\text{Minimize} & \left\{ w, \xi \right\} \frac{1}{2} ||w||^2 + \sum_{i=1}^{\left|\mathcal{P}\right|} C_i \xi_i \\
\text{subject to} & \begin{cases} 
\langle w, x_a \rangle - \langle w, x_b \rangle \geq \epsilon_i - \xi_i, & i = 1, \ldots, \left|\mathcal{P}\right|; (a, b) \leftarrow \mathcal{P}_i \\
\xi_i \geq 0, & i = 1, \ldots, \left|\mathcal{P}\right|
\end{cases}
\end{align*}
\]  

(5.8)

(5.9)

The mentioned above problem can be solved using SMO method as described in Section 4.1.1. The crucial question is how to fill \( \mathcal{P} \) such that the final surrogate \( \hat{f} \) would be ideal for a given algorithm \( \mathcal{A} \). Any kind of preference information can be used to fill \( \mathcal{P} \), however, one should look at the following sources, relevant for the multi-objective context, to decide whether a preference pair \((a, b)\) should be added to \( \mathcal{P} \):

**Pareto Dominance** Check whether \( x_a \) dominates \( x_b \).

**Quality Indicator** Check whether \( I(x_a) \) is preferred to \( I(x_b) \) w.r.t. a given Quality Indicator \( I \), e.g., hypervolume contribution indicator (see Section 2.4.1).

**Decision Maker** Check whether the Decision Maker prefers \( x_a \) to \( x_b \).

When \( x_a \) is preferred to \( x_b \) \(((a, b) \) is the \( i \)-th preference constraint \( \mathcal{P}_i \)) we learn such \( \hat{f} \) that \( \hat{f}(x_a) - \hat{f}(x_b) = \langle w, x_a \rangle - \langle w, x_b \rangle \geq \epsilon_i - \xi_i \), where \( \epsilon_i \) is usually set to 1. Such definition corresponds to the preference constraint in its usual sense, referred to as ”>”-constraint. We also may define equality of \( x_a \) and \( x_b \), setting \( \epsilon_i \) to a relatively small value, say \( 10^{-3} \), and adding two pairs of constraints: i) \( \langle w, x_a \rangle - \langle w, x_b \rangle \geq 10^{-3} \), and ii).
\[ \langle w, x_b \rangle - \langle w, x_a \rangle \geq 10^{-3}. \] Thus, the learning algorithm will try to satisfy both constraints, mapping \( x_a \) and \( x_b \) to the same value of \( f \) with a small fluctuation in the order of \( 10^{-3} \). Such "=-"-constraint pairs can be especially useful to define equality of points belonging to the same non-dominated front.

It should be noted that for a training set of size \( \ell \) we may have at most \( n_{\text{alpha}} = \ell^2 - \ell \) constraints (\( \ell^2 - \ell \) if use only ">-"-constraints). Let us recall that the computational complexity of Ranking SVM scales with \( O(n_{\text{alpha}}^2) \), i.e., \( O(\ell^4) \) if all constraints are used. Moreover, from the the single-objective surrogate-assisted search, we know that \( \ell \) may scale with \( O(n^{1.7}) \) that would lead to \( O(n^7) \) complexity of learning full model. This is clearly an unsatisfactory computational complexity, therefore we should find a way to build reasonably good \( \hat{f} \) using much less constraints, preferably \( O(\ell) \) number of constraints.

### 5.2.2 Dominance-based Preference Learning

For the sake of tractability of RASM model learning we propose the following simple procedure (see, e.g., [Vapnik, 1979] for "chunking" procedure and [Joachims, 2005] for a recent review of the idea of selection of the most violated constraint):

1. Initialize a set of active constraints \( \Omega_{\text{active}} \) by primary constraints and a set of passive constraints \( \Omega_{\text{passive}} \) by secondary constraints.

2. Optimize Eq. (5.8) with \( P = \Omega_{\text{active}} \).

3. Remove the most violated constraint from \( \Omega_{\text{passive}} \) and add it to \( \Omega_{\text{active}} \). Optimize Eq. (5.8) with \( P = \Omega_{\text{active}} \). Repeat this step until stopping criterion is met.

Figure 5.2-Left illustrates a set of solutions in the objective space for which one should find a suitable set of constraints to learn \( \hat{f} \). For fast model learning the size of the set of constraints should be linear in number of solutions. Primary constraints are constraints which best describe the model and should be learnt in the first place. Secondary constraints are all other constraints which represent a source of the information about \( f \), but cannot be learnt all together due to limited computational resources. Let primary dominance constraints be associated to pairs \( (x_i, x_j) \) such that \( x_j \) is the nearest neighbor of \( x_i \) conditionally to the fact that \( x_i \) dominates \( x_j \) (continuous arrow, Figure 5.2-Left, Algorithm 5.1, lines 4-10), and let secondary dominance constraints be associated to pairs \( (x_i, x_j) \) such that \( x_i \) belongs to the current Pareto front and \( x_j \) belongs to another front from non-dominated sorting (dotted arrow, Figure 5.2-Left, Algorithm 5.1, lines 17-22).

After optimization of Eq. (5.8) for \( N_{\text{IterActive}} = 1000 \times |\Omega_{\text{active}}| \) iterations (Algorithm 5.2, line 3), the model \( \hat{f} \) may still may incorrectly predict some (test) preference relations from \( \Omega_{\text{passive}} \). We may identify the most violated constraint \( \text{constr}_{\text{mv}} \) from \( \Omega_{\text{passive}} \) (lines 6-13) and add it to \( \Omega_{\text{active}} \) (line 17) to continue learning the model for \( N_{\text{IterPerPassive}} = 10 \times |\Omega_{\text{active}}| \). After adding \( N_{\text{ConstrToAdd}} \) such constraints (typically 10% the number of primary constraints in the presented experiments, see Section 5.4), a robust \( \hat{f} \) can be obtained with relatively cheap learning.
5.3 ASM-assisted MOEAs with Pre-selection

This Section describes the Pareto-SVM algorithm, exploiting the ASM and RASM mono surrogates (ASMs) to speed up Evolutionary Multi-Objective Optimization. The terms ASM and RASM will be used interchangeably in this Section.

5.3.1 Discussion

As mentioned earlier, surrogate-assisted multi-objective optimization most commonly proceeds by replacing the objective function with the surrogate model, computing the true objective on carefully selected points, and retraining the model from time to time using recently evaluated individuals.

The situation here is different as the optimization problem is a multi-objective one, and the single ASM surrogate model is being built. The most natural idea, optimizing directly the ASM model, raises the following two issues. Firstly, the true Pareto set expectedly lies away from the dominated points and beyond the current Pareto set; the ASM would thus be used to explore yet unexplored regions, i.e., for extrapolation. Secondly and more importantly, identifying the Pareto set critically relies on the population diversity. While all individuals in the current Pareto set are equally mapped on the same \( \hat{f} \) value, some will be ‘more equal than others’, in the sense that they will get a higher ASM value by chance. Optimizing ex abrupto the ASM model would thus favor some regions of the Pareto set and hinder the population diversity.

For these reasons, the ASM model will be used to implement the pre-selection (aka informed operators) approach (see Section 3.3.1.4). Next Sections respectively outline the full algorithm, and describe the two specific modules of the Pareto-SVM algorithm, the surrogate model update and its use within informed operators.
Algorithm 5.1: Selection of Constraints for RASM model

1: given \((x_1, y_1), \ldots, (x_\ell, y_\ell), C\)
2: initialize \(\Omega_{\text{active}}, \Omega_{\text{passive}} \leftarrow \emptyset\)
3: \(Rank_{1,\ldots,\ell} \leftarrow \text{NonDominatedSorting}((x_1, y_1), \ldots, (x_\ell, y_\ell))\)
4: for \(i = 1, \ldots, \ell\) do
5: \(\Delta y_{\text{min}} \leftarrow \infty\)
6: for \(j = 1, \ldots, n\) do
7: if \((x_i \succ x_j) \text{ AND } (i \neq j)\) then
8: \(\Delta y \leftarrow \|y_i - y_j\|\)
9: if \(\Delta y < \Delta y_{\text{min}}\) then
10: \(\Delta y_{\text{min}} \leftarrow \Delta y; \ j_{\text{min}} \leftarrow j\) // nearest neighbor in objective space
11: end if
12: end if
13: end for
14: \(\text{constr.}\alpha \leftarrow \text{random} \in [0, C]; \ \text{constr.}\ a \leftarrow i; \ \text{constr.}\ b \leftarrow j_{\text{min}}; \ \text{constr.}\ \epsilon \leftarrow 1.0\)
15: \(\Omega_{\text{active}} \leftarrow \Omega_{\text{active}} \cup \{\text{constr}\}\)
16: end for
17: for \(i = 1, \ldots, \ell\) do
18: for \(j = 1, \ldots, \ell\) do
19: if \((Rank_i < Rank_j) \text{ AND } (Rank_i = 1)\) then
20: \(\text{constr.}\alpha \leftarrow \text{random} \in [0, C]; \ \text{constr.}\ a \leftarrow i; \ \text{constr.}\ b \leftarrow j; \ \text{constr.}\ \epsilon \leftarrow 1.0\)
21: if \(\text{constr} \notin \Omega_{\text{active}}\) then
22: \(\Omega_{\text{passive}} \leftarrow \Omega_{\text{passive}} \cup \{\text{constr}\}\)
23: end if
24: end if
25: end for
26: end for
27: return \(\Omega_{\text{active}}, \Omega_{\text{passive}}\)

5.3.2 The Algorithm

The general description of MOEA (Algorithm 5.3) is based on the usual parent-selection \(\rightarrow\) variation \(\rightarrow\) survival selection loop, with optionally some archive maintenance (line 5), as many popular MOEAs need to maintain some archive of the non-dominated individuals encountered during the search [Deb et al., 2000]. Note that line 4 describes both the parent selection and the application of the variation operators; it implicitly accounts for any choice procedure among multiple operators.

The Pareto-SVM algorithm is described likewise in Algorithm 5.4. The main differences are the model update (line 5) and the call to the informed operators (line 7) that replaces the standard call to variation operators, with the surrogate model \(\hat{f}\) as additional argument. The archive maintenance is limited to storing all newborn offspring (line 8). Actual update, including the ASM update, takes place every \(K_{\text{learn}}\) generations (line 4).
Algorithm 5.2: SVM Optimization

1: given $\Omega_{\text{active}}, \Omega_{\text{passive}}, N_{\text{IterActive}}, N_{\text{IterPerPassive}}, N_{\text{ConstrToAdd}}$
2: for $i = 1, \ldots, N_{\text{IterActive}}$ do
3: $\Omega_{\text{active}} \leftarrow \text{SVMOptimizationIteration}(\Omega_{\text{active}})$
4: end for
5: for $i = 1, \ldots, N_{\text{ConstrToAdd}}$ do
6: $\text{constr}_{\text{mv}} \leftarrow \Omega_{\text{passive}}$
7: $\xi_{\text{max}} \leftarrow 0$
8: for $j = 1, \ldots, |\Omega_{\text{passive}}|$ do
9: $\text{constr} \leftarrow \Omega_{\text{passive}}$
10: $\xi_j \leftarrow \hat{f}(x_{\text{constr},i}) + \epsilon_{\text{constr,}i} \cdot \hat{f}(x_{\text{constr},j})$
11: if $\xi_{\text{max}} > \xi_j$ then
12: $\xi_{\text{max}} \leftarrow \xi_j$
13: $\text{constr}_{\text{mv}} \leftarrow \text{constr}$ // find the most violated constraint
14: end if
15: end for
16: if $\xi_{\text{max}} > 0$ then
17: $\Omega_{\text{active}} \leftarrow \Omega_{\text{active}} \cup \{\text{constr}_{\text{mv}}\}$
18: $\Omega_{\text{passive}} \leftarrow \Omega_{\text{passive}} \setminus \{\text{constr}_{\text{mv}}\}$
19: for $j = 1, \ldots, N_{\text{IterPerPassive}}$ do
20: $\Omega_{\text{active}} \leftarrow \text{SVMOptimizationIteration}(\Omega_{\text{active}})$
21: end for
22: end if
23: end for
24: return $\Omega_{\text{active}}$

Algorithm 5.3: Standard MOEA

1: initialize $\text{Archive} \leftarrow \emptyset$
2: $\text{Pop} \leftarrow \text{MOEA.Init()}$
3: while NOT Stopping Criterion do
4: $\text{Offspring} \leftarrow \text{VarOp(ParentSelect(Pop))}$
5: $\text{UpdateArchive(Pop,Offspring)}$
6: $\text{Pop} \leftarrow \text{SurvivalSelect(Pop,Offspring)}$
7: end while
8: return $\text{Pop.BestIndividual}$

5.3.2.1 Model Update

The model update (Algorithm 5.5) starts from the current archive (as produced from the previous update) augmented by all newborn offspring (line 8 of Algorithm 5.4) and first removes the possible duplicates (line 1 of Algorithm 5.5). In most cases (depending on $K_{\text{learn}}$ and the number of offspring generated per generation), the size of the archive
Algorithm 5.4: Pareto-SVM

1: initialize Archive ← ∅
2: Pop ← MOEA.Init()
3: while NOT Stopping Criterion do
4:  if #generation ≡ 0 [Klearn] then
5:    \( \hat{f} = \text{UpdateModel}(\text{Archive}, \text{Pop}) \) // every \( K_{\text{learn}} \) generation
6:  end if
7:  Offspring ← InfOp(ParentSelect(Pop), \( \hat{f} \))
8:  Archive ← Archive \( \cup \) Offspring
9:  Pop ← SurvivalSelect(Pop, Offspring)
10: end while
11: return Pop.BestIndividual

Figure 5.3: A schematic view of the training data selection in objective space (Left) and SVM-informed selection of children from the pool of pre-children in decision space (Right).

increases far too much to make it possible to efficiently apply the Pareto-SVM learning. Furthermore, pruning the archive should not be done solely based on Pareto dominance, as in most standard MOEAs where only the best Pareto points are of interest. We need instead to ensure a good coverage of the dominated region that has been visited in the past, to make sure that the ASM will label these regions as 'dominated'. Borrowing ideas from PESA [Corne et al., 2001], the objective space is equally partitioned into \( N_{\text{archive}} \) boxes, and the archive keeps one point per box. Boxes are computed in lines 2 and 3, points are put in their respective boxes in line 5, and all boxes are pruned (line 8), keeping a uniformly chosen point among the non-dominated points of the box if any.

ASM is learned from a training set made of one point per box (line 10), plus the current population that is likely to contain non-dominated points (line 11). The training set is pruned to remove the duplicates and thereafter sorted using non-dominated sort to distinguish between current Pareto and dominated points (line 13). Finally it is passed to the ASM learning algorithm that returns the new ASM surrogate model to the main
Algorithm 5.5: UpdateModel(Archive, Pop)

1: EliminateDuplicates(Archive)
2: ComputeObjectiveBounds(Archive)
3: PartitionObjectiveSpace($N_{Archive}$)
4: for all $P \in$ Archive do
5:   FindBox($P$) // Assign $P$ to the box it belongs to
6: end for
7: for all Boxes $B$ do
8:   Ind[$B$] $\leftarrow$ Random(NonDominated($B$)) // Select one point per box
9: end for
10: Archive $\leftarrow \bigcup_B$ Ind[$B$] // at most $N_{Archive}$ points
11: TrainingData $\leftarrow$ Archive $\cup$ Pop
12: EliminateDuplicates(TrainingData)
13: NonDominatedSort(TrainingData)
14: return Pareto-SVM(Training Data) // returns $\hat{f}$

Algorithm 5.6: InfOp(Parents, $F$)

Require: OP(s) // variation operator(s)
1: Offspring $\leftarrow \emptyset$
2: for iOff = 1 to RequiredSize do
3:   Choose variation operator Op // Eventually
4:   GainBest $\leftarrow$ 0
5:   for i = 1 to $\lambda_{P_{re}}$ do
6:     Ind $\leftarrow$ Op(Parents)
7:     IndPop $\leftarrow$ NearestNeighbor(Ind, ND-Parents)
8:     Gain $\leftarrow$ $\hat{f}$(IndPop) - $\hat{f}$(Ind)
9:     if Gain > GainBest then
10:        GainBest $\leftarrow$ Gain
11:        Best $\leftarrow$ Ind
12:    end if
13: end for
14: Offspring $\leftarrow$ Offspring $\cup$ {Best}
15: end for
16: return Offspring

algorithm (line 14).

5.3.2.2 Informed Operators

The Pareto-SVM algorithm uses the ASM to yield informed operators [Rasheed and Hirsh, 2000]. Upon calling a variation operator, $\lambda_{P_{re}}$ pre-children are generated and ordered according to their quality estimate on $\hat{f}$; a probabilistic selection attached
5.4 Experimental Validation

Figure 5.4: **Left**: Learning time of the proposed dominance-based RASM on ZDT1 function. **Right**: Mapping the ranks of pre-children to a normal distribution.

to the offspring indices is achieved, using a normal distribution with variance $\sigma_{sel}^2$ (Figure 5.4-**Right**). Granted that the number of offspring is sufficiently large, parameter $\sigma_{sel}^2$ thus controls the selection pressure and the exploration vs exploitation trade-off. However we use at the moment a small value $\sigma_{sel}^2 = 0.001$ for the normal distribution for the ranked point to be chosen (Figure 5.4-**Right**) to simulate the situation when new offspring always has the first rank.

An additional difficulty is raised in the multi-objective context, as a better surrogate value does not imply a smaller distance from the Pareto set. Quite the contrary, a child that is far from its parent can have a better ASM value than its parent while being nevertheless farthest from the Pareto front than some other points, because of the errors in the surrogate model, and/or the $\epsilon$ tolerance in the ASM and RASM formulations. In order to handle this issue, confirmed from preliminary experiments, the pre-children filtering is based on their ASM gain with respect to the closest point in the current Pareto set.

Formally, Algorithm 5.6 describes how all offspring are generated from the current parent population. For each offspring to be generated (outer loop, lines 2 to 15), a variation operator is chosen (line 3) if more than one are available (depending on the type of MOEA) and applied $\lambda_{Pre}$ times (line 6). To each pre-child thus obtained, is associated its nearest neighbor among current non-dominated parents (line 7), and the ASM improvement of the pre-child compared to its nearest neighbor determines whether the pre-child is kept (line 9). An example for $\lambda_{Pre} = 3$ is shown in Figure 5.3-**Right**.
5.4 Experimental Validation

5.4.1 Experimental Setting

Two state-of-the-art EMOA algorithms are considered: (100+100)-NSGA-II (see Section 2.4.3) and 100 × (1 + 1)-MO-CMA-ES (see Section 2.4.4). Both algorithms use the hypervolume indicator as second-level sorting criterion to rank individuals on the same level of non-dominance.

The Pareto-SVM approach is assessed by comparing the original algorithms with their ASM and RASM enhanced versions, considering the widely used ZDT1:3-6 [Zitzler et al., 2000] and their rotated variants IHR1:3-6 [Igel et al., 2007b] benchmark problems. The dimension is set to 30 (resp. to 10) for ZDT1-3 problems (resp. for all other problems). As the true Pareto front of all ZDT problems lies on the boundary of the decision space, and for the sake of an unbiased assessment (to prevent MO-CMA-ES from exploiting this specificity), the penalization term is set to $\alpha = 1$ instead of the original $10^{-6}$ [Igel et al., 2007b] (see Section 6.2.2.2 for a discussion).

Both ASM and RASM are based on the RBF kernel, where the bandwidth $\sigma$ is set as the average distance between all pairs of training points. The SVM penalization constant $C$ is set to 1000 after few preliminary experiments.

The training set $\mathcal{D}$ that is used at each generation to build the RASM model is an archive that contains at most $N_{\text{archive}} = 1000$ points. The current population is added to the archive at each generation. When this archive gets larger than $N_{\text{archive}}$, it is pruned by removing the worst individuals after non-dominated sorting. Furthermore, in order to improve the diversity of the training set (many points too close together can lead to poor surrogate model), an additional filtering procedure is applied to the archive. The 2-objective space has been divided into $100 \times 100$ boxes, and at most one point among the archived non-dominated points of each box is retained in the archive (see Figure 5.3).

The parameters of ASM models have been calibrated using a few preliminary experiments (their automatic tuning will be considered in further work):

**ASM** The allowed deviation $\epsilon$ of non-dominated points on $\hat{f}$ is set to $10^{-5}$. The ASM learning was stopped after 300,000 iterations, corresponding to circa 0.5 – 1.0 sec. on a 2.26 GHz processor for ZDT1.

**RASM** As detailed in Section 5.2, RASM maintains the set $\Omega_{\text{active}}$ of active constraints, initialized to the set of primary dominance constraints. After an initial round of $N_{\text{IterActive}} = 1000|\Omega_{\text{active}}|$ iterations, $\Omega_{\text{active}}$ is incrementally enriched every $N_{\text{IterPerPassive}} = 10|\Omega_{\text{active}}|$ iterations with the most violated constraint among the secondary dominance constraints, until a total of at most $N_{\text{ConstrToAdd}} = 0.1|\Omega_{\text{active}}|$ secondary the most violated constraints have been added.

The $\hat{f}$ update frequency $K_{\text{learn}}$ is set to 10. The ASM-enhanced operators were computed as described in Algorithm 5.4. In the MO-CMA-ES case, in order to introduce an additional diversity, the global mutation step size of pre-selected offspring was modified to $\sigma' = \sigma \exp(-d+2dk)$ where $d = 0.7$ and $k$ is uniformly distributed in $[0,1]$. 

142
5.4 Experimental Validation

5.4.2 Performance Measures

Many ways of measuring the performance of EMO algorithms have been proposed. After [Knowles et al., 2006], this study uses Pareto-compliant quality indicators, more particularly the widely used hypervolume indicator $I_H$. Let $P$ be a $\mu$-size approximation of Pareto front and let $P^*$ be the approximate $\mu$-optimal distribution of optimal Pareto points [Auger et al., 2009]. Several algorithms proposed in [Auger et al., 2009] can be used for approximation of $\mu$-optimal distribution. The approximation error of the Pareto front is defined by $\Delta H(P^*, P) = I_H(P^*) - I_H(P)$. All reported results are averaged over 10 independent runs with at most 100,000 fitness evaluations.

In the case of IIHR problems, the arbitrary rotation matrix $O \in \mathbb{R}^{n \times n}$ must be fixed.

5.4.3 Model Validation

The goal of the first experiments is to empirically evaluate the accuracy of ASM and RASM models.

5.4.3.1 ASM model

In order to evaluate its accuracy on ZDT1 and IIHR1 problems, the ASM model was trained using calibrated training data set $A$: 20000 points were generated at a given distance from the (known) nearly-optimal Pareto points, and non-dominated sorting was applied to rank those points. Front $ndom_1(A)$ denotes the closest front from the true Pareto front, $ndom_2(A)$ denotes the second one and so forth.

Figure 5.5 illustrates the distribution of $\hat{f}$ values for training and test data in decision and objective spaces, where the training set respectively includes $ndom_{80}(A)$ and $ndom_{100}(A)$ as non-dominated and dominated points. As shown in Figure 5.5, $\hat{f}$ correctly approximates the Pareto-dominance in the sense that for all $k$, $\hat{f}(ndom_k(A)) > \hat{f}(ndom_{k+20}(A))$ on average.

As seen for the IIHR1 problem, although the $\hat{f}$ value usually lies in an $\epsilon$-width tube for training Pareto points, the new Pareto front may be non-linear. This behavior is quite normal when we deal with difficult problems. It may lead to premature convergence if we use very selective $f$-based filter ($\lambda_{Pre}$ is large), as high $f$-based selection pressure may accelerate the exploration of the prospective regions of Pareto front and entail some loss of diversity. Note that, as PARETO-SVM was devised to speed up the EMOA convergence and does not specifically take into account the Pareto diversity, it may be inefficient with regard to the approximation of the $\mu$-optimal distribution of nearly-optimal Pareto points.

As mentioned in the introduction, the ASM problem is over-constrained as all Pareto points must be mapped on a narrow interval $[\rho - \epsilon, \rho + \epsilon]$; in such cases, it results in a poor generalization error of the ASM (visible, e.g., from its errors on the rest of the Pareto archive). This problem was fixed using an additional $k$ factor, replacing $\rho$ by $k\rho$ in Eq. (5.1). The best $k$ value w.r.t. the ASM generalization error was determined in original algorithm from a preliminary trial, leading to $k = 1$ for one set of problems and $k = -1$.
Figure 5.5: ASM model for 30-dimensional ZDT1 and IHR1 problems. Non-dominated fronts \( ndom_{80}(A) \) (white circles) \( \prec ndom_{100}(A) \) (green circles) form the training data, while \( ndom_k(A) \) (blue circles) for \( k < 80 \) represent the test data. See text for details.

for another set. Probably, a better solution is to reformulate Eq. (5.1) as follows:

\[
\text{Minimize } \frac{1}{2} \| w \|^2 + C \sum_{i=1}^{\ell} (\xi_{up}^i + \xi_{low}^i) + C \sum_{i=\ell+1}^{m} \xi_{up}^i + \frac{1}{2} D \rho^2
\]

In the new formulation, \( D \) is an additional problem-dependent parameter, which in some sense makes the learning problem harder. On-going work aims at understanding this phenomenon +/-\( \rho \), and relating it to the structure of the multi-objective landscape.
5.4 Experimental Validation

Figure 5.6: **Left:** at each iteration of RASM model learning procedure the most violated secondary constraint is added and model quality $\hat{f}$ is estimated on 1000 test points. **Right:** normalized Pareto rank predicted by RASM model for 1000 test points on 30-dimensional ZDT1 functions.

### 5.4.3.2 RASM model

Some experiments are first conducted to estimate the complexity of RASM model training on 30-dimensional ZDT1 problem. The empirical complexity with respect to the number of training points is circa 2.2 (slope in Figure 5.4-Left in log scale). The fact that the complexity is super-quadratic is not surprising since the computational complexity of SMO optimization procedure is quadratic in number of constraints and the number of constraints is up to 10% larger than the number of training points. The complexity however remains bounded as the size of the training set (extracted from the archive) is less than 1,000, limiting de facto the computational cost of the RASM learning.

Figure 5.6-Left shows RASM model learning for 30-dimensional ZDT1 problem using 500 training points, uniformly randomly generated in $[0, 1]^{30}$ search space. We sorted all training points using non-dominated sorting procedure and filled the sets of primary and secondary constraints as described in Section 5.2.2. After the initial learning of RASM model with primary constraints, we iteratively add the most violated secondary constraint and continue to learn the model. Figure 5.6-Left shows that the model error estimated at each iteration on 1000 (hidden from the learning part) test points decreases. The learning stops when the most violated constraint does not violate preference relations anymore. In the following we limit the number of the most violated constraints to be added by 10% of the number of primary constraints (e.g., would be 50 the most violated constraint for this example). Figure 5.6-Right shows normalized Pareto Rank, predicted by RASM model. The prediction, indeed, is not perfect but seems to be suitable to direct the search toward the Pareto front.
5.4.4 Benchmarking of ASM and RASM assisted MOEAs

The second set of experiments investigates the effect of using ASM and RASM model within existing MOEAs on different benchmark functions.

The first experiments with ASM and RASM-based MOEAs show that PARETO-SVM indeed speeds up both S-NSGA-II and MO-CMA-ES on most problems. Figure 5.7 shows the on-line behavior of the algorithms for ZDT1 and IHR1 with ASM models (the results for RASM are quite similar).

For the ZDT1 problem, the optimal Pareto front is linear and lies on the boundary of the decision space. Therefore, dominated points often lie at the decision space center, while Pareto points go toward the boundary, making the ASM model fairly simple: the One-Class SVM for dominated points covers the internal region of the decision space, while a small subspace of the Pareto points is covered by SVM-Regression with a given $\epsilon$ value.

ASM-based S-NSGA-II (ASM-NSGA) works nearly 1.5 times faster with $\lambda_{Pre} = 2$ and more than 2 times faster with $\lambda_{Pre} = 10$ than the original version with regard to the $\Delta H$ value and the number of function evaluations. The value $\Delta H = 0.001$ for ZDT problems corresponds to the situation when all points are non-dominated.

The IHR problems, rotated variants of ZDT problems, are non-separable and thus significantly more difficult for the MOEAs with variation operators which use separability. The Pareto set of IHR1 for a given rotation matrix is shown in Figure 5.5-c. The MO-CMA-ES inherits invariance properties from the CMA-ES (though the hypervolume indicator is not invariant w.r.t. rank-preserving transformations of objectives, the invariance can be achieved by an approach described in Section 6.4.3), therefore it is also efficient on these rotated problems, while S-NSGA-II can approximate only a small part of optimal Pareto front which corresponds to the center of decision space.

The variance of results on ZDT1 problem is small because this problem is very simple for surrogate modeling and even if some premature convergence initially leads to sample only a small part of the Pareto set, the algorithm quickly explores the rest of the set thanks to separability. On rotated IHR1 problem, such quick moving is difficult, hence the higher variance of results which corresponds to slowly moving along the Pareto front. A high selection pressure also accelerates this effect.

Both MO-CMA-ES and S-NSGA-II only approximate a small part of the Pareto front of IHR1 in first generations, but in contrast to S-NSGA-II, MO-CMA-ES can gradually approximate the whole front. This can be seen clearly in Figure 5.7-b, witnessed by the flat line between 10000 and 40000 evaluations. In this case, while the ASM model helps MO-CMA-ES to converge faster to the Pareto front, it can not give any preference to the extreme points which in fact help to move along the Pareto front.

This observation sustains the idea that quality indicators should probably be taken into account during the ASM learning. The hypervolume indicator may provide useful additional information, especially because extreme points usually have the highest importance. Also, hypervolume or epsilon indicators are well suited for many-objective optimization, when most points are non-dominated.

Table 5.1 shows the comparative results of all baselines, ASM and RASM-based EMOs; in the latter cases, both $\lambda_{Pre} = 2$ and $\lambda_{Pre} = 10$ pre-offspring are considered. Different
5.4 Experimental Validation

Figure 5.7: On-line performances of original and ASM-informed S-NSGA-II on ZDT1 (left) and MO-CMA-ES on IHR1 (right) problems with different values of number of pre-children $p = \lambda_{Pre}$. Error bars indicate the 20% and 80% percentiles (almost indistinguishable for ZDT1).

target values for $\Delta H$ have been set, and the number of evaluation needed to reach those values are reported - normalized by the smallest value of the table (recalled on the top row, legend “Best”). Hence 1 indicates the best result, while e.g., 2 indicates that this algorithm needed twice the number of evaluations of the best algorithm to reach the target $\Delta H$ value. These results first confirm that S-NSGA-II performs best on separable functions ZDTx and MO-CMA-ES on non-separable functions IHRx. They also show that both RASM-NSGA and RASM-MO-CMA work nearly 1.5 times faster with $\lambda_{Pre} = 2$ and more than 2 times faster with $\lambda_{Pre} = 10$ than the baseline versions with regards to the $\Delta H$ value and the number of function evaluations.

ASM-NSGA and RASM-NSGA yield comparable performances. A more thorough analysis shows that RASM-MO-CMA is usually faster at the beginning (up to 10000-15000 function evaluations) though it might suffer from a premature convergence thereafter: experiments on concave IHR2 (and to some extent, also on ZDT2) show that RASM-MO-CMA converges to the value $\Delta H = 0.1$ nearly 1.5 times faster than with ASM model, and fails to go further. This failure is blamed on the fact that the diversity of the population is hardly preserved; a small part of the optimal Pareto front is sampled. A general trend is observed, that increasing the selection pressure leads to a faster convergence. However, increasing the number of pre-children can also lead to premature convergence, like for MO-CMA-ES on IHR problems with $\lambda_{Pre} = 10$. This happens because the filter prefers the points which are possibly better than their parents according to $f$ though they might be farther from the true Pareto front than other parents. The comparison of the pre-children with the closest parent in decision space (Algorithm 5.6, line 7) addresses this drawback to some extent. Indeed, RASM learning and the RASM-based offspring selection only aim at speeding up the convergence; further work will be required to extend the approach and...
approximate the $\mu$-optimal distribution of nearly-optimal Pareto points.

Table 5.1: Comparative results of two baseline EMOAs, namely S-NSGA-II and MO-CMA-ES and their ASM and RASM variants. Median number of function evaluations (out of 10 independent runs) to reach $\Delta H_{\text{target}}$ values, normalized by Best: a value of 1 indicates the best result, a value $X > 1$ indicates that the corresponding algorithm needed $X$ times more evaluations than the best to reach the same precision.

<table>
<thead>
<tr>
<th>$\Delta H_{\text{target}}$</th>
<th>1</th>
<th>0.1</th>
<th>0.01</th>
<th>0e-3</th>
<th>1e-4</th>
<th>1</th>
<th>0.1</th>
<th>0.01</th>
<th>0e-3</th>
<th>1e-4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ZDT1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ZDT2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best</td>
<td>1100</td>
<td>3000</td>
<td>5300</td>
<td>7800</td>
<td>38800</td>
<td>1400</td>
<td>4200</td>
<td>6600</td>
<td>8500</td>
<td>32700</td>
</tr>
<tr>
<td>S-NSGA-II</td>
<td>1.6</td>
<td>2</td>
<td>2</td>
<td>2.3</td>
<td>1.1</td>
<td>1.8</td>
<td>1.7</td>
<td>1.8</td>
<td>2.3</td>
<td>1.2</td>
</tr>
<tr>
<td>ASM-NSGA $\lambda_{p_{re}}=2$</td>
<td>1.2</td>
<td>1.5</td>
<td>1.4</td>
<td>1.5</td>
<td>1.6</td>
<td>1.2</td>
<td>1.2</td>
<td>1.4</td>
<td>1.4</td>
<td>1</td>
</tr>
<tr>
<td>ASM-NSGA $\lambda_{p_{re}}=10$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RASM-NSGA $\lambda_{p_{re}}=2$</td>
<td>1.2</td>
<td>1.4</td>
<td>1.4</td>
<td>1.5</td>
<td>1</td>
<td>1.3</td>
<td>1.2</td>
<td>1.2</td>
<td>1.5</td>
<td>1</td>
</tr>
<tr>
<td>RASM-NSGA $\lambda_{p_{re}}=10$</td>
<td>1</td>
<td>1.1</td>
<td>1.1</td>
<td>1.5</td>
<td>1</td>
<td>1.1</td>
<td>1</td>
<td>1.2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>MO-CMA-ES</td>
<td>16.5</td>
<td>14.4</td>
<td>12.3</td>
<td>11.3</td>
<td></td>
<td>14.7</td>
<td>10.7</td>
<td>10</td>
<td>10.1</td>
<td></td>
</tr>
<tr>
<td>ASM-MO-CMA $\lambda_{p_{re}}=2$</td>
<td>6.8</td>
<td>8.5</td>
<td>8.3</td>
<td>8</td>
<td></td>
<td>5.9</td>
<td>8.2</td>
<td>7.7</td>
<td>7.5</td>
<td></td>
</tr>
<tr>
<td>ASM-MO-CMA $\lambda_{p_{re}}=10$</td>
<td>6.9</td>
<td>10.1</td>
<td>10.4</td>
<td>12.1</td>
<td></td>
<td>5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-MO-CMA $\lambda_{p_{re}}=2$</td>
<td>5.1</td>
<td>7.7</td>
<td>7.6</td>
<td>7.4</td>
<td></td>
<td>5.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-MO-CMA $\lambda_{p_{re}}=10$</td>
<td>3.6</td>
<td>4.3</td>
<td>4.9</td>
<td>7.2</td>
<td></td>
<td>3.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>ZDT3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>ZDT6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best</td>
<td>1300</td>
<td>3500</td>
<td>7100</td>
<td>10100</td>
<td>15200</td>
<td>2500</td>
<td>3600</td>
<td>5200</td>
<td>12300</td>
<td></td>
</tr>
<tr>
<td>S-NSGA-II</td>
<td>1.4</td>
<td>1.9</td>
<td>1.6</td>
<td>1.9</td>
<td>2.2</td>
<td>2.1</td>
<td>3.4</td>
<td>3.8</td>
<td>2.7</td>
<td></td>
</tr>
<tr>
<td>ASM-NSGA $\lambda_{p_{re}}=2$</td>
<td>1.1</td>
<td>1.3</td>
<td>1.1</td>
<td>1.2</td>
<td>1.3</td>
<td>1.4</td>
<td>2.4</td>
<td>2.6</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ASM-NSGA $\lambda_{p_{re}}=10$</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1.8</td>
<td>2.3</td>
<td>2.3</td>
<td></td>
</tr>
<tr>
<td>RASM-NSGA $\lambda_{p_{re}}=2$</td>
<td>1.1</td>
<td>1.3</td>
<td>1.2</td>
<td>1.4</td>
<td>1.6</td>
<td>1.5</td>
<td>2.4</td>
<td>2.8</td>
<td>2.1</td>
<td></td>
</tr>
<tr>
<td>RASM-NSGA $\lambda_{p_{re}}=10$</td>
<td>1</td>
<td>1.1</td>
<td>1.1</td>
<td>2</td>
<td></td>
<td>1.4</td>
<td>2</td>
<td>2.3</td>
<td>1.8</td>
<td></td>
</tr>
<tr>
<td>MO-CMA-ES</td>
<td>15.4</td>
<td>17.8</td>
<td></td>
<td></td>
<td></td>
<td>2.5</td>
<td>2.6</td>
<td>2.5</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>ASM-MO-CMA $\lambda_{p_{re}}=2$</td>
<td>9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.1</td>
<td>1.2</td>
<td>1.1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ASM-MO-CMA $\lambda_{p_{re}}=10$</td>
<td>8</td>
<td>25.6</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1.1</td>
<td>1.3</td>
<td>2.5</td>
<td></td>
</tr>
<tr>
<td>RASM-MO-CMA $\lambda_{p_{re}}=2$</td>
<td>8.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.5</td>
<td>1.2</td>
<td>1.2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>RASM-MO-CMA $\lambda_{p_{re}}=10$</td>
<td>8.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>1.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IHR1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IHR2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best</td>
<td>500</td>
<td>2000</td>
<td>35300</td>
<td>41200</td>
<td>50300</td>
<td>1700</td>
<td>7000</td>
<td>12900</td>
<td>52900</td>
<td></td>
</tr>
<tr>
<td>S-NSGA-II</td>
<td>1.6</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td>1.1</td>
<td>3.2</td>
<td>6.2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASM-NSGA $\lambda_{p_{re}}=2$</td>
<td>1.2</td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td>3.9</td>
<td>4.9</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASM-NSGA $\lambda_{p_{re}}=10$</td>
<td>1</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td>1.4</td>
<td>6.4</td>
<td>4.6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-NSGA $\lambda_{p_{re}}=2$</td>
<td>1.2</td>
<td>1.2</td>
<td></td>
<td></td>
<td></td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-NSGA $\lambda_{p_{re}}=10$</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td>1.2</td>
<td>5.1</td>
<td>4.8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO-CMA-ES</td>
<td>8.2</td>
<td>6.5</td>
<td>1.1</td>
<td>1.2</td>
<td>1.2</td>
<td>5.8</td>
<td>2.7</td>
<td>2.1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>ASM-MO-CMA $\lambda_{p_{re}}=2$</td>
<td>4.6</td>
<td>2.9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3.1</td>
<td>1.6</td>
<td>1.4</td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td>ASM-MO-CMA $\lambda_{p_{re}}=10$</td>
<td>9.2</td>
<td>6.1</td>
<td>1.3</td>
<td>1.2</td>
<td></td>
<td>5.9</td>
<td>2.6</td>
<td>2.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-MO-CMA $\lambda_{p_{re}}=2$</td>
<td>2.6</td>
<td>2.3</td>
<td>2.4</td>
<td>2.1</td>
<td></td>
<td>2.2</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-MO-CMA $\lambda_{p_{re}}=10$</td>
<td>1.8</td>
<td>1.9</td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>IHR3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>IHR6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Best</td>
<td>800</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>16500</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>S-NSGA-II</td>
<td>1.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASM-NSGA $\lambda_{p_{re}}=2$</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASM-NSGA $\lambda_{p_{re}}=10$</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-NSGA $\lambda_{p_{re}}=2$</td>
<td>1.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-NSGA $\lambda_{p_{re}}=10$</td>
<td>1.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MO-CMA-ES</td>
<td>9.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASM-MO-CMA $\lambda_{p_{re}}=2$</td>
<td>7.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ASM-MO-CMA $\lambda_{p_{re}}=10$</td>
<td>12.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>-</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-MO-CMA $\lambda_{p_{re}}=2$</td>
<td>3.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RASM-MO-CMA $\lambda_{p_{re}}=10$</td>
<td>2.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
5.5 Discussion

In this Chapter, we investigated two Aggregated Surrogate Models, ASM and RASM, for surrogate-assisted multi-objective optimization with pre-selection of promising offspring.

The original ASM model is built by combining One-class and regression SVMs; thanks to the non-linear kernel, ASM can be learned efficiently in non-linear functional spaces. ASM learning problem, however, was found to be over-constrained; in such cases, it results in a poor generalization error. In contrast, using a Learning to Rank framework in RASM, the resulting surrogate model does not require that all Pareto points are mapped onto the same value. It is thus both more constrained in the dominated region, and less constrained on the Pareto front, than ASM.

Furthermore, RASM approach opens new and interesting perspectives for real world multi-objective problems, enabling for instance to account for the user’s preferences in a flexible way by simply adding user-defined constraints to the order-based SVM formulation. Most importantly, the rank constraint formalization enables to accommodate conflicting preferences: to the best of our knowledge, this corresponds to a significant advance on the state of the art.

The experimental validation of the proposed approach shows that RASM-EMO usually converges faster than ASM-EMO, with the caveat that it sometimes leads to premature convergence (e.g., on ZDT2 and IHR2 problems). This premature convergence was blamed on the selection pressure and the adjustment of parameter $\sigma^2_{sel}$. A further work will explore the adjustment of $\sigma^2_{sel}$ and $\lambda_{pre}$ depending on the model error $\hat{f}$ in a similar way to $\hat{n}$ in the case of single-objective surrogate-assisted optimization (see Section 4.3.1.3).

It is emphasized that RASM might incorporate additional specific constraints in each generation. Some possible constraints are described in Figure 5.2-Right: such non-dominance constraints involved points on the current Pareto front, and include inequality constraints from the extremal points over their neighbors (continuous arrows), and equality constraints for all neighbor pairs on the Pareto front (continuous double arrow), as well as between extremal points (dotted double arrows). Along the same lines, constraints could be weighted, e.g., the weight of constraints related to points with the largest hypervolume contributions can be increased online. We already tested several strategies for primary and secondary constraints selection and found that the following simple strategy for primary constraints selection often shows better results: select a pair $(a,b)$ with probability $p$ (e.g., 10%) if $x_b$ is located in the following Pareto front to the front of $x_a$. The main drawback of this strategy is its computational complexity which is growing faster than for the proposed in this Chapter approach due to usually larger number of constraints needed to learn the model. However, the strategy seems to be prospective and should be further investigated.

Historically, most of the work presented in this Chapter was done in 2009 and 2010, before the introduction of surrogate model control and hyper-parameters adaptation for single-objective surrogate-assisted optimization in Chapter 4. Thus, the extension of these approaches to the multi-objective case will be in the focus of future research. Preliminary results for direct optimization of SVR-based surrogate models (one model per objective)
and an analysis of offline tuning of $\hat{n}$ are presented in [Yagoubi, 2012], where NSGA-II surrogate-assisted algorithm was successfully applied for real-world multi-objective optimization of Diesel combustion. [Pilát and Neruda, 2011] proposed Multiobjective Memetic Algorithm with Aggregate Surrogate Model (ASM-MOMA), where ASM model is used for direct local optimization and the best found individual during the optimization is placed back to the population. The authors combine direct optimization with pre-selection in [Pilát and Neruda, 2012] and suggest to use Evolution Strategy for ASM model optimization. The speedup between a factor of 2 and 10 is reported for optimization of ZDT problems. Thus, the question whether individual surrogates per objective or ASM models are better for surrogate-assisted search remains open. The answer on this question may become clearer when self-adaptation mechanisms investigated in Chapter 4 will be translated to the multi-objective case and assessed on a rich set of benchmark problems, organized in a kind of BBOB for multi-objective optimization.
Chapter 6
Exploring new frontiers using CMA-like algorithms

In this Chapter, we explore and expand frontiers of the Evolutionary Computation domain, and CMA-ES in particular, motivated by pure curiosity and answering questions, which have arisen in the exploitation part of the thesis in Chapters 4, 5. The discussed questions and corresponding approaches are presented in chronological order.

In Chapter 3 (see Section 4.2.1), we showed that the use of the decorrelated coordinate system, provided by the covariance matrix of CMA-ES, may drastically improve the results of surrogate-assisted CMA-ES on non-separable and ill-conditioned functions. Given the fact that CMA-ES can be viewed as CSA-ES performing in this decorrelated space [Hansen, 2008], we wondered whether there is a simpler algorithm which is competitive to CSA-ES in this space? As candidate algorithm we take one of the simplest derivative-free optimizer, Coordinate Descent method, and combine it with the Adaptive Encoding procedure proposed in [Hansen, 2008] to design Adaptive Coordinate Descent (ACiD), a method, which has a competitive performance to CMA-ES on unimodal functions and, to the best of our knowledge, is the only linear time algorithm with a reasonably good performance on non-separable ill-conditioned problems. This algorithm is investigated in Section 6.1. Some of the results have been published in [Loshchilov et al., 2011a].

In Chapter 5, we observed that the multi-objective optimization may significantly slow down when only few individuals of the population are able to improve the current Pareto front approximation. This is the case for IHR1 problem (see Figure 5.7), where a sub-part of the Pareto front can be relatively quickly found, while the rest of the front is approximated slowly and substantially thanks to the extreme (right) point in the objective space which generates new (hopefully) extreme point. We wondered whether there is a way to improve the search allowing the most successful individuals to generate offspring more often? To answer this question we introduce reward-based parent selection schemes for multi-objective optimization in Section 6.2 and demonstrate that a significant speedup can be obtained by carefully looking at the dynamic of successful individuals. Some of the results have been published in [Loshchilov et al., 2011b].

In our recent experiments with *aACM-ES in Chapter 4 we found that the speedup obtained on multi-modal functions is much less significant than on unimodal ones. This is mainly due to the difficulty to learn the landscapes of these functions, which typically require a very large number of training points. The latter requirement is usually difficult to satisfy because i). we simply do not have enough training points around the actual region of search, and ii). the computational complexity of learning grows quickly (usually quadratically) with the number of training points, given that the runtime is usually very
large on multi-modal functions. In this case, rather than build computationally expensive surrogate models, it might be reasonable to look at alternative restart strategies for CMA-ES to improve the search without any increase of the computational cost. We address this idea in Section 6.3, where we propose two new alternatives to the original IPOP and BIPOP restart strategies, that have also been published in [Loshchilov et al., 2012a, Loshchilov et al., 2012d].

In Section 6.4, we briefly discuss other prospective approaches that we investigated during the thesis. Finally, Section 6.5 concludes the Chapter and presents some perspectives for further research.

6.1 Adaptive Coordinate Descent

Independence from the coordinate system is one source of efficiency and robustness for the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). The recently proposed Adaptive Encoding (AE) [Hansen, 2008] procedure generalizes CMA-ES adaptive mechanism, and can be used together with any optimization algorithm. Adaptive Encoding gradually builds a transformation of the coordinate system such that the new coordinates are as decorrelated as possible with respect to the objective function. But any optimization algorithm can then be used together with Adaptive Encoding, and this Section proposes to use one of the simplest of all, that uses a dichotomy procedure on each coordinate in turn. The resulting algorithm, termed Adaptive Coordinate Descent (ACiD), is analyzed on the Sphere function, and experimentally validated on BBOB testbed where it is shown to outperform the standard (1 + 1)-CMA-ES, and is found comparable to other state-of-the-art CMA-ES variants on unimodal functions.

In this Section, we present an extended version of ACiD, originally published in [Loshchilov et al., 2011a], with a more detailed discussion of i). Adaptive Encoding parametrization in Section 6.1.3.1; ii). extension to noisy optimization in Section 6.1.3.5; iii). extension to large-scale optimization in Section 6.1.3.6.

6.1.1 Introduction

Separable continuous optimization problems are problems in which the objective function can be optimized coordinate-wise. Finding the global optimum of a separable function in $\mathbb{R}^n$ amounts to perform $n$ simple line searches along each of the $n$ coordinates. Unfortunately, interesting problems are usually not separable. Nevertheless, many optimization methods implicitly assume some form of separability of the objective function, or at least are much more efficient on separable functions as they explicitly use the coordinate system in their search. A well-known exception is the CMA-ES that performs a rotation-invariant search, and is thus independent of any coordinate system. The basic idea of CMA-ES is to evolve, besides a population of solutions to the optimization problem at hand, a “Covariance Matrix” that can be viewed as a coordinate transform: in case of a quadratic objective function, CMA-ES Covariance Matrix has been empirically demonstrated to gradually converge to a matrix which is proportional to the inverse Hessian matrix of the objective function. In the coordinate system defined by this inverse Hessian, the quadratic
objective function has become separable, and the optimization problem, almost trivial. Of course, CMA-ES covariance matrix is only, in the quadratic case, an approximation of the inverse Hessian. And many interesting problems are not quadratic indeed (and if they were, they would be easy to solve directly). Nevertheless, twice continuously differentiable objective functions can be viewed as close-to-quadratic around their optima (local or global), and adapting the coordinate system with respect to the “cumulative path” of the search makes it easier and faster to find the optimum.

The basic principles of this adaptive coordinate transformation have been generalized to general search strategies, under the name of Adaptive Encoding in [Hansen, 2008], and experimented with Cauchy mutations in a stochastic search framework. The original Cauchy-based ES is heavily coordinate-dependent, and its results deteriorate when the degree of non-separability of the objective function increases. However, this limitation of Cauchy mutation almost vanishes with Adaptive Encoding, demonstrating the usefulness of a well-designed adaptive coordinate system.

Putting things together, a natural idea is then to couple with Adaptive Encoding some simple optimization method, i.e., some successive coordinate-wise line searches: coordinate line-searches only work well for separable functions, but Adaptive Encoding should gradually lead the search toward a transformed coordinate system where the objective function resembles more a separable function than in the original system, paving the road for the coordinate line-searches. Though the resulting algorithm has little to do with Evolutionary Computation, it heavily relies on Adaptive Encoding, the backbone of CMA-ES algorithms. The idea of adapting the coordinate system during the search can be attributed to the famous Rosenbrock’s method [Rosenbrock, 1960].

This Section is organized the following way: Section 6.1.2 first introduces the algorithmic background, namely Adaptive Encoding and some Coordinate Descent Method, before detailing their coupling into the Adaptive Coordinate Descent algorithm. Section 6.1.3 presents the experiments that validate Adaptive Coordinate Descent first on the Sphere function, the well-known separable test function, establishing performance bounds for the proposed approach. Extensive experiments on the BBOB testbed are then presented and discussed. Finally, Section 6.1.4 concludes the Section and sketches directions for further research.

6.1.2 Algorithms

6.1.2.1 Adaptive Encoding

Though historically introduced as a derandomization of self-adaptive Evolution Strategies [Hansen et al., 2003], CMA-ES was only recently revisited as a hybrid between some ES with adaptive step-size and some Adaptive Encoding (AE) procedure [Hansen, 2008]. AE can be applied to any continuous domain search algorithm, in order to make it independent from any given coordinate system. As a result, some search algorithms that performed rather poorly on non-separable functions can be tremendously boosted (e.g., by a factor up to 3 orders of magnitude for Evolution Strategy with Cauchy mutation [Hansen, 2008]).

An iteration of CMA-ES, decomposed into Adaptive Encoding and Evolution Strategy
Chapter 6. Exploring new frontiers using CMA-like algorithms

a) Principal Component Analysis

b) Adaptive Encoding Update

c) Coordinate Descent Method

d) Adaptive Coordinate Descent Method

Figure 6.1: AE_CMA-like Adaptive Encoding Update (b) mostly based on Principal Component Analysis (a) is used to extend some Coordinate Descent method (c) to the optimization of non-separable problems (d). See text for details.

Algorithm 6.1: CMA-ES = Adaptive Encoding + ES

1: $x_i \leftarrow m + \sigma N_i(0, I)$, for $i = 1 \ldots \lambda$
2: $f_i \leftarrow f(Bx_i)$, for $i = 1 \ldots \lambda$
3: if Evolution Strategy (ES) with 1/5th success rule then
4: $\sigma \leftarrow \sigma \exp \propto \left(\frac{\text{success rate}}{\text{expected success rate}} - 1\right)$
5: end if
6: if Cumulative Step-Size Adaptation ES (CSA-ES) then
7: $\sigma \leftarrow \sigma \exp \propto \left(\frac{\text{evolution path}}{\text{expected evolution path}} - 1\right)$
8: end if
9: $B \leftarrow \text{AdaptiveEncoding}(Bx_1, \ldots, Bx_\mu)$

with step-size adaptation, is described in Algorithm 6.1. In standard Evolution Strategy, $\lambda$ offspring are sampled (line 1) from a normal distribution with step-size $\sigma$ and mean $m$, where $m$ is the centroid of best $\mu$ individuals of the previous iteration. The $\lambda$ offspring are evaluated (line 2 with $B = I$ the identity matrix). Depending on the choice of the step-size adaptation rule, the step-size is then adapted, either by some rule similar to the one-fifth rule [Rechenberg, 1973] (line 4), or using the Cumulative Step-size Adaptation [Hansen and Ostermeier, 1996] (line 7).

CMA-ES differs from standard ES on lines 2 and 9, that describe the use of the Adaptive Encoding procedure. CMA-ES maintains a coordinate system transformation matrix $B$, and though it evaluates the individuals in the original coordinate system of $\mathbb{R}^n$ (line 2), it generates the offspring, using some isotropic normal distribution, in some transformed
coordinate system (line 1). The $n \times n$ matrix $B$ is the matrix of the transformation. In Algorithm 6.1, offspring $x_i$ are represented in this transformed coordinate system, and $Bx_i$ are their images in the original coordinate system. Matrix $B$ is iteratively adapted by the AE procedure using information from the most successful $\mu$ offspring (line 9).

The CMA update rule for $B$, denoted as $\text{AE_{CMA}}$, derived from the original Covariance Adaptation rule of the $(\mu, \lambda)$-CMA-ES [Hansen and Ostermeier, 2001, Hansen et al., 2003], is detailed in Algorithm 6.2. The Adaptive Encoding update is similar to some Principal Component Analysis (PCA) of the successful search steps. The goal of PCA is to find an orthogonal transformation to convert the set of possibly correlated variables into a set of uncorrelated variables, called principal components, that are the eigenvectors of the covariance matrix of the data. However, while PCA is usually used to reduce the dimensionality of the data by taking into account only the main principal components (corresponding to the largest eigenvalues), CMA retains all principal components. These components are determined at each iteration by the eigendecomposition of the current covariance matrix $C$ (line 16 of Algorithm 6.2). The transformation matrix $B$ is the square-root of the covariance matrix $C$ (line 17). An illustration of Principal Components Analysis is shown in Figure 6.1-a, where the principal components are depicted as the dotted lines, such that the largest variance by any projection of the data comes to lie on the first principal component, the second largest variance on the second, and so on. The idea of such a transformation is to make the objective function in the transformed space as similar as possible to the Sphere function, which is known to be simple for analysis and optimization.

Figure 6.1-b illustrates an Adaptive Encoding Update iteration, where only the $\mu$ (green/bold) best among $\lambda$ generated offspring are used to compute a partial covariance matrix $C_\mu$ (line 14), which replaces a fraction $c_\mu$ of the current covariance matrix $C$ (line 15). Additionally, the path of the mean of distribution (evolution path $p$) is recorded in order to increase the variance of favorable directions (line 13, and bold arrow in Figure 6.1-b). A fraction $c_1$ of the current covariance matrix $C$ is also replaced by the rank-one matrix of eigendirection $p$ (line 15). However, the update in line 15 might become unstable if $c_\mu + c_1 > 1$. Hence the parameter setting (line 3) needs to be chosen specifically for the algorithm at hand.

It has been shown in [Hansen, 2008] that the $\text{AE_{CMA}}$-update applied to Evolution Strategy with Cauchy distribution improves the performance on some non-separable functions by a factor of roughly one thousand. These results and the fact that the Sphere-like transformed space is usually simpler to analyze than the original one, make it reasonable to explicitly exploit this property in search.

### 6.1.2.2 Coordinate Descent by Dichotomy

Coordinate Descent (CD) is probably one of the oldest multidimensional optimization method. It became especially popular in numerical linear algebra under the name of Gauss-Seidel method for solving systems of linear equations. In Evolutionary Computation community, when used for optimization, this method is called Coordinate Strategy [Schwefel, 1993]. CD is based on the idea that an $n$-dimensional optimization problem
Algorithm 6.2: Adaptive Encoding

1: given $x_1, \ldots, x_\mu$
2: if Initialize then
3: \hspace{1em} $w_i \leftarrow \frac{1}{\mu}$; $c_p \leftarrow \frac{1}{\sqrt{n}}$; $c_1 \leftarrow \frac{0.5}{n}$; $c_\mu \leftarrow \frac{0.5}{n}$
4: \hspace{1em} $p \leftarrow 0$
5: \hspace{1em} $C \leftarrow I$; $B \leftarrow I$
6: \hspace{1em} $m \leftarrow \sum_{i=1}^{\mu} x_i w_i$
7: \hspace{1em} return.
8: end if
9: 
10: \hspace{1em} $m^{-} \leftarrow m$
11: \hspace{1em} $z_0 \leftarrow \sqrt{n} \left\| B^{-1}(m - m^-) \right\| (m - m^-)$
12: \hspace{1em} $z_i \leftarrow \sqrt{n} \left\| B^{-1}(x_i - m^-) \right\| (x_i - m^-)$
13: \hspace{1em} $p \leftarrow (1 - c_p)p + \sqrt{c_p}(2 - c_p)z_0$ \hspace{1em} // evolution path update
14: \hspace{1em} $C_{\mu} \leftarrow \sum_{i=1}^{\mu} w_i z_i z_i^T$ \hspace{1em} // rank-$\mu$ update covariance matrix
15: \hspace{1em} $C \leftarrow (1 - c_1 - c_\mu)C + c_1 pp^T + c_\mu C_{\mu}$ \hspace{1em} // covariance matrix update
16: \hspace{1em} $B^*DDB^{*T} \leftarrow \text{eigendecomposition}(C)$
17: \hspace{1em} $B \leftarrow B^*D$
18: return $B$

can be decomposed into $n$ one-dimensional sub-problems. Each variable is updated in turn, while all other variables remain fixed, by solving a one-dimension optimization sub-problem using any suitable one-dimension optimization algorithm. Note that CD can be viewed as a special case of Block Coordinate Descent, that partitions the coordinates into $N$ blocks: $f$ is iteratively optimized with respect to one of the coordinate block while other coordinates are fixed [Tseng, 2001]. Obviously, it is reasonable to use CD when dealing with unimodal separable problems.

Adaptive Dichotomy

One of the simplest one-dimension optimization algorithm to use is a dichotomy method (inspired by the bisection method to find a zero of a given function). Let us consider an interval $[a, b]$ where the optimum is known to lie, and assume that the value of the objective function $f$ is known at the center $m = \frac{a+b}{2}$ of the interval. Evaluate the two points $x_1 = m - \frac{b-a}{2}$ and $x_2 = m + \frac{b-a}{2}$, centers of the left and right parts of $[a, b]$. If $f$ is unimodal, only three cases are possible: $x_1$ is better than $m$ and $x_2$, $x_2$ is better than $m$ and $x_1$, or both $x_1$ and $x_2$ are worse than $m$ (if $x_1$ and $x_2$ are both better than $m$, then the problem is multi-modal).

If $x_1$ is better than $m$ and $x_2$, then the optimum lies in the interval $[a, m]$; replace $b$ with $m$ and $x$ with $x_1$. Similarly, if $x_2$ is better than $m$ and $x_1$, replace $a$ with $m$ and $m$ with $x_2$. Finally, if $x_1$ and $x_2$ are worse than $m$, then the optimum lies in the interval $(x_1, x_2)$; replace $a$ with $x_1$ and $b$ with $x_2$. In all 3 cases, we end up with a new interval $[a, b]$ which contains the optimum, whose length is half that of the original $[a, b]$, and for which
6.1 Adaptive Coordinate Descent

Figure 6.2: Coordinate Descent (CD) (a-e) and Adaptive Coordinate Descent (ACiD) (f) on Sphere (a-d) and Rosenbrock (e,f) functions. The initial point $x_0 = (-3.1,-4,1)$ and the target point $x_{target}$ with $f(x_{target}) < 10^{-10}$. While the CD with the dichotomy (a,b) performs best on the Sphere function cyclically dividing by two the step size for the corresponding coordinate (depicted as the line), the increasing of the step size by factor $k_{succ}$ in the case of successful sampling leads to a better but still slow convergence on non-separable Rosenbrock function (e). The adaptation of the coordinate system allows significantly speed up the search (f).
we know the value of $f$ at its center, thus closing the loop, and ready for next iteration.

When dealing with multi-dimensional problems, dichotomy steps can be achieved on each coordinate successively: Figure 6.1-c illustrates the 2-dimensional case and displays an example of one dichotomy step in each direction.

Another point of view on the dichotomy method is to consider it as a derandomized $(1 + 2)$-EA algorithm with step-size adaptation: Assuming the current step-size is $\sigma$ and current solution is $m$, the basic step of the dichotomy method described above generates 2 offspring $x_1$ and $x_2$ in a deterministic way. The best of $m$, $x_1$, and $x_2$ becomes the next parent, and $\sigma$ is divided by 2. In the case of one-dimensional unimodal problems, if the initial interval contains the global optimum, this algorithm will get an approximation as wanted of. Similarly, in the case of multi-dimensional unimodal problems, if the initial rectangle contains the global optimum, the algorithm will find it, either by running the dichotomy method on each coordinate up to a given precision (see Figure 6.2-a), or by alternating one step of the dichotomy method in each direction in turn (see Figure 6.2-b). Figure 6.2-a shows the result of such an optimization of the Sphere function $f(x) = \sum_{i=1}^{n} x_i^2$ starting from the initial point $x_0 = (-3.1, -4.1)$. Dichotomy proceeds for 20 iterations for the first coordinate and then for 20 iterations for the second coordinate, reaching the target function value $10^{-10}$ after 80 evaluations. Exactly the same result is obtained by cyclically repeating this procedure over each coordinate in turn, as shown in Figure 6.2-b. The second variant, however, seems to be more robust if the problem is not perfectly separable, exploring a larger region of the search space rather than rapidly reducing one dimension to a single value.

However, if the optimum lies outside the initial interval, or if the interval is somehow transformed after a rotation of the coordinate system (e.g., due to Adaptive Encoding, see Section 6.1.2.3), it might be necessary to allow more exploration in case of successful sampling (one offspring was better than the parent $m$). Such dichotomy method with step-size (interval) adaptation will be called Adaptive Dichotomy (AD), and works as follows: Generate two offspring $x_1$ and $x_2$ as above. If at least one of these two offspring is better than its parent $m$, then $\sigma \leftarrow \sigma k_{\text{succ}}$, otherwise $\sigma \leftarrow \sigma k_{\text{unsucc}}$. In the case of standard dichotomy, $k_{\text{succ}} = k_{\text{unsucc}} = 0.5$, which is suitable for the unimodal separable problems, when initial interval contains the optimum. However, whereas $k_{\text{unsucc}} = 0.5$ seems a good choice for all situations, and will be used in the following, $k_{\text{succ}} > 0.5$ is mandatory in most cases (otherwise, even on the Sphere function, the algorithm will not converge if the initial domain does not contain the optimum). Figure 6.2-c and Figure 6.2-d illustrate runs where the optimum does not lie in the initial rectangle. However, with $k_{\text{succ}} = 1.0$ and $k_{\text{unsucc}} = 2.0$ respectively, the Coordinate Descent with Adaptive Dichotomy converges, though at the price of additional functions evaluations.

A more formal description of CD with Adaptive Dichotomy will be given in Section 6.1.2.3 (Algorithm 6.3).

**Convergence Rates** Before turning to Adaptive Encoding and non-separable functions, let us analyze the convergence rate of CD with Adaptive Dichotomy on the Sphere function, and compare it that of standard Evolution Strategies, whose behavior is well studied in this context.
6.1 Adaptive Coordinate Descent

Figure 6.3: **Left**: Evolution of distance to the optimum versus number of function evaluations for the (1+1)-ES, (1+1)-ES opt, CD $k_{\text{suc}}=0.5$, CD $k_{\text{suc}}=1.0$, CD $k_{\text{suc}}=2.0$ and CD $k_{\text{suc}}=2.0$ ms on $f(x) = \|x\|^2$ in dimension 10. **Right**: Convergence rate $c$ (the lower the better) multiplied by the dimension $n$ for different algorithms depending on the dimension $n$. The convergence rates have been estimated for the median of 101 runs.

Linear convergence to the optimal point $x^*$ takes place if there is a constant $c \neq 0$, such that

$$\frac{1}{T_k} \ln \frac{\|x_k - x^*\|}{\|x_0 - x^*\|} \to c,$$

(6.1)

where $x_0$ is the initial point and $x_k$ the best point found after $k$ iterations for a cost of $T_k$ fitness function evaluations.

The empirical convergence rate on the Sphere function of the proposed CD with $k_{\text{suc}} = 0.5, 1.0$ and $2.0$, as well as that of two variants of (1+1)-Evolution Strategy, as estimated from the median of 101 independent runs, is shown in Figure 6.3. The algorithm denoted as "(1+1)-ES" corresponds to the (1+1)-Evolution Strategy with the initial step-size $\sigma_0 = 1.8$, while the search interval is $[-3; 3]^n$. The "(1+1)-ES opt" algorithm is the (1+1) Evolution Strategy with the scale-invariant step-size: the optimal step-size for ES on Sphere function is proved to be $\sigma = \frac{\sqrt{n}}{n} \|x - x^*\|$, i.e., proportional to the distance to the optimum.

It is clear that the convergence rate of the CD with (standard) dichotomy ($k_{\text{suc}} = 0.5$) is linear with dimension $n$, and is equal to $-\ln(2)/2n = -0.3465/n$. The rates for CD with Adaptive Dichotomy, with $k_{\text{suc}} = 1.0$ and $k_{\text{suc}} = 2.0$ are 1.5 and 2.0 times slower, respectively, than with $k_{\text{suc}} = 0.5$.

The recently proposed technique of **mirrored sampling** (ms) and **sequential selection** for Evolution Strategy [Brockhoff et al., 2010], can also be used with the CD method proposed here, because the sampled points are symmetric by definition. We hence propose a modified version of the CD with Adaptive Dichotomy algorithm, called "CD $k_{\text{suc}}=2.0$ ms ", in which the second offspring is not evaluated if the first one is better than the
Chapter 6. Exploring new frontiers using CMA-like algorithms

parent. This does reduce the number of fitness function evaluations, though marginally, as can be seen in Figure 6.3 (line with label “ms”).

It is important to note that the optimal convergence rate of “(1+1)-ES opt” is not achievable in practice, because the optimal step-size is unknown for a given black-box function. In the case of CD, parameter \( k_{succ} \), which controls the exploration rate, can be used to implicitly tune the target convergence rate.

A final remark on one-dimensional algorithms: Obviously, any other one-dimensional optimization method could be used instead of the Dichotomy method. The Golden Section method (also called Fibonacci method) is known to have a better convergence rate \( c = -\ln(1 + \sqrt{5})/n = -0.4812/n \). However, the Golden Section generates new points with respect to two evaluated points on a line. Therefore, after a change of coordinate (due to Adaptive Encoding, see next Section) it will require to recompute the fitness of these rotated point whereas dichotomy only requires the fitness value of the center of the current domain, that is preserved by the change of coordinate.

6.1.2.3 Adaptive Coordinate Descent

The Adaptive Encoding procedure (Section 6.1.2.1) iteratively learns the coordinate systems in which the objective function is “as close to separable as possible”. The Coordinate Descent method (Section 6.1.2.2) takes advantage of the separability of the problem at hand, iteratively optimizing \( n \) independent one-dimensional problems. Combining both approaches leads to propose Adaptive Coordinate Descent (ACiD), which benefits from these two ideas, interleaving CD and AE, learning the same coordinate transform than the original AE, and performing CD steps in the transformed space.

Figure 6.1-d illustrates how AE acts on the iterations of one-dimensional search steps of Figure 6.1-c. The advantages of ACiD over CD become obvious on non-separable functions. Figure 6.2-e and Figure 6.2-f show sample runs of CD and ACiD respectively, on Rosenbrock function in 2-D: ACiD (with \( k_{succ} = 2.0 \)) quickly adapts an appropriate coordinate system and finds the optimum 70 times faster than CD.

Algorithm 6.3 describes the proposed ACiD algorithm. Note that it also describes the non-adaptive CD method by taking \( B = I \) and removing the call to AdaptiveEncoding (line 30). Algorithm starts by randomly initializing the current parent \( m \) uniformly in the given rectangle domain \( \Pi_i[x_i^{min}, x_i^{max}] \), and evaluating it. Initial step-sizes \( \sigma_i \) are set to \( 1/4 \) of the corresponding interval length. \( i_x \) is the index of the current coordinate, going from 1 to \( n \) cyclically (line 8). The two offspring \( x_1 \) and \( x_2 \) are generated from \( m \) with offset \( \pm \sigma_{i_x} \) on coordinate \( i_x \) (lines 9-11). If one of them has better fitness value than \( m \) (minimization assumed here), \( m \) and \( f_{best} \) are updated accordingly (line 14 or 17). The \( i_x \) step-size \( \sigma_{i_x} \) is then updated multiplicatively depending on the success indicator (line 21 or 23). The coordinates and fitness of both offspring are then stored (lines 25 and 26).

At the end of the coordinate loop (i.e., when \( i_x = n \), line 29), the Adaptive Encoding procedure is called to update the transformation matrix \( B \), using information from the \( \mu \) best offspring encountered during the \( n \) coordinate steps (line 30).

---

1 Random cycles were also tried, but no significant impact on performance was ever observed, so only the cyclic variant is shown here for the sake of simplicity.
6.1.3 Experimental Validation

Adaptive Coordinate Descent has been benchmarked on the noiseless BBOB testbed. Thanks to the publically available results of many algorithms on the same testbed, and to automatic comparison procedures provided by this framework, ACiD results will be compared to those of the state-of-the-art algorithms: BIPOP-CMA-ES, IPOP-CMA-ES,
Figure 6.4: Performance analysis of ACiD in $\alpha_{cov} \times \alpha_{c Theta}$ space. All colorscales are different. See text for details.
6.1 Adaptive Coordinate Descent

IPOP-aCMA-ES, (1+1)-CMA-ES and \((1+2_s^m)\)-CMA-ES [Auger et al., 2010].

6.1.3.1 Adaptive Encoding Calibration

The main parameters in Adaptive Encoding procedure that [Hansen, 2008] suggested to calibrate are \(c_1\) and \(c_\mu\). To scale them with the dimension we can define: \(c_1 \leftarrow \frac{0.5}{n^{\alpha_1}}\) and \(c_\mu \leftarrow \frac{0.5}{n^{\alpha_\mu}}\) (with \(k_{\text{succ}} = 2\)). Figure 6.4 illustrates the potential speedup for different values of \(\alpha_1\) and \(\alpha_\mu\) w.r.t. to the baseline algorithm (\(\alpha_1 = 1.0\) and \(\alpha_\mu = 1.0\)). First, for each of 8 benchmark problems the baseline SP1 value to reach the target value 1e-10 is calculated (30 runs, \(\alpha_1 = 1.0\) and \(\alpha_\mu = 1.0\)). Then this procedure is repeated for 10000, 4000 and 1500 times with random values of \(\alpha_1\) (x-axis) and \(\alpha_\mu\) (y-axis) on 2-, 10- and 40-dimensional problems respectively (left, center, right). The average speedup for 8 tested problems is plotted on top. It should be noted that the bottom left corner (respectively, top right corner) of each small plot corresponds to large values of \(c_1\) and \(c_\mu\) (respectively, small values of \(c_1\) and \(c_\mu\)).

It can be clearly seen that optimal parameters of \(\alpha_1\) and \(\alpha_\mu\) depend very much both on the problem and its dimension. In 2-D the results are somewhat difficult to interpret, while in 10-D and 40-D, they are much more illustrative. Some general observations:

- very large \(\alpha_1\) and \(\alpha_\mu\) (bottom left corner) slow down the search, because the performed PCA takes into account only local information, obtained from few recent individuals, and, therefore is not sufficiently correct.
- very small \(\alpha_1\) and \(\alpha_\mu\) (top right corner) also slow down the search, because the covariance matrix stores a lot of outdated information.
- the optimal \(\alpha_1\) and \(\alpha_\mu\) are located somewhere in the middle, but often close to the "bad" region, where the performance quickly drops.
- on some problems the Pareto shape is observed in the sense that a set of \(\alpha_1\) and \(\alpha_\mu\) from the same "front" leads to the same performance.

The baseline results (for \(\alpha_1 = 1\) and \(\alpha_\mu = 1\)) on Sphere and Linear Slope functions can be improved by using quite large \(c_1\) and \(c_\mu\). On Sum of Different Powers, the results are very sensitive to \(c_1\) and \(c_\mu\) and the optimal region is quite small. It would be interesting to note that while the region of average speedup (top) seems to be quite large, the actual region of choice is rather constrained by the region of Rotated Ellipsoid function, where the speedup quickly drops if \(\alpha_1\) or \(\alpha_\mu\) is smaller than 0.8 (in 40-D). Another important observation is that \(\alpha_1\)-\(\alpha_\mu\) hyper-parameters optimization problem sometimes is multi-modal (let us forget about noise) as can be clearly observed for 40-dimensional Discus function, where two peaks are located symmetrically to each other! This is rather unexpected observation.

In overall, the baseline results can be improved by 10-70% by choosing optimal \(\alpha_1\) and \(\alpha_\mu\) parameters for each problem.
6.1.3.2 Coordinate Descent Calibration

Next experiment is concerned with the sensitivity of parameter $k_{\text{succ}}$. Figure 6.5-Left shows the performance of ACiD in terms of BBOB-SP1 depending on $k_{\text{succ}}$, for several problems in 10-D. $k_{\text{succ}}$ determines how fast the step-size will increase for a given coordinate if the last step along that coordinate was successful. There is a strong link between $k_{\text{succ}}$ and the covariance matrix learning coefficients $c_1$ and $c_\mu$, since they both determine the comparative impact of the new steps.

The experiments show that ACiD does not converge for $k_{\text{succ}} \leq 1$ on non-separable problems, whereas on the Sphere function, ACiD obtains nearly the same results than CD with $0.5 \leq k_{\text{succ}} < 1.0$. The reason for this is easy to understand on a small example: if $k_{\text{unsucc}} = 0.5$ and $k_{\text{succ}} = 1.1$, then in the case of 2 consecutive unsuccessful steps, the step-size $\sigma$ is divided by 4; so in order to come back to initial step size, 13 consecutive successful steps are needed! The optimal pair $k_{\text{succ}}, k_{\text{unsucc}}$ of course depends on the problem, and on the other parameters of the ACiD, too. But $k_{\text{succ}} = \frac{1}{k_{\text{unsucc}}} = 2.0$ seems to be both robust and simple, at least for the BBOB problems.

6.1.3.3 Experimental Settings

In order to validate ACiD with a robust version, a value of $k_{\text{succ}} = 2$ will be used as a baseline (while $k_{\text{unsucc}} = 0.5$). Indeed, while other values for $k_{\text{succ}} \in [0.5, 2.0]$ may lead to faster convergence, they also sometimes result in premature convergence on some problems, even on the Sphere function, as demonstrated by preliminary experiments. Also note that the basic rates for the covariance matrix update are set to the simple values $c_1 = \frac{0.5}{n}$ and $c_\mu = \frac{0.5}{n}$ rather than the default ones $c_1 = \alpha c \frac{0.2}{(n+1.3)^2+\mu_w}$, $c_\mu = \alpha c \frac{0.2(\mu_w-2+\frac{1}{\mu_w})}{(n+2)^2+0.2\mu_w}$, where $\alpha c = 10$ for CMA-ES, leading to $c_1 \approx \frac{2}{n\pi}$, $c_\mu \approx \frac{\mu_w}{n\pi}$, [Hansen, 2008]). A total of 4 variants will be tested in the following: the baseline described above is denoted $ACiD$; $ACiD-k_{\text{succ}}1.5$ is the same variant, but with $k_{\text{succ}} = 1.5$; $ACiD-ms$ uses the value $k_{\text{succ}} = 2$, but implements mirrored sampling and sequential selection (Section 6.1.2.2); finally, $ACiD-a_{\mu}c\mu0.8$ uses $c_\mu = \frac{0.5}{n\pi}$ rather than the default value, as our experiments in Section 6.1.3.1 have shown that $c_\mu$ is a sensitive parameter indeed, and the value of $\frac{0.5}{n\pi}$ can bring up to 10-70% improvement on some problems.

Because ACiD can be considered as (1 + 2)-EA, a restart procedure is necessary to improve the performance on multi-modal functions. Similarly to the other (1 + 1)-CMA-ES that ACiD will be compared to within BBOB testbed, the algorithm is restarted if the improvements of the best solution is smaller than $10^{-25}$ during the last $10 + \lfloor 20n^{1.5} \rfloor$ function evaluations. The maximum number of function evaluations is $10^4 n$, and the initial interval is $[-3,3]^n$. All results are statistics over 15 independent runs, one run per BBOB problem instance (e.g., 15 runs on different instances of the Sphere functions).

The MatLab source code of ACiD is available online at http://sites.google.com/site/acdgecco/.
6.1.3.4 Results on Noiseless BBOB Testbed

Figure 6.5: **Left**: The performance of ACiD in **10-D**: BBOB-SP1 (the average number of function evaluations to reach target value $10^{-8}$ divided by success rate) versus step-size multiplier $k_{\text{succ}}$. **Right**: BBOB-like results for noiseless functions in **20-D**: ECDF for ACiD (continuous lines) and (1 + 1)-CMA-ES (dashed lines), of the running time, normalized by dimension $n$, needed to reach target precision $f_{\text{opt}} + 10^k$ (for $k = +1, -1, -4, -8$). The vertical black line indicates the maximum number of function evaluations. Light yellow (or gray in b&w) lines in the background show similar ECDFs for target value $10^{-8}$ of all algorithms benchmarked during BBOB 2009.

Figure 6.5-Right presents the ECDFs of both ACiD and (1 + 1)-CMA-ES aggregated over the 24 noiseless benchmark problems in 20-D. A closer look at the results (details not shown here) reveals that for 50% of the solved functions, ACiD is faster about by 40% than (1 + 1)-CMA-ES. ACiD converges on Attractive Sector function $f_6$, while (1 + 1)-CMA-ES does not. On Rosenbrock $f_8$ and $f_9$, Ellipsoid $f_2$ and $f_{10}$ and Discus $f_{11}$, some ACiD algorithms are up to 2 times faster than (1+1)-CMA. Furthermore, with a budget of 1000$n$ function evaluations, ACiD performs best in 20-D among all algorithms that entered BBOB-2009 competition.

Figure 6.6 gives a more general view of similar experiments in 10-D and 40-D – but here experiments with 50 different target values are aggregated. Overall, the functions are not easy to solve. Even the best CMA-ES algorithm, BIPOP-CMA-ES, can solve less than 90% of the problems using maximum number of function evaluations (the large crosses). Furthermore, the elitist algorithms (ACiD, (1 + 1)-CMA-ES, (1 + $2_{n_0}$)-CMA-ES) are more likely to get stuck in a local optimum than the generational algorithms (BIPOP-CMA-ES, IPOP-CMA-ES and IPOP-aCMA-ES), especially in the case where the step-size decreases after each unsuccessful step. The increase of the population size after restart in the case of the premature convergence is the main tool, which leads to a superior performance of the generational CMA-ES algorithms over the single-population algorithms on the multi-modal problems. Indeed, only 10 out of 24 problems are unimodal and this 42% threshold can be seen in Figure 6.6. However, the good news is that all ACiD algorithms have at least
comparable performance with \((1 + 1)\)-CMA-ES and \((1 + 2^s_m)\)-CMA-ES: they outperform one another depending on the problem, but the differences are not significant.

### 6.1.3.5 Extension to Noisy Optimization

The original ACiD is an elitist algorithm, where offspring replaces the parent only if it is better on \(f\). This approach seems to be too "greedy" if \(f\) is noisy and the algorithm may suffer from premature convergence. To reduce this effect we propose here a very simple modification: i). we manually set or check online whether \(f\) is noisy or not; ii). if \(f\) is noisy, at each iteration we re-evaluate the best solution found so far before the line 8 of Algorithm 6.3. This simple modification increases the budget of evaluations by 50% per iteration, but according to our (not shown here) preliminary experiments, drastically improves the results on functions with moderate noise. We omit these preliminary experiments for brevity.

### 6.1.3.6 Extension to Large-Scale Optimization

The original ACiD algorithm performs \(n\) dichotomy steps and evaluates 2 individuals iteratively on \(n\) principal components (overall \(2n\) evaluations), then calls the Adaptive Encoding procedure in line 30 of Algorithm 6.3. The computation complexity of the Adaptive Encoding procedure is defined by the eigendecomposition of the covariance matrix \(C\) and scales with \(O(n^3)\). Thus, performing the eigendecomposition every \(n\) iterations or \(2n\) function evaluations, the computational complexity of ACiD per function evaluation becomes quadratic. The computational complexity is also quadratic for i). \((\mu, \lambda)\)-CMA-ES, where the eigendecomposition of \(C\) is postponed, and ii). the recently proposed version of \((1+1)\)-CMA-ES [Suttorp et al., 2009] thanks to the use of Cholesky factor update. However, it
6.1 Adaptive Coordinate Descent

Figure 6.7: Results for ACiD with linear time complexity (LACiD) on Sphere, Ellipsoid and Rosenbrock functions for different problem dimensions. **Left**: SP1 to reach $10^{-8}$. **Right**: CPU time per function evaluation.

should be noted that there is no version of CMA-ES which can be linear in time and at the same time be able to learn the full covariance matrix (SEP-CMA-ES learns only the diagonal of $C$ [Ros and Hansen, 2008]). This is due to the procedure of sampling of new offspring, which involves matrix-vector multiplication which is already quadratic in $n$.

In order to reduce the computational complexity of ACiD we may replace $x'_i \leftarrow -\sigma_i x$; $x \leftarrow m + B x'$ in line 10 of Algorithm 6.3 by $x \leftarrow m - \sigma_i x B (i, i)^T$. Thus, we replace quadratic in $n$ matrix-vector multiplication by linear in time multiplication of vector ($i_x$-th eigenvector) and scalar. In contrast, in (full covariance matrix based) CMA-ES in order to generate an offspring we first should multiply its randomly generated pre-image vector $x'_1$ by $B$, that is quadratic in $n$. However, we still have overall quadratic complexity of ACiD due to eigendecomposition of $C$. To reduce the complexity to be linear we postpone the eigendecomposition (lines 16-17 of Algorithm 6.2) and perform it only every $n$ iterations. We also should remove rank-$\mu$ update (remove lines 12 and 14, set $c_{\mu} = 0$) which would lead to quadratic complexity of ACiD. We additionally simplify rank-one update such that $m$ equals to the best solution $x_{best}$ found so far. We call this modified version of ACiD, linear time ACiD (LACiD).

Figure 6.7-**Left** shows the results of (1+1)-CMA-ES and LACiD on Sphere, Ellipsoid and Rosenbrock function for different problem dimensions. It should be noted that the LACiD usually performs worse than the original ACiD because of described above simplifications. However, as can be seen from the results, LACiD performs as well as (1+1)-CMA-ES on Sphere function and slightly slower on Ellipsoid and Rosenbrock. The loss of ACiD in comparison with (1+1)-CMA-ES is growing with dimension and is about a factor of 1.5-2 for $n = 512$ (the results for (1+1)-CMA-ES are incomplete for large $n$ due to large required time of computations).

Figure 6.7-**Right** shows CPU time complexity per function evaluation (in seconds) for (1+1)-CMA-ES and LACiD, estimated on Ellipsoid problem. While ACiD is invariant
to orthogonal transformations of the search space, we generate a random initial coordinate system defined by matrix $B$ to imitate the optimization of Rotated Ellipsoid function (otherwise, the computation of fitness function would be very expensive itself). For our experiments we use fast version of $(1+1)$-CMA-ES proposed by [Suttorp et al., 2009], where the Cholesky factor of $C$ is used for sampling new solutions. The Cholesky factor (and its inverse) is iteratively updated, thus, requiring only matrix-vector multiplication operations. It can be seen from the Figure that $(1+1)$-CMA-ES scales almost quadratically with $n$ and very closely to the computational complexity of one matrix-vector multiplication operation. In contrast, LACiD scales almost linearly with $n$ and is only by a factor of 10 slower than one scalar-vector multiplication. Thus, the complexity of LACiD is about $10n$ operations.

In overall, the LACiD is about $n$ times faster in terms of time and is about 2 times slower in terms of number of function evaluations for $n > 1000$. If the objective function is relatively cheap to compute, then the overall speedup for $n = 1000$ is about a factor of 500.

The LACiD can be compared to SEP-CMA-ES, proposed by [Ros and Hansen, 2008]. However, the latter is not suitable for learning the full covariance matrix, and, therefore, does not solve Rotated Ellipsoid function in a reasonable number of function evaluations [Ros and Hansen, 2008]. The same is also true for the elegant Rank-One NES (linear time Natural Evolution Strategy), recently proposed by [Sun et al., 2011]. While CPU time experiments performed on different platforms and languages are hardly comparable, it should be noted that CPU cost of Rank-One NES is about $10^{-3}$ for $n = 512$, that is about a factor of 100 slower than LACiD. As well as SEP-CMA-ES, Rank-One NES can be used for non-separable optimization, but is efficient if only one (or few) principal components are sufficient to reach the optimum quickly. However, it is not able to benefit from learning all principal components.

### 6.1.4 Conclusion and Perspectives

The very powerful Covariance Matrix Adaptation part of the state-of-the-art CMA-ES algorithm has been generalized into the Adaptive Encoding procedure that can be used in conjunction with any optimization algorithm, gradually learning an optimal coordinate system where the objective function at hand is (in the best case, and at least locally) separable. Simple yet powerful algorithms can be used to optimize separable functions, as for instance the Coordinate Descent (CD), that performs up to 2 times faster than the $(1 + 1)$-ES on the Sphere function. The Adaptive Coordinate Descent algorithm (ACiD) uses AE coupled with adaptive CD. ACiD has been shown to be competitive with the CMA-ES algorithms and even up to 2 times faster than $(1 + 1)$-CMA-ES on several functions of the BBOB test suite.

The superiority of the ACiD as an absolutely deterministic algorithm was not obvious a priori. These experiments confirm the hypothesis that the efficiency of CMA-ES is greatly due to the Adaptive Encoding procedure, and that the second component of the algorithm (Evolution Strategies, and Gaussian mutations) can be replaced without significant (or even any) loss of performance, at least in the case of the single-individual algorithms.
The IPOP-aCMA-ES is the only algorithm among the ones presented here that uses all $\lambda$ offspring in its covariance matrix update. While the best $\mu$ points are used to increase the variance along the successful directions, the worst $\lambda - \mu$ points are used with negative sign to exclude irrelevant directions of search. It is clear that such kind of negative update can be applied to ACiD too. This will be the subject of further work.

Of course, the generational versions of CMA-ES outperform the single-individual ones like (1 + 1)-CMA-ES and ACiD, on most multi-modal problems. But this is essentially due to the restarts with increasing population size. Some further work will be concerned with designing a generational extension of ACiD.

Partial experiments indicate that the off-line tuning of the covariance matrix learning rates $c_1$ and $c_\mu$ can lead to 10-70% speedup depending on the problem and dimension: more detailed experiments must be made in this direction. However, CMA-ES algorithms would of course also profit from tuning their parameters for the BBOB benchmark.

Borrowing ideas from [Pošík, 2004], an extension of the Adaptive Encoding procedure to the non-linear case using Kernel Principal Component Analysis (KPCA) [Scholkopf et al., 1996] is envisioned. Such extension should for instance make it feasible to sample the non-linear distribution along the parabolic shaped optimal valley of the Rosenbrock function (see Section 6.4.1).

In ACiD, the evolution path somehow approximates the gradient of the fitness function, and this information is used in the Adaptive Encoding update. However, the line search along the gradient could also be performed explicitly, as in quasi-Newton methods (e.g., BFGS method [Shanno, 1970]) and Pattern Search methods (e.g., Hooke and Jeeves method [Hooke and Jeeves, 1961]).

We anticipate successful applications of ACiD algorithm to constrained problems. For CMA-ES in large dimensions, resampling the infeasible points does not work, and leads to a rapid decrease of the step-size that further limits the exploration [Arnold and Hansen, 2012]. Within ACiD, the resampling on a line is easy, both in the transformed and in the original spaces.

Another possible extension of ACiD is concerned with surrogate models: the computationally cheap meta-model assisted one-dimensional search becomes favorable even with some budget of 3 to 5 function evaluations, at least for unimodal problems. Furthermore, in order to preserve the invariance properties of the ACiD, comparison-based surrogate models can be used (see Chapter 4).

Finally, the one-dimensional search procedure used in ACiD could be replaced by some $k$-dimensional search ($k \leq d$). For $k = 2$, the budget is $2k = 4$ function evaluations to find the best of 8 possible states (see Figure 6.1-c). By taking into account all available information, such as the projection of the evolution path on 2-D, we could increase the chances to directly find new best points, for example in the corner. In this case, by simply increasing both step-sizes, the resulting speedup would increase to 4. We suppose that even such simple strategies, together with sequential selection, can make ACiD significantly faster.
Chapter 6. Exploring new frontiers using CMA-like algorithms

6.2 Reward-based Parent Selection

In Steady State variants of Multi-Objective CMA-ES (SS-MO-CMA-ES, see Section 2.4.4) an offspring is generated from a uniformly selected parent. In our own experiments (see Section 5.4.4) we observed that such selection may be inefficient if only few parents may significantly improve current optimal Pareto front approximation. Figure 6.8 illustrates two examples where this may be the case. A common case is when some non-dominated points are already relatively close to the optimal Pareto front (Figure 6.8-Left), while other non-dominated points are quite far but their mating selection might lead to a faster convergence. A less common case we observed on IHR1 problem (a similar behaviour is shown in Figure 6.8-Right, see also Section 5.4.4) when one offspring typically has better chances to improve the overall quality of the population. In both cases it might be reasonable to encourage parents which best contribute to the convergence toward the optimal Pareto front.

In this Section, we investigate some alternative parent selection schemes for SS-MO-CMA-ES, based on the conjecture that not all (non-dominated) parents are equal. The new schemes involve the definition of multi-objective rewards, estimating the expectation of the offspring survival and its hypervolume contribution. Two selection modes, respectively using tournament, and inspired from the Multi-Armed Bandit framework [Auer et al., 2002], are used on top of these rewards.

This Section provides a more detailed description of the work published in [Loshchilov et al., 2011b]. Section 6.2.1 details the proposed parent selection operators and how they fit in the generic scheme of MO-CMA-ES. In Section 6.2.2, the resulting algorithms are experimentally assessed on some well-known benchmark unimodal functions, comparatively to the previous versions of MO-CMA-ES. Section 6.2.3 provides a discussion with some perspectives for further research.
6.2 Reward-based Parent Selection

6.2.1 New Parent Selections for Steady-State MO-CMA-ES

The original generational and steady-state versions of MO-CMA-ES are described in Section 2.4.4.

After [Igel et al., 2007a], the more "greedy" variant \( \mu + 1 \)-MO-CMA outperforms all other variants on unimodal problems. In contrast, on multi-modal problems such as ZDT4 and IHR4, \( \mu + 1 \)-MO-CMA performs better than \( \mu + 1 \)-MO-CMA [Igel et al., 2007a], but it does not perform too well, and neither does the generational version of MO-CMA-ES, comparatively to other MOEAs (see, e.g., [Igel et al., 2007b] for a comparison of the generational MO-CMA-ES with NSGA-II on ZDT4).

These remarks naturally lead to propose more "greedy" parent selection operators within SS-MO-CMA-ES (line 4 of Algorithm 6.4), in order to further improve its performances on unimodal problems, leaving aside at the moment the multi-modality issue. A parent selection operator is based on i). a selection mechanism (line 4 of Algorithm 6.4, and Algorithm 6.5), and ii). a rewarding procedure (line 16 of Algorithm 6.4, and Algorithm 6.6). A family of such operators is presented in this Section; the selection procedure either is based on a standard tournament selection (Section 6.2.1.1), or inspired from the Multi-Armed Bandit paradigm (Section 6.2.1.2). The rewarding procedures are described in Section 6.2.1.3.

6.2.1.1 Tournament Selection

Standard tournament selection is parameterized from a tournament size \( t_{size} \in \mathbb{N} \). Given a set \( Q \), \( t_{size} \)-tournament selection proceeds by uniformly selecting \( t_{size} \) individuals (with or without replacement) from \( Q \) and returning the best one according to the criterion at hand (here, the \( \prec_Q \) criterion, see Section 2.4.1). The parent selection procedure (line 4 of Algorithm 6.4) thus becomes tournament selection (line 11 of Algorithm 6.5). The rewarding procedure (line 16 of Algorithm 6.4) only computes for each parent its Pareto rank and hypervolume contribution.

The Steady-State MO-CMA-ES using \( t_{size} \)-size Tournament Selection is denoted \((\mu + t_{size} 1)\)-MO-CMA in the following, or \((\mu + t_{size} 1) \) for short. Parameter \( t_{size} \) thus controls the selection greediness; the larger \( t_{size} \); the more often points with high hypervolume contribution will be selected on average.

6.2.1.2 Multi-Armed Bandit Selection

Another parent selection procedure (line 4 of Algorithm 6.4) inspired from the Multi-Armed Bandit (MAB) paradigm is described here. How to define the underlying rewards (line 16 of Algorithm 6.4) will be detailed in next Section.

The standard MAB setting considers several options, also called arms, each one with an (unknown but fixed) reward distribution [Auer et al., 2002]. The MAB problem is to find a selection strategy, selecting an arm \( i \) in each time step \( t \) and getting an instance of the corresponding reward distribution, such that this strategy optimizes the cumulative reward.
Chapter 6. Exploring new frontiers using CMA-like algorithms

Algorithm 6.4: \((\mu+\lambda)\)-MO-CMA-ES

1. initialize \(t \leftarrow 0\), random parent population \(Q^t=0\)
2. repeat
3. for \(k = 1, \ldots, \lambda\) do
4. \(i_k \leftarrow ParentSelection(Q^t, k)\)
5. Generate offspring \(a_k^{t+1}\) using parent \(a_{i_k}^t\)
6. \(Q^t \leftarrow Q^t \cup \{a_k^{t+1}\}\)
7. end for
8. for \(k = 1, \ldots, \lambda\) do
9. Update CMA-ES parameters of \(a_k^{t+1}\) and \(a_{i_k}^t\)
10. end for
11. \(Q^{t+1} \leftarrow \{Q^t_{\prec q} | 1 \leq i \leq \mu \}\) // Deterministic Selection according to \(\prec q^t\)
12. for \(k = 1, \ldots, \lambda\) do
13. \(i^t_{\text{parent}} \leftarrow i_k\)
14. \(i^{t+1}_{\text{parent}} \leftarrow \{i_{a_{i_k}^t} = a_{i_k}^t | 1 \leq i \leq \mu + \lambda\}\)
15. \(i^{t+1}_{\text{offspring}} \leftarrow \{i_{a_{i_k}^t} = a_{i_k}^{t+1} | 1 \leq i \leq \mu + \lambda\}\)
16. \(Q^{t+1} \leftarrow \text{ComputeRewards}(Q^t, Q^{t+1}, i^t_{\text{parent}}, i^{t+1}_{\text{parent}}, i^{t+1}_{\text{offspring}}, \tau, W)\)
17. end for
18. \(t \leftarrow t + 1\)
19. until stopping criterion is met.

The Upper Confidence Bound (UCB) algorithm yielding an optimal result has been proposed by [Auer et al., 2002]; this algorithm proceeds by selecting the arm which maximizes

\[
\hat{q}_{i,\tau} + C \sqrt{\frac{2\log \sum_k n_{k,\tau}}{n_{i,\tau}}},
\]  

(6.2)

the sum of an exploitation term \(\hat{q}_{i,\tau}\) (the empirical quality, or average of rewards the arm has ever actually received) and an exploration term \(\sqrt{\frac{2\log \sum_k n_{k,\tau}}{n_{i,\tau}}\)} (ensuring that non-optimal arms will be selected sufficiently often to enforce the identification of the truly optimal arm). The constant \(C\) defines the trade-off between these two terms.

Considering that our setting is a dynamic one (as evolution proceeds toward the Pareto front), no algorithm with theoretical guarantees is available, and some heuristic adaptation of the above MAB algorithm is used:

1. The average reward of an arm (a parent) is replaced by its average reward \(\hat{q}_{i,\tau}\) along a time window of size \(w\);
6.2 Reward-based Parent Selection

Algorithm 6.5: ParentSelection

1: given \(Q^t, k\);
2: if SelectionScheme = \((\mu + \mu)\) then
3: \(i_k \leftarrow k\);
4: end if
5: if SelectionScheme = \((\mu + 1)\) then
6: \(i_k \leftarrow U(1, |Q^t|)\);
7: end if
8: if SelectionScheme = \((\mu_{<} + 1)\) then
9: \(i_k \leftarrow U(1, \text{ndom}(Q^t))\);
10: end if
11: if SelectionScheme = \((\mu + \text{tsize} 1)\) then
12: \(i_k \leftarrow U(1, |Q^t|)\);
13: for \(k = 2, \ldots, \text{tsize}\) do
14: \(i_{\text{cur}} \leftarrow U(1, |Q^t|)\);
15: if \(i_{\text{cur}} < i_k\) then
16: \(i_k \leftarrow i_{\text{cur}}\);
17: end if
18: end for
19: end if
20: if SelectionScheme = \((\mu + 1_{\text{suc}})or (\mu + 1_{\text{rank}})or (\mu + 1_{\Delta H})or (\mu + 1_{\Delta H})\) then
21: \(i_k \leftarrow \arg\max_i (\hat{q}_{i_k,t})\);
22: end if
23: return \(i_k\);

2. The exploration is enforced by selecting once every arm which i). occurs only once in the time window, and ii). is about to disappear from the time window (it was selected \(w\) time steps ago);

3. In all other cases, the selection is on the exploitation side, and the arm with best average reward along the last \(w\) time steps is selected. The constant factor \(C\) of Eq. (6.2) is set to 0.

In summary, the MAB-like selection always selects the parent with best average reward in the last \(w\) time steps, except for case 2 (a current parent is about to disappear from the time window). Parameter \(w\) thus controls the exploration strength of the selection. Experimentally however, the sensitivity of the algorithm w.r.t. \(w\) seems to be rather low, and \(w\) was set to 500 in all experiments (Section 6.2.2).

Figure 6.9 illustrates simple steps of MAB-based parent selection.

6.2.1.3 Defining Rewards

This Section describes the rewards underlying the MAB-like selection mechanism (Algorithm 6.6). A key related issue is how to share the reward between parents and offspring.
Algorithm 6.6: ComputeRewards

1: given $Q^t$, $Q^{t+1}$, $i_{\text{parent}}^{t+1}$, $i_{\text{parent}}^{t}$, $i_{\text{offspring}}^{t+1}$, $i_{\text{offspring}}^{t}$, $W$
2: $r^t \leftarrow 0$
3: if $i_{\text{offspring}}^{t+1} \leq \mu$ then
4:   // offspring will be inserted in population
5:   if $\text{SelectionScheme} = (\mu + 1_{\text{succ}})$ then
6:      $r^t \leftarrow 1.0$
7:   end if
8:   if $\text{SelectionScheme} = (\mu + 1_{\text{rank}})$ then
9:      $r^t \leftarrow 1.0 - \frac{i_{\text{offspring}}^{t+1}}{\mu}$
10: end if
11: if $\text{SelectionScheme} = (\mu + 1_{\Delta H_1})$ or $(\mu + 1_{\Delta H_i})$ then
12:   $F^{t+1} \leftarrow \text{NomDomFronts}(Q^{t+1})$
13:   $F^t \leftarrow \text{NomDomFronts}(Q^t)$
14:   for $i = 1, \ldots, \min(|F^{t+1}|, |F^t|)$ do
15:      $H_i \leftarrow \sum_{a_{t+1} \in F^{t+1}_i} \Delta H(a_{t+1}, F^{t+1}_i) - \sum_{a_t \in F^t_i} \Delta H(a_t, F^t_i)$
16:   end for
17:   if $\text{SelectionScheme} = (\mu + 1_{\Delta H_1})$ then
18:      $r^t \leftarrow H_1$
19:   end if
20:   if $\text{SelectionScheme} = (\mu + 1_{\Delta H_i})$ then
21:      $r^t \leftarrow \left\{ \frac{H_i}{|H_i|} \arg\min_i (H_i > 0), 1 \leq i \leq |H_i| \right\}$
22:   end if
23: end if
24: $r_{i_{\text{offspring}}^{t+1}}^{\ast} \leftarrow r_{i_{\text{offspring}}^{t+1}}^{\ast} \cup \{r^t\}$
25: end if
26: if $i_{\text{parent}}^{t+1} \leq \mu$ then
27:   // parent remains in population
28:   $r_{i_{\text{parent}}^{t+1}}^{\ast} \leftarrow r_{i_{\text{parent}}^{t+1}}^{\ast} \cup \{r^t\}$
29: end if
30: for $i = 1, \ldots, \mu$ do
31:   if $r^t - W \in r_i^{\ast}$ then
32:      $r_i^{\ast} \leftarrow r_i^{\ast} \setminus \left\{r^t - W\right\}$
33:   end if
34:   $n_i^{\ast, t} \leftarrow |r_i^{\ast}|$
35:   if $n_i^{\ast, t} = 0$ then
36:      $n_i^{\ast, t} \leftarrow 1$
37:      $q_i^{\ast, t} \leftarrow 1.0$
38:   else
39:      $q_i^{\ast, t} \leftarrow \frac{1}{n_i^{\ast, t}} \sum_{r \in r_i^{\ast}} r$
40:   end if
41: end for
On the one hand, if an offspring survives, it is better that some old parents and might thus be a good starting point for further advances toward the Pareto front. The offspring must thus inherit a sufficient fraction of its parent reward, to enable its exploitation. On the other hand, the reward of a parent should be high when it yields good-performing offspring, and in particular no reward should be awarded to the parent if the newborn offspring does not survive. Several reward indicators have been considered.

\begin{equation}
A \text{ first possibility is to consider boolean rewards. If an offspring makes it to the next generation, both the offspring and the parent receive reward 1 (line 6 of Algorithm 6.6). Formally:}
\end{equation}

\begin{equation}
r_t^\tau = 1.0 \text{ if } a_{t+1}^u \in Q^{t+1}
\end{equation}

Such boolean rewards entail a very "greedy" behavior. The newborn offspring, receiving 1 as instant reward, gets 1 as average reward over the time window; it will thus very rapidly (if not immediately) be selected at next parent. Likewise, its parent which already had a top average reward (it was selected), will improve its average reward and tend to be selected again.

\begin{equation}
A \text{ smoother reward is defined by taking into account the rank of the newly inserted offspring (line 9):}
\end{equation}

\begin{equation}
r_t^\tau = 1.0 - \frac{\text{rank}(a_{t+1}^u) - 1}{\mu} \text{ if } a_{t+1}^u \in Q^{t+1}
\end{equation}

where \text{rank}(a_{t+1}^u) \text{ is the rank of the newly inserted offspring in population } Q^{t+1} \text{ (using comparison operator } \prec_{Q^t} \text{ defined by Eq. (2.9); the top individual gets rank 0). Along this line, the reward ranges linearly from 1 (for a non-dominated individual with best hypervolume contribution) to 0. A newborn offspring will here be selected rapidly only if

\textbf{Figure 6.9: Multi-Armed Bandits-based parent selection.}
it makes it into the top-ranked individuals of the current population. The average reward of the parent can decrease if its offspring gets a poor rank, even if the offspring survives.

Another way of getting smooth rewards is based on the hypervolume contribution of the offspring. Let us set the reward to 0 for dominated offspring (noting that most individuals are non-dominated at the end of evolution); for non-dominated offspring, one sets the reward to the increase of the total hypervolume contribution from generation $t$ to $t + 1$ (line 18):

$$r^T_t = \sum_{a \in Q^{t+1}} \Delta H(a, Q^t) - \sum_{a \in Q^t} \Delta H(a, Q^t)$$

In the early stages of evolution, many offspring are dominated and the above hypervolume-based reward thus gives little information. A relaxation of the above reward, involving a rank-based penalization is thus defined. Formally, if $k$ denote the Pareto rank of the current offspring, the reward is (line 21):

$$r^T_t = \frac{1}{2^{k-1}} \left( \sum_{a \in ndom_k(Q^{t+1})} \Delta H(a, ndom_k(Q^{t+1})) - \sum_{a \in ndom_k(Q^t)} \Delta H(a, ndom_k(Q^t)) \right)$$

After the reward $r^T_t$ is estimated using one of the described above strategies, it should be added to a list of rewards $r^*_{t \text{offspring}}$ of offspring $i_{t \text{offspring}}$ (line 24) and to a list of rewards $r^*_{t+1 \text{parent}}$ of parent $i_{t+1 \text{parent}}$. Of course, adding the reward makes sense only if the parent survives ($i_{t+1 \text{parent}} \leq \mu$) and/or offspring survives ($i_{t+1 \text{offspring}} \leq \mu$). Note that the newly created offspring inherits its parent’s list of rewards. For each individual $i$ the list of rewards $r^*_i$ should be shrunken to a size of time window $W$ (line 32). If the new size $n_{i,t}$ (line 34) of the list is 0 (i.e., $i$-th individual was checked last time at least $W$ time steps ago), then we reward $i$-th individual by 1 (lines 36-37) to be very likely selected in the next generation. If the list of rewards is not null, then the average reward $\tilde{q}_{i,t}$ is computed in line 39 to be used later in parent selection in line 21 of Algorithm 6.5.

6.2.1.4 Discussion

The difficulty of associating a reward to a pair (parent, offspring) in Multi-Objective optimization is twofold. On the one hand, defining absolute indicators (e.g., reflecting some aggregation of the objective values) goes against the very spirit of MO. On the other hand, relative indicators such as above-defined must be taken with care: they give a snapshot of the current situation, which evolves along the population progresses toward the Pareto front. The well-founded Multi-Armed Bandit setting, and its trade-off between exploration and exploitation, must thus be modified to account for non-stationarity.

Another difficulty is related to the finiteness of the population: while new arms appear, some old arms must disappear. The parent selection, e.g., based on the standard deterministic selection ($\prec Q^t$) is biased toward exploitation as it does not offer any way of “cooling
6.2 Reward-based Parent Selection

Figure 6.10: Reward-based multi-objective optimization with bounded population. "#6:+0.4" means reward 0.4 on 6th iteration.

down" the process. Such a bias is illustrated in Figure 6.10. Let the population size of a steady-state EMOA be 5, and consider a sequence of 10 evolution steps, generating 10 new points oldest (respectively, newest) points are black (respectively, white). At each iteration the parent with best reward generates an offspring, then 6 points are compared using $\preceq_{Q^t}$, and the worst point (crossed out) is eliminated. The instant parent reward reflects the quality of the offspring. Along evolution, some prospective points/arms are thus eliminated because they progress more slowly than others, although they do progress, due to the fixed population size. Expectedly, this bias toward exploitation adversely affects the discovery of multi-modal and/or disconnected Pareto front. We will also discuss this issue in Section 6.2.3.

6.2.2 Experimental Validation

This Section reports the experimental validation of the proposed schemes, comparatively to the baseline MO-CMA-ES algorithms, detailing the experimental setting in Section 6.2.2.1 before discussing the experimental evidence in Section 6.2.2.3.

6.2.2.1 Experimental Setting

Algorithms.

The experiments involve:

- The steady-state MO-CMA-ES with tournament-based parent selection, where the tournament size $t_{size}$ is set to 2 or 10 (respectively noted $(\mu + 2 \: 1)$ and $(\mu + 10 \: 1)$);

- The steady-state MO-CMA-ES with MAB-based parent selection, considering the four rewards described in Section 6.2.1.2 (respectively noted $(\mu + 1_{succ})$, $(\mu + 1_{rank})$, $(\mu + 1_{\Delta H_1})$ and $(\mu + 1_{\Delta H_i})$);
Chapter 6. Exploring new frontiers using CMA-like algorithms

Figure 6.11: MO-CMA-ES on 30-dimensional (Left) ZDT1 problem and (Right) Shifted ZDT1 (sZDT1) problem for different settings of the penalty parameter $\alpha$.

- The baseline algorithms include the generational ($\mu + \mu$)-MO-CMA and its steady-state variants ($\mu + 1$)-MO-CMA and ($\mu < 1$)-MO-CMA (see Section 2.4.4).

All parameters of MO-CMA-ES are set to their default values [Igel et al., 2007b] (in particular, $\mu = 100$); all algorithms only differ by their parent selection procedure. All reported results are based on 31 independent runs with at most 200,000 fitness evaluations. Median results are reported when the target precision was reached.

Problems.

The well-known bi-criteria ZDT1:3-6 and their rotated variants IHR1:3-6 have been considered. Note however that the true Pareto front of all ZDT problems lies on the boundary of the decision space, which might make it easier to discover. For the sake of an unbiased assessment, the true Pareto front is thus shifted in decision space: $x'_i \leftarrow |x_i - 0.5|$ for $2 \leq i \leq n$, where $n$ is the problem dimension. The shifted ZDT problems are denoted sZDT.

The set of recently proposed benchmark problems LZ09-1:5 [Li and Zhang, 2009] have also been used for their complicated Pareto front in decision space (Figure 6.14).

The performance assessment procedure described in Section 2.4.5 is used.

6.2.2.2 Constraints Handling in MO-CMA-ES

In the original paper on MO-CMA-ES ([Igel et al., 2007b]) the following box-constraints handling procedure was used. Consider an optimization problem with $m$ objectives $f_1, \ldots, f_m : X \mapsto \mathbb{R}$ with $X = [x_1^l, x_1^u] \times \ldots \times [x_n^l, x_n^u] \subset \mathbb{R}^n$. For $x \in \mathbb{R}^n$ let $\text{feasible}(x) = (\min(\max(x_1, x_1^l), x_1^u), \ldots, \min(\max(x_n, x_n^l), x_n^u))^T$. The penalized fitness is defined as follows:

$$f_{m\text{penalty}}(x) = f_m(\text{feasible}(x)) + \alpha \| x - \text{feasible}(x) \|_2^2,$$  \hspace{1cm} (6.3)

where $\alpha > 0$ is a penalty parameter. [Igel et al., 2007b] suggest to set $\alpha = 10^{-6}$.
In order to investigate the performance of MO-CMA-ES depending on the choice of $\alpha$ we run the algorithm on ZDT1 problem. The results shown in Figure 6.11-Left confirms that MO-CMA-ES is very sensitive to the value of $\alpha$. MO-CMA-ES demonstrates almost identical behaviour for $\alpha = 10^{-6}$ and $\alpha = 0$ which can be interpreted as follows. (1+1)-CMA-ES algorithm of the initial population relatively often samples offspring outside of the box-constraint due to a relatively large step-size $\sigma$. The objective values of these offspring are computed and penalized by a small location-based value proportional to $\alpha$. For a particular case of ZDT problems, where the Pareto front lies on the boundary of the decision space, each box-outlier $x_i$ translates back in the feasible region, i.e., directly on the Pareto front for $x_1$ (given that ZDT problems are separable). In other words, MO-CMA-ES samples points outside of the box and translates them back on the Pareto front without significant penalization. This process can be clearly seen for small $\alpha$ in Figure 6.11-Left. In order to make penalization "more realistic" we may increase $\alpha$, and as can be seen, it significantly worsen the results. Roughly speaking, there are different types of constraint handling techniques, depending on whether the constraints tend to "repel" candidate solutions (that corresponds to large $\alpha$), or to "attract" candidate solutions (that corresponds to small $\alpha$). We argue that a reasonable value for $\alpha$ for ZDT1 problem ($\alpha$ is problem-dependent) should be in the order of 1 rather than in order of $10^{-6}$. Indeed, in this case, some trade-off of "repel"/”attract" effects can be achieved such that the influence of both is relatively negligible and the solution of ZDT1 can be found almost as fast as the one of sZDT1 (a shifted variant of ZDT1). This is visible when looking at the evolution of the hypervolume indicator for MO-CMA-ES with $\alpha = 1$ on ZDT1 and MO-CMA-ES with different $\alpha$ on sZDT1 at the point of 40,000 functions evaluations on Figure 6.11-Right. As can be seen, the results of MO-CMA-ES on shifted ZDT1 function (sZDT1) (as well as on IHR functions) are insensitive to $\alpha$ . . . and the case $\alpha = 1$ for ZDT1 corresponds to a very similar behavior (though this is by no mean a proof).

We argue that the results of MO-CMA-ES on ZDT problems presented in [Igel et al., 2007b] and several following papers should be reviewed and/or interpreted w.r.t. the choice of $\alpha$.

Note that a better box-constraints handling may be obtained by replacing the original (1+1)-CMA-ES by the recently proposed prospective version of (1+1)-CMA-ES for constrained optimization [Arnold and Hansen, 2012], and this will be a part of our further work.

6.2.2.3 Result Analysis

All empirical results are displayed in Table 6.1. These results show that the proposed algorithms generally outperform the baseline MO-CMA-ES approaches, with the exception of problems sZDT3, IHR6 and LZ09-5. A first general remark is that the steady-state variants of MO-CMA-ES outperform the generational one on unimodal benchmark problems; as already noted in [Igel et al., 2007a], the greedier $(\mu \prec + 1)$-MO-CMA is usually faster than the original steady-state on sZDT and IHR problems; in counterpart, it is too "greedy" on LZ09 problems (see below). Another general remark is that $(\mu + 1\Delta H_1)$-MO-CMA is usually more robust and faster than $(\mu + 1\Delta H_1)$-MO-CMA; this fact is explained as
Chapter 6. Exploring new frontiers using CMA-like algorithms

Figure 6.12: On-line performances of baseline and proposed variants of steady-state MO-CMA-ES on sZDT1, IHR1, LZ09-3 and LZ09-4 problems (median out of 31 runs).

the former exploit a better informed hypervolume contribution-based reward, considering also the contribution of dominated points.

The on-line performance of most considered algorithms on sZDT1, IHR1, LZ09-3 and LZ09-4 shows the general robustness of \((\mu + 1)\text{-MO-CMA}\) (Figure 6.12, displaying \(\Delta H(P^*, P)\) versus the number of function evaluations). The comparatively disappointing results of \((\mu + 1)\text{-MO-CMA}\) on IHR1 are explained from the structure of the Pareto front, which includes an easy-to-find segment. This segment can be discovered by selecting the extreme parent (in objective space), thus with probability \(1/\mu\) within a uniform selection scheme. Quite the contrary, reward-based selection schemes quickly catch the fruitful directions of search.

The price to pay for this is depicted in Figure 6.13, showing \((\mu + 1\text{succ})\text{-MO-CMA}\)
6.2 Reward-based Parent Selection

Figure 6.13: Typical behavior of \((\mu + 1_{\text{succ}})\)-MO-CMA on sZDT2 (Left) and IHR3 (Right) problems: premature convergence after 5,000 fitness function evaluations.

on sZDT2 and IHR3 problems. On these problems, a premature convergence toward a small segment of the Pareto front is observed after circa 5,000 function evaluations. The interpretation provided for this premature convergence goes as follows. As one part of the Pareto front is easier to find than others, points aiming at this part quickly reach their goal; due to non-dominated sorting (and to the fixed population size), these eliminate other points, resulting in a very poor diversity (in objective space) of the population. This remark suggests that some additional diversity preserving technique should be used together with MO-CMA-ES; note that, even in the original MO-CMA-ES, a premature convergence is observed on IHR3.

LZ09 problems have also been considered because of their non-linear Pareto front in decision space (Figure 6.14), contrasting with the linear Pareto front of all sZDT and IHR problems. The results depicted in Figure 6.14 show that \((\mu + 1_{\text{rank}})\)-MO-CMA better approximates the Pareto front than \((\mu + 1)\) and \((\mu + 1_{10})\)-MO-CMA, for all problems except LZ09-5. It is interesting to note that the results of \((\mu + 1_{\text{rank}})\)-MO-CMA after 100,000 fitness evaluations match those of MOEA/D-DE after 150,000 fitness evaluations [Li and Zhang, 2009].

Overall (see also Table 6.1), \((\mu + 1_{\text{rank}})\)-MO-CMA and \((\mu + 1_{10})\)-MO-CMA perform best on most problems, while \((\mu + 1_{\Delta H_i})\)-MO-CMA is slightly more robust. Most generally, all “greedy” versions of MO-CMA-ES get better results on problems with a convex Pareto front; on problems with a concave or disconnected Pareto front, they suffer from premature convergence, entailed by a loss of diversity, due to non-dominated sorting and bounded population size.

6.2.3 Conclusion and Perspectives

The goal and main contribution of the Section is to speed up MO-CMA-ES using new parent selection schemes, based on tournament and reward-based approaches inspired from the Multi-Armed Bandit framework, in order to quickly identify the most fruitful directions.
Table 6.1: Comparative results of two baseline EMOAs, namely generational and steady-state MO-CMA-ES and several versions of steady-state MO-CMA-ES with different parent selection schemes. Median number of function evaluations (out of 31 independent runs) to reach $\Delta H_{\text{target}}$ values, normalized by Best: a value of 1 indicates the best result, a value $X > 1$ indicates that the corresponding algorithm needed $X$ times more evaluations than the best to reach the same precision.

<table>
<thead>
<tr>
<th>$\Delta H_{\text{target}}$</th>
<th>1</th>
<th>0.1</th>
<th>0.01</th>
<th>1</th>
<th>0.1</th>
<th>0.01</th>
<th>1</th>
<th>0.1</th>
<th>0.01</th>
<th>1</th>
<th>0.1</th>
<th>0.01</th>
</tr>
</thead>
<tbody>
<tr>
<td>Best</td>
<td>2500</td>
<td>12000</td>
<td>47000</td>
<td>2500</td>
<td>15000</td>
<td>59000</td>
<td>3000</td>
<td>18500</td>
<td>70000</td>
<td>4500</td>
<td>141200</td>
<td></td>
</tr>
<tr>
<td>($\mu + \mu$)</td>
<td>7.3</td>
<td>4.3</td>
<td>2.2</td>
<td>8.6</td>
<td>4.3</td>
<td>2.1</td>
<td>5.5</td>
<td>3</td>
<td>1.9</td>
<td>8.2</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>($\mu + 1$)</td>
<td>6.5</td>
<td>3.9</td>
<td>2.1</td>
<td>7.6</td>
<td>3.9</td>
<td>2</td>
<td>5.1</td>
<td>2.8</td>
<td>1.4</td>
<td>7.2</td>
<td>1.1</td>
<td></td>
</tr>
<tr>
<td>($\mu + \mu + 1$)</td>
<td>1.5</td>
<td>2.2</td>
<td>1.7</td>
<td>1</td>
<td>2</td>
<td>1.5</td>
<td>1.3</td>
<td>1.8</td>
<td>1.2</td>
<td>1</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>($\mu + 2$)</td>
<td>3.7</td>
<td>2.4</td>
<td>1.5</td>
<td>1</td>
<td>3</td>
<td>2.4</td>
<td>1.4</td>
<td>3.1</td>
<td>1.7</td>
<td>1</td>
<td>4.3</td>
<td></td>
</tr>
<tr>
<td>($\mu + 10$)</td>
<td>1.2</td>
<td>1</td>
<td>1.1</td>
<td>1.4</td>
<td>1</td>
<td>1.3</td>
<td>1.1</td>
<td>1</td>
<td>1</td>
<td>1.3</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>($\mu + 1 \Delta n_i$)</td>
<td>3.5</td>
<td>1.5</td>
<td>1</td>
<td>3.4</td>
<td>1.6</td>
<td>1</td>
<td>2.5</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>($\mu + 1 \Delta n_{\mu}$)</td>
<td>1.7</td>
<td>1.3</td>
<td>1</td>
<td>1.8</td>
<td>1.3</td>
<td>1</td>
<td>1.1</td>
<td>.</td>
<td>.</td>
<td>1.5</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>($\mu + 1 \Delta n_{\mu}$)</td>
<td>1.2</td>
<td>1.7</td>
<td>1.1</td>
<td>1.6</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>2.1</td>
<td>.</td>
<td></td>
</tr>
<tr>
<td>($\mu + 1 \Delta n_{\mu}$)</td>
<td>1</td>
<td>1.4</td>
<td>1</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>1</td>
<td>.</td>
<td>.</td>
<td>1.7</td>
<td>.</td>
<td></td>
</tr>
</tbody>
</table>

| Best                     | 500 | 1500 | 6000 | 1500 | 4000 | 8500 | 1000 | . | . | 6000 | . | . |
| IHR1                     | . | . | . | . | . | . | . | . | . | . | . | . |
| IHR2                     | . | . | . | . | . | . | . | . | . | . | . | . |
| IHR3                     | . | . | . | . | . | . | . | . | . | . | . | . |
| IHR6                     | . | . | . | . | . | . | . | . | . | . | . | . |

| Best                     | 500 | 1500 | 17000 | 3500 | 144000 | 1500 | 35000 | 120500 | 1000 | 10000 | 40500 | 57 | 3.2 | 2.4 |
| IZ09-1                   | . | . | . | . | . | . | . | . | . | . | . | . |
| IZ09-2                   | . | . | . | . | . | . | . | . | . | . | . | . |
| IZ09-3                   | . | . | . | . | . | . | . | . | . | . | . | . |
| IZ09-4                   | . | . | . | . | . | . | . | . | . | . | . | . |

| Best                     | 1500 | 19000 | . | . | . | . | . | . | . | . | . | . |
| IZ09-5                   | . | . | . | . | . | . | . | . | . | . | . | . |

IHR1 - Incremental Hillclimber Rosenbrock 1
IHR2 - Incremental Hillclimber Rosenbrock 2
IHR3 - Incremental Hillclimber Rosenbrock 3
IHR6 - Incremental Hillclimber Rosenbrock 6
IZ09-1 - Improved ZDT1
IZ09-2 - Improved ZDT2
IZ09-3 - Improved ZDT3
IZ09-4 - Improved ZDT4
IZ09-5 - Improved ZDT5
IZ09-6 - Improved ZDT6

182
Figure 6.14: Plots of all 10 populations found by $(\mu + 1)$-MO-CMA (Left), $(\mu + 10)$-MO-CMA (Center) and $(\mu + 1_{rank})$-MO-CMA (Right) in the $x_1 - x_2 - x_3$ space on LZ09-1:5 after 100,000 function evaluations.
Chapter 6. Exploring new frontiers using CMA-like algorithms

of search. Experiments on several bi-objective problems have shown a significantly speedup of MO-CMA-ES on unimodal problems (for both generational and previous steady-state variants). However, the proposed approach results in a poor convergence on multi-modal multi-objective problems, or problems where some parts of the Pareto front are much easier to reach than others, such as IHR3 (Figure 6.13, and discussion in Sections 6.2.1.4 and 5.4.3).

These remarks open some perspectives for further research, aimed at preserving the benefits of parent selection schemes while addressing the premature convergence on multi-modal landscapes. A first perspective is to maintain the points located at the border of the already visited region, and to give them some chance to produce offspring as well although they are dominated. The question thus becomes to handle yet another exploration vs exploitation dilemma, and distribute the fitness evaluations between the current population and the borderline points; it also remains to extend the reward definition for the borderline points. Such an approach is similar in spirit to the so-called BIPOP-CMA-ES designed to overcome premature convergence within single-objective evolutionary optimization [Hansen et al., 2010b]; BIPOP-CMA-ES proceeds by maintaining one large population for exploration purposes, and a small one for fast and accurate convergence, for exploitation purposes.

A second perspective is to design a more integrated multi-objective CMA-ES based algorithm, by linking the reward mechanism used in the parent selection and the internal update rules of CMA-ES. Indeed, the success rate used to control the (1 + 1)-ES evolution and the empirical success expectation used in \((\mu + 1_{\text{succ}})\)-MO-CMA are strongly related. Further work will consider how to use the success rate in lieu of reward for parent selection, expectedly resulting in a more consistent evolution progress. Meanwhile, the CMA update rules might want to consider the discarded offspring (possibly weighting their contribution depending on their hypervolume contribution), since they might contain useful information even though they are discarded.

Last but not least, the MO-CMA-ES and the proposed parent selection schemes must be analyzed and compared with other state-of-the art MOEAs, specifically SMS-EMOA [Emmerich et al., 2005], the first algorithm, to our best knowledge, to advocate the use of steady state within EMO; it also proposed separable variation operators, resulting in excellent results comparatively to MO-CMA-ES on separable problems. How to extend these variation operators in the non-separable case, following ideas of Adaptive Encoding will also be investigated.

6.3 Alternative Restart Strategies for CMA-ES

This Section focuses on analyzing and improving the restart strategy of CMA-ES (IPOP and BIPOP), viewed as a noisy hyper-parameter optimization problem in a 2-dimensional space (population size, initial step-size). Two restart strategies are defined. The first one, NIPOP-aCMA-ES (for \textit{New IPOP-aCMA-ES}), differs from IPOP-CMA-ES as it simultaneously increases the population size and decreases the step size. The second one, NBIPOP-aCMA-ES, allocates computational power to different restart settings de-
pending on their current results. While these strategies have been designed with the BBOB benchmarks in mind [Hansen et al., 2012], their generality is also demonstrated on an independent real-world problem suite related to spacecraft trajectory optimization [Vinko and Izzo, 2008].

The Section is organized as follows. The original restart strategies and their analysis are presented in Sections 6.3.1 and 6.3.2, respectively. The proposed restart schemes are described in Section 6.3.3. Section 6.3.4 reports on their experimental validation. The Section concludes with a discussion and some perspectives for further research.

6.3.1 The CMA-ES with Restarts: IPOP and BIPOP CMA-ES

CMA-ES has been extended with restart strategies to accommodate multi-modal fitness landscapes, and to specifically handle objective functions with many local optima. As observed by [Hansen and Kern, 2004], the probability of reaching the optimum (and the overall number of function evaluations needed to do so) is very sensitive to the population size. The default population size $\lambda_{\text{default}}$ has been tuned for unimodal functions; it is hardly large enough for multi-modal functions. Accordingly, [Auger and Hansen, 2005] proposed a “doubling trick” restart strategy to enforce global search: the restart ($\mu/\mu_w, \lambda$)-CMA-ES with increasing population, called IPOP-CMA-ES, is a multi-restart strategy where the population size of the run is doubled at each restart until meeting a stopping criterion.

The BIPOP-CMA-ES instead considers two restart regimes. The first one, which corresponds to IPOP-CMA-ES, doubles the population size at every restart $i_{\text{restart}}$ ($\lambda_{\text{large}} = 2^{i_{\text{restart}}} \lambda_{\text{default}}$) and uses an initial step-size $\sigma_{\text{large}}^{0} = \sigma_{\text{default}}^{0}$. The second regime uses a small population size $\lambda_{\text{small}}$ and initial step-size $\sigma_{\text{small}}^{0}$, which are randomly drawn in each restart as:

$$\lambda_{\text{small}} = \left[ \lambda_{\text{default}} \left( \frac{1}{2} \lambda_{\text{large}} \right)^{\frac{1}{2}} U[0,1] \right]^{2}, \quad \sigma_{\text{small}}^{0} = \sigma_{\text{default}}^{0} \times 10^{-2} U[0,1]$$  \hspace{1cm} (6.4)

where $U[0,1]$ stands for the uniform distribution in $[0,1]$. Population size $\lambda_{\text{small}}$ thus varies in $[\lambda_{\text{default}}, \lambda_{\text{large}}/2]$. BIPOP-CMA-ES launches the first run with default population size and initial step-size. In each restart, it selects the restart regime with less function evaluations used so far. Clearly, the second regime consumes less function evaluations than the doubling regime; it is therefore launched more often.

6.3.2 Preliminary Analysis

The restart strategies of IPOP- and BIPOP-CMA-ES can be viewed as a search in the hyper-parameter space.

IPOP-CMA-ES only aims at adjusting population size $\lambda$. It is motivated by the results observed on multi-modal problems [Hansen and Kern, 2004], suggesting that the population size must be sufficiently large to handle problems with global structure. In such cases, a large population size is needed to uncover this global structure and to lead the algorithm to discover the global optimum. IPOP-CMA-ES thus increases the population size in each

\hspace{1cm}
Figure 6.15: Restart performances in the 2-dimensional hyper-parameter space (population size and initial mutation step size in log. coordinates). For each 20-dimensional objective function (Rastrigin - Top-Left, Gallagher 21 peaks - Top-Right, Katsuuras - Bottom-Left and Lunacek bi-Rastrigin Bottom-Right), the median best function value out of 15 runs is indicated. Legends indicate that the optimum of the objective function up to precision $f(x) = 10^{-10}$ is found always (+), at least once ($\oplus$) or never (○). Black regions are better than white ones.

restart, irrespective of the results observed so far; at each restart, it launches a new CMA-ES with population size $\lambda = \rho_{\text{inc}}^{\text{restart}} \lambda_{\text{default}}$ (see ○ in Figure 6.15). Factor $\rho_{\text{inc}}$ must be not too large to avoid "overjumping" some possibly optimal population size $\lambda^*$; it must also be not too small in order to reach $\lambda^*$ in a reasonable number of restarts. The use of the doubling trick ($\rho_{\text{inc}} = 2$) guarantees that the loss in terms of function evaluations (compared to the "oracle" restart strategy which would directly set the population size to the optimal value $\lambda^*$) is at most about a factor of 2.

On the Rastrigin 20-D function, IPOP-CMA-ES performs well and always finds the optimum after about 5 restarts (Figure 6.15, Top-Left). The Rastrigin function displays indeed a global structure where the optimum is the minimizer of this structure. For such functions, IPOP-CMA-ES certainly is the method of choice. For some other functions such as the Gallagher function, there is no such global structure; increasing the population size
6.3 Alternative Restart Strategies for CMA-ES

Figure 6.16: An illustration of $\lambda$ and $\sigma$ hyper-parameters distribution for 9 restarts of IPOP-aCMA-ES ($\circ$), BIPOP-aCMA-ES ($\circ$ and $\cdot$ for 10 runs), NIPOP-aCMA-ES ($\Box$) and NBIPOP-aCMA-ES ($\Box$ and many $\triangle$ for $\lambda/\lambda_{\text{default}} = 1$, $\sigma/\sigma_{\text{default}} \in [10^{-2}, 10^0]$). The first run of all algorithms corresponds to the point with $\lambda/\lambda_{\text{default}} = 1$, $\sigma/\sigma_{\text{default}} = 1$.

does not improve the results. On Katsuuras and Lunacek bi-Rastrigin functions, the optimum can only be found with small initial step-size (less than the default one); this explains why it can be solved by BIPOP-CMA-ES, sampling the two-dimensional ($\lambda, \sigma$) space.

Actually, the optimization of a multi-modal function by CMA-ES with restarts can be viewed as the optimization of the function $h(\theta)$, which returns the optimum found by CMA-ES defined by the hyper-parameters $\theta=(\lambda, \sigma)$. Function $h(\theta)$, graphically depicted in Figure 6.15 can be viewed as a black box, computationally expensive and stochastic function (reflecting the stochasticity of CMA-ES). Both IPOP-CMA-ES and BIPOP-CMA-ES are based on implicit assumptions about $h(\theta)$: IPOP-CMA-ES achieves a deterministic uni-dimensional trajectory, and BIPOP-CMA-ES randomly samples the 2-dimensional search space.

Function $h(\theta)$ also can be viewed as a multi-objective fitness, since in addition to the solution found by CMA-ES, $h(\theta)$ could return the number of function evaluations needed to find that solution. $h(\theta)$ could also return the computational effort SP1 (i.e., the average number of function evaluations of all successful runs, divided by proportion of successful runs). However, SP1 can only be known for benchmark problems where the optimum is known; as the empirical optimum is used in lieu of true optimum, SP1 can only be computed \textit{a posteriori}.

6.3.3 Alternative Restart Strategies

Two new restart strategies for CMA-ES, respectively referred to as NIPOP-aCMA-ES and NBIPOP-aCMA-ES, are presented.

If the restart strategy is restricted to the case of increasing population size (IPOP),
we propose to use NIPOP-aCMA-ES, where we additionally decrease the initial step-size by some factor $\rho_{\sigma \text{dec}}$. The rationale behind this approach is that the CMA-ES with relatively small initial step-size is able to explore small basins of attraction (see Katsuuras and Lunacek bi-Rastrigin functions in Figure 6.15), while with initially large step-size and population size, it will neglect the local structure of the function, but converge to the minimizer of the global structure. Moreover, initially, relatively small step-size will quickly increase if it makes sense, and this will allow the algorithm to recover the same global search properties than with initially large step-size (see Rastrigin function in Figure 6.15).

NIPOP-aCMA-ES thus explores the two-dimensional hyper-parameter space in a deterministic way (see □ symbols in Figure 6.16). For $\rho_{\sigma \text{dec}} = 1.6$ used in this study, NIPOP-aCMA-ES thus reaches the lower bound ($\sigma = 10^{-2}\sigma_{\text{default}}$) used by BIPOP-CMA-ES after 9 restarts, expectedly reaching the same performance as BIPOP-CMA-ES albeit using only a large population.

The second restart strategy, NBIPOP-aCMA-ES, addresses the case where the probability to find the global optimum does not much vary in the ($\lambda, \sigma$) space. Under this assumption, it makes sense to have many restarts for a fixed budget (number of function evaluations). Specifically, NBIPOP-aCMA-ES implements the competition of the NIPOP-aCMA-ES strategy (increasing $\lambda$ and decreasing initial $\sigma^0$ in each restart) and a uniform sampling of the $\sigma$ space, where $\lambda$ is set to $\lambda_{\text{default}}$ and $\sigma^0 = \sigma_{\text{default}}^0 \times 10^{-2U[0,1]}$ The selection between both regimes (NIPOP-aCMA-ES and the uniform sampling) depends on the allowed budget, like in BIPOP-aCMA-ES. The difference is that NBIPOP-aCMA-ES adaptively sets the budget allowed to each restart strategy, where the restart strategy leading to the overall best solution found so far is allowed twice ($\rho_{\text{budget}} = 2$) the budget given to the other strategy.

6.3.4 Experimental Validation

The experimental validation of NIPOP-aCMA-ES and NBIPOP-aCMA-ES investigates the performance of the approach comparatively to IPOP-aCMA-ES and BIPOP-aCMA-ES on BBOB noiseless problems and one black-box real-world problem related to spacecraft trajectory optimization. The default parameters of CMA-ES [Hansen and Ros, 2010a, Hansen, 2009] are used. This Section also presents the first experimental study of BIPOP-aCMA-ES$^2$, the active version of BIPOP-CMA-ES [Hansen, 2009].

6.3.4.1 Performance on Rastrigin Function

Figure 6.17 shows one run of NIPOP-aCMA-ES on 20-dimensional Rastrigin function, where 6 restart were performed before reaching the target objective function (cyan curve) value $10^{-8}$. It should be noted that NIPOP-aCMA-ES finds the optimum even with each restart smaller and smaller initial step-sizes (green curve), and it can be clearly seen that the initially small step-size quickly increases and reaches an "appropriate" value. This is

$^2$For the sake of reproducibility, the source code for NIPOP-aCMA-ES and NBIPOP-aCMA-ES is available at https://sites.google.com/site/ppsnbipop/
6.3 Alternative Restart Strategies for CMA-ES

Figure 6.17: One run of NIPOP-aCMA-ES on 20-dimensional Rastrigin function.

due to the fact that i). the multi-modality of Rastrigin function is hardly observable far away from the optimum, and ii). relatively large population reinforces the latter effect. At the same time, the performance is not surprising and corresponds to the results illustrated in Figure 6.15.

To summarize, a smaller than default initial step-size does not necessarily lead to a premature convergence, but can be adapted by CMA-ES to an "appropriate" step-size.

6.3.4.2 Benchmarking with BBOB Framework

The BBOB framework is made of 24 noiseless and 30 noisy functions (see Section 2.3.3). Only the noiseless case has been considered here. Furthermore, only the 12 multi-modal functions among these 24 noiseless functions are of interest for this study, as CMA-ES can solve the 12 other functions without any restart.

With same experimental methodology as in [Hansen et al., 2012], the results obtained on these benchmark functions are presented in Figure 6.18 and Table 6.2. The results are given for dimension 40, because the differences are larger in higher dimensions. The expected running time (ERT), used in the figures and table, depends on a given target function value, \( f_t = f_{\text{opt}} + \Delta f \). It is computed over all relevant trials as the number of function evaluations required in order to reach \( f_t \), summed over all 15 trials, and divided by the number of trials that actually reached \( f_t \) [Hansen et al., 2012].

NIPOP-aCMA-ES. On 6 out of 12 test functions (\( f_{15}, f_{16}, f_{17}, f_{18}, f_{23}, f_{24} \)) NIPOP-aCMA-ES obtains the best known results for BBOB-2009 and BBOB-2010 workshops. On \( f_{23} \) Katsuuras and \( f_{24} \) Lunacek bi-Rastrigin, NIPOP-aCMA-ES has a speedup of a factor from 2 to 3, as could have been expected. It performs unexpectedly well on \( f_{16} \) Weierstrass functions, 7 times faster than IPOP-aCMA-ES and almost 3 times faster than BIPOP-aCMA-ES. Overall, according to Figure 6.18, NIPOP-aCMA-ES performs as well as BIPOP-aCMA-ES, while restricted to only one regime of increasing population size.
weakly structured multi-modal functions

Figure 6.18: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{-8.2}$ for all functions and weakly structured multi-modal subgroup in 40-D. The “best 2009” line corresponds to the best ERT observed during BBOB 2009 for each single target.

**NBPOP-aCMA-ES.** Thanks to the first regime of increasing population size, NBPOP-aCMA-ES inherits some results of NIPOP-aCMA-ES. However, on functions where the population size does not play any important role, it performs significantly better than BIPOP-aCMA-ES. This is the case for $f_{21}$ Gallagher 101 peaks and $f_{22}$ Gallagher 21 peaks functions, where NBPOP-aCMA-ES has a speedup of a factor of 6. It seems that the adaptive choice between two regimes works efficiently on all functions except on $f_{16}$ Weierstrass. In this last case, NBPOP-aCMA-ES mistakingly prefers small populations, with a loss factor 4 compared to NIPOP-aCMA-ES. According to Figure 6.18, NBPOP-aCMA-ES performs better than BIPOP-aCMA-ES on weakly structured multi-modal functions, showing overall best results for BBOB-2009 and BBOB-2010 workshops in dimensions 20 (results not shown here) and 40.

### 6.3.4.3 Interplanetary Trajectory Optimization

The NIPOP-aCMA-ES and NBPOP-aCMA-ES strategies, designed for the BBOB benchmark functions, can possibly *overfit* this benchmark suite. In order to test the generality of these strategies, a real-world black-box problem is considered, pertaining to a completely different domain: Advanced Concepts Team of European Space Agency is making available several difficult spacecraft trajectory optimization problems as black box functions to invite the operational research community to compare different derivative-free solvers on these test problems [Vinko and Izzo, 2008].

The following results consider the 18-dimensional bound-constrained black-box function "TandEM-Atlas501", that defines an interplanetary trajectory to Saturn from the Earth with multiple fly-bys, launched by the rocket Atlas 501. The final goal is to maximize the mass $f(x)$, which can be delivered to Saturn using one of 24 possible fly-by
sequences with possible maneuvers around Venus, Mars and Jupiter.

The first best results was found for a sequence Earth-Venus-Earth-Earth-Saturn ($f_{\text{max}} = 1533.45$) in 2008 by [Addis et al., 2008]. The best results so far ($f_{\text{max}} = 1673.88$) was found in 2011 by [Stracquadanio et al., 2011].

All versions of CMA-ES with restarts have been launched with a maximum budget of $10^8$ function evaluations. All variables are normalized in the range $[0,1]$. In the case of sampling outside of boundaries, the fitness is penalized and becomes $f(x) = f(x_{\text{feasible}}) - \alpha \|x - x_{\text{feasible}}\|^2$, where $x_{\text{feasible}}$ is the closest feasible point from point $x$ and $\alpha$ is a penalty factor, which was arbitrarily set to 1000.

As shown in Figure 6.19, the new restart strategies NIPOP-aCMA-ES and NBIPOP-aCMA-ES respectively improve on the former ones (IPOP-aCMA-ES and BIPOP-aCMA-ES); further, NIPOP-aCMA-ES reaches same performances as BIPOP-aCMA-ES.

The best solution found by NBIPOP-aCMA-ES $^3$ improves on the best solution found in 2008, while it is worse than the current best solution, which is blamed on the lack of problem specific heuristics [Addis et al., 2008, Stracquadanio et al., 2011], on the possibly insufficient time budget ($10^8$ fitness evaluations), and also on the lack of appropriate constraint handling heuristics (that might be improved using the technique proposed in [Arnold and Hansen, 2012]).

### 6.3.5 Conclusion and Perspectives

This contribution of this Section regards two new restart strategies for CMA-ES. NIPOP-aCMA-ES is a deterministic strategy simultaneously increasing the population size and decreasing the initial step-size of the Gaussian mutation. NBIPOP-aCMA-ES implements a competition between NIPOP-aCMA-ES and a random sampling of the initial mutation

---

$^3$ $x = [0.83521, 0.45092, 0.50289, 0.61389, 0.75773, 0.43376, 1, 0.89512, 0.77264, 0.11229, 0.20774, 0.018255, 6.2057e-09, 4.0371e-08, 0.2028, 0.36272, 0.32442]; \text{fitness}(x) = \text{mass}(x) = 1546.5$
step-size, adaptively adjusting the computational budget of each one depending on their current best results. Besides the extensive validation of NIPOP-aCMA-ES and NBIPOP-aCMA-ES on the BBOB benchmark, the generality of these strategies has been tested on a new problem, related to interplanetary spacecraft trajectory planning.

The main limitation of the proposed restart strategies is to quasi implement a deterministic trajectory in the $\theta$ space. Further work will consider $h(\theta)$ as yet another expensive noisy black-box function, and the use of a CMA-ES in the hyper-parameter space will be studied. The critical issue is naturally to keep the overall number of fitness evaluations beyond reasonable limits. A surrogate-based approach will be investigated, learning and exploiting an estimate of the (noisy and stochastic) $h(\theta)$ function.

### Table 6.2: Overall results on multi-modal functions $f_3 - 4$ and $f_{15} - 24$ in dimension $n = 40$: Expected running time (ERT in number of function evaluations) divided by the respective best ERT measured during BBOB-2009 for precision $\Delta f$ ranging in $10^i$, $i = 1 \ldots 7$. The median number of conducted function evaluations is additionally given in italics, if ERT($10^{-7}$) = $\infty$. #succ is the number of trials that reached the final target $f_{opt} + 10^{-8}$. Best results are printed in bold.

<table>
<thead>
<tr>
<th>$\Delta f_{opt}$</th>
<th>$f_3$</th>
<th>$f_4$</th>
<th>$f_5$</th>
<th>$f_6$</th>
<th>$f_7$</th>
<th>$#succ$</th>
<th>$\Delta f_{opt}$</th>
<th>$f_3$</th>
<th>$f_4$</th>
<th>$f_5$</th>
<th>$f_6$</th>
<th>$f_7$</th>
<th>$#succ$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1e-1</td>
<td>1e1</td>
<td>1e2</td>
<td>1e3</td>
<td>1e5</td>
<td>1e7</td>
<td></td>
<td>1e1</td>
<td>1e2</td>
<td>1e3</td>
<td>1e5</td>
<td>1e7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BIPOP-3395(2759)</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e1(1)</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
</tr>
<tr>
<td>IPPO-3395(2759)</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e1(1)</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
</tr>
<tr>
<td>NBIPPO-3395(2759)</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e1(1)</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
</tr>
<tr>
<td>NBIPPO-3395(2759)</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e1(1)</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
<td>1e0</td>
</tr>
</tbody>
</table>

#### 6.4 Other Approaches

In this Section, we very briefly describe several approaches that we investigated, but whose detailed description is omitted.
6.4 Other Approaches

6.4.1 Kernel PCA-based CMA-ES

Kernel Principal Component Analysis (Kernel PCA or KPCA) is an extension of PCA to non-linear PCA using kernel methods [Scholkopf et al., 1996]. The original PCA can be formulated in terms of dot product between data points, therefore, it can be extended to non-linear case by replacing dot product in the original space by kernel function to compute dot product in a reproduced kernel Hilbert space [Aronszajn, 1950].

We implemented a version of CMA-ES decomposed into CSA-ES and Adaptive Encoding procedure, replacing the original PCA by KPCA in the latter. Thus, instead of sampling new points on principal components in the original search space, we sample new points on principal components found in some feature space. However, finding pre-images of sampled points in the original space for their further evaluation of $f$ is not straightforward, but can be done solving so-called pre-image problem [Kwok and Tsang, 2004]. Another difficulty is to reformulate step-size adaptation rule to be adequate to metrics of the feature space, we did not find any elegant solution for this problem, and, therefore, use CSA-ES adaptation for points in the original space.

Figure 6.20 illustrates generations $k$ (Left) and $k+2$ (Right) of the KPCA-based CMA-ES on 2-dimensional Rosenbrock, where KPCA was built using 20 best (red) points found so far and tested to illustrate the sampling distribution using 1000 (blue) test points. It can be seen that KPCA clearly can catch non-linear landscape of Rosenbrock function and sample accordingly, in contrast to the original CMA-ES which would sample on 2 principal components (some ellipsoid). We suppose that the extension of CMA-ES to non-linear case, in the sense of learning non-linear principal components, is a prospective direction of research. However, for robustness of the algorithm several issues should be addressed: step-size control, pre-image problem, computational complexity. Some of these issues were already discussed in [Pošik, 2007], where a KPCA-based GA was proposed.
Figure 6.21: Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/D) for 50 targets in $10^{[-8..2]}$ for all noise-less functions in 20-D. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each single target.

### 6.4.2 High-precision BFGS

In Section 4.3.3.7, we showed that IPOP\(^{-s}\)aACM-ES and BIPOP\(^{-s}\)aACM-ES algorithms outperform BFGS on a set of 24 noise-less BBOB benchmark problems. A more detailed analysis of the results of BFGS [Ros, 2009] demonstrates that the algorithm performs quite well in the beginning of the optimization of ill-conditioned problems (5 problems out of 24), i.e., reaches target objective values $10^1$, $10^0$, $10^{-2}$ relatively quickly in comparison with other algorithms, but then slows down and hardly finds better target objective values. We suppose that this happens mainly due to a loss of precision of estimation of gradient of $f$ (and as a consequence, of Hessian of $f$) and its cumulation during the optimization. A discussion about updating BFGS formula in ill-conditioned environment can be found in [Powell, 1987].

In order to check whether the performance of BFGS degrades because of numerical/rounding problems or some other reasons, we implemented a version of BFGS with high-precision arithmetic, referred to as pBFGS. The MatLab implementation of BFGS was rewritten in C++ and integrated with a software packages for high-precision arithmetic called ARPREC [Bailey et al., 2002]. ARPREC supports arbitrary high level of numerical precision (up to approximately ten million decimal digits). In our experiments we found that double-double precision with about 32 decimal digits (double precision has about 16 digits) is sufficient to overcome numerical problems observed on BBOB benchmark problems. Having more decimal digits we may benefit from using a deviation
value $\epsilon = 10^{-12}$ (estimating the gradient as $f'(x) = \frac{f(x+\epsilon) - f(x)}{\epsilon}$), smaller than the default $\epsilon = 10^{-8}$ used for gradient estimation in the finite difference method.

Figure 6.21 shows the results of benchmarking pBFGS and BFGS in comparison with BIPOP-$^*$aACM-ES and BIPOP-aCMA-ES on noise-less BBOB testbed. It can be clearly seen that the results for pBFGS are much better than for BFGS, and high-precision computation indeed leads to a faster and more robust convergence. However, the results of pBFGS with restarts are still worse than the ones of BIPOP-$^*$aACM-ES.

An important remark is that in order to demonstrate the performance of pBFGS without loss of precision, we implemented not only the high-precision version of BFGS, but also a high-precision version of BBOB framework. The loss of performance occurs not only on the side of BFGS which computes the finite difference, but also on the side of BBOB framework which evaluates objective values for candidates, different in the decision space by a small $\epsilon$ value. It should be noted that a double-double version of some programming code is usually computationally more expensive, thus, the speedup from the use of pBFGS should be larger than the loss (typically a factor of 2) because of slower computations of $f$. Moreover, in a black-box scenario the source code of $f$ is unavailable.

Despite the uncovered difficulties of using the high-precision arithmetic within BFGS, we now have a better estimation of its "best case" performance. It would be also interesting to find a better implementation of BFGS with pBFGS-like performance, a promising candidate might be an implementation suggested in [Voglis et al., 2012] with an adaptable $\epsilon$ (see also [Voglis et al., 2009]). However, given that pBFGS is already outperformed by BIPOP-$^*$aACM-ES, it also may completely fail under certain rank-preserving transformations of $f$.

### 6.4.3 Comparison-based Multi-objective Optimization and Comparison-based Hypervolume Indicator

A natural extension of $^*$ to surrogate-assisted multi-objective optimization would be to build individual $\hat{f}_i$ surrogates for each of $f_i$ objectives ($i$ is the index of objective). The use of ordinal regression-based surrogate models will lead to their invariance w.r.t. rank-preserving transformation of $\hat{f}_i$. Surprisingly, this is a new feature for multi-objective optimization in the sense that most of MOEAs are not invariant w.r.t. rank-preserving transformations of $f_i$, because they use value-based metrics in the environmental selection (e.g., crowding distance, hypervolume indicator, epsilon indicator). Before going into details of the extension of $^*$, we wondered whether we could make MOEAs invariant w.r.t. rank-preserving transformations of $f_i$. We propose the following simple solution to this problem.

During the optimization we evaluate $\ell$ candidate solutions $x_{ij}$, $j = 1, \ldots, \ell$, on $f$ and store corresponding objective values $f_i(x_j)$ in archives $A_i$. Finally we have $m$ such archives ($m$ is the number of objectives) containing $\ell$ objective values of evaluated points. Then, objective values can be sorted such that we can associate to each candidate solution $x_j$ not only its raw objective value $f_i(x_j)$, but also its rank-based objective value $f_i^{\text{rank}}(x_j)$ which corresponds to the rank of its solution among other solutions on objective $f_i$. Thus, we may simply replace the raw objective values by rank-based values, and this does not change any
Pareto dominance relations among the points. We may use such intermediate procedure within any MOEAs, making them invariant w.r.t. rank-preserving transformations of $f_i$.

Our preliminary experiments within rank-based NSGA-II showed quite similar results to the original NSGA-II. Two important aspects should be noted: i) the fact of using rank-based objectives values does not change position of optimal Pareto front in the decision space, but ii) it may change the final Pareto front approximation, because the objectives now are differently scaled. The described above aspects mean that it is rather unlikely to have the same Pareto front approximation from the rank-based and value-based versions of the same MOEA, because their objectives are different.

Using the proposed rank-based framework, Quality Indicators also become rank-based. We anticipate interesting theoretical and practical results from rank-based hypervolume indicator. The fact that all objective values are integers and that adding a new point into a set we shift objective values of some points by +1 or -1 may bring some interesting results on computational complexity of hypervolume indicator computation. It should be noted, that the sorting of points in archives $A_i$ is actually not required, since the operation of insertion, deletion and access can be performed in $O(log(\ell))$ operations by representing archives as trees.

\section{Discussion}

In this Chapter, we investigated a set of algorithms based on CMA-ES for continuous optimization problems with different properties: single-objective (ACiD, NBIPOP-aCMA-ES, NIPOP-aCMA-ES), multi-objective (MO-CMA-ES with reward-based parent selection), large scale (ACiD, LACiD with linear time complexity) and multi-modal (NBIPOP-aCMA-ES, NIPOP-aCMA-ES).

We showed that simple deterministic coordinate descent method, performed on principal components of an adaptive coordinated system is at least as efficient as (1+1)-CMA-ES algorithm which operates by adapting multi-variate normal distribution in a stochastic way. The proposed Adaptive Coordinate Descent (ACiD) was also extended to LACiD, which requires only about $10n$ elementary operations per function evaluation in contrast to the fastest version of (1+1)-CMA-ES with $O(n^2)$ computational complexity. The overall speedup of LACiD in comparison with (1+1)-CMA-ES is in the order of $n$. To further improve the original ACiD in terms of number of function evaluations one should consider alternative strategies for search on principal components. A more extreme modification would be to replace PCA by Kernel PCA and perform the search on kernel principal components, which might be beneficial for highly non-linear optimization problems. A possible solution to further reduce the number of operations per function evaluation of LACiD would be to call the eigendecomposition even less often; this, however, may increase the overall number of function evaluations to reach the optimum. To reduce the $O(n^2)$ space complexity of LACiD, one should consider iterative search on principal components of a smaller dimension $n_{\text{small}} << n$. This may allow to extend the algorithm to $n > 10000$, but should be carefully done to prevent the explosion of overall number of function evaluations on fully non-separable ill-conditioned problems.
As was shown for MO-CMA-ES, not all parent individuals are equally interesting for generating offspring, and the proposed reward-based parent selection schemes may significantly speedup the convergence toward the optimal Pareto front. These schemes are efficient not only during the convergence phase of the search, but also for optimal $\mu$-distribution approximation before all points become equally prospective. While the schemes were tested for MO-CMA-ES where (1+1) selection scenario is applied, they also can be extended to multi-parent scenario (e.g., for NSGA-II) by sharing the reward between the parents.

The proposed alternative restart strategies for CMA-ES, NIPOP and NBIPOP, improve the results on multi-modal functions, especially in larger dimensions, thanks to a different principles of search in $(\lambda, \sigma)$-space of hyper-parameters of CMA-ES and the adaptive allocation of budgets of function evaluations for different regimes of restarts depending on their performance. We envision that the results can be further improved if the restarts of CMA-ES are viewed as individual evaluations in $(\lambda, \sigma)$-space: perform some surrogate-assisted search in this space. While the use of restarts of CMA-ES is currently a dominating approach to deal with multi-modal optimization, we suppose that the population size can be adapted during the search depending on the observed landscape. Our preliminary results suggests that a speedup of a factor of about 2 can be achieved by adjusting the population size during the search depending on the fitness-distance correlation metric. The latter can be be translated to the rank-based fitness distance correlation by using ranks of fitnesses and distances of individuals. Further research should also consider alternative niching methods [Shir and Bäck, 2006].

It also would be interesting to incorporate the Multi-Armed Bandits approach into CMA-ES for adaptive selection of different regimes, e.g., different regimes of uncertainty handling, parameters adaptation, etc.
Part IV

General Conclusion
Chapter 7

Conclusion

In this thesis, we have explored a wide range of approaches to the problem of efficient continuous optimization. What have we learned about continuous optimization, and what are the most promising direction for future research? This final Chapter seeks to answer these questions.

7.1 Summary of Contributions

In this thesis, we followed the strategy described in Chapter 1 and searched for optimal approaches to continuous optimization by first performing an exploitation step (Chapters 4 and 5) aiming at improvements with respect to "greedy" indicators of performance (e.g., number of function evaluations to reach the optimum), and then an exploration step (Chapter 6) aiming at introducing new prospective paradigms and approaches that would keep the diversity on a sufficiently rich level for further improvements. We applied our strategy to CMA-ES as a candidate algorithm with a relatively good performance at least in single-objective case (see Section 2.3.3).

In the exploitation part of the thesis we proposed to use ordinal regression-based (comparison-based) surrogate (approximate) models for expensive continuous optimization within CMA-ES.

In Chapter 4, we first showed why ordinal regression-based surrogates should be preferred to metric regression-based surrogates in the black-box scenario, when the landscape of the objective function is unknown. The most important reason is that the ordinal regression-based (comparison-based) surrogate models are invariant with respect to rank preserving transformations of the objective function. This, indeed, represents a source of robustness. Another source of such robustness is the invariance with respect to orthogonal transformations of the search space, which can be obtained from the information about correlations between the variables, provided by the covariance matrix adapted by CMA-ES. We showed that the approach of Support Vector Machines can be efficiently used to build surrogates models for surrogate-assisted CMA-ES such that the described above invariance properties, inherent to CMA-ES, can be preserved. When the surrogate model is built using a set of hyper-parameters and a set training points, it can be directly optimized by CMA-ES for a given number of generations, which is a function (e.g., linear function) of the estimated model quality. An important contribution of this thesis is the proposed surrogate model hyper-parameters adaptation procedure: the user is only needed to define the range of controlling hyper-parameters of the surrogate learning, and the procedure will optimize them automatically during the optimization of the objective function, such that the surrogate model is (optimized to be) best suited to the
current search region. The proposed framework and its CMA-ES instances IPOP-∗∗ aACM-ES and BIPOP-∗∗ aACM-ES were extensively benchmarked on BBOB noiseless and noisy benchmark problems, where they improved (often by a factor of 2-3) the best known results for certain problems obtained from more than 40 other classical and Evolutionary Algorithms. The aggregated results for noiseless problems are best illustrated in Figure 4.19. We also showed that the approach can be efficiently parallelized: this is, indeed, not the case for many surrogate-assisted algorithms (e.g., some trust-region based algorithms).

In Chapter 5, we investigated surrogate-assisted multi-objective search within MO-CMA-ES and NSGA-II using a relatively simple model exploitation procedure based on offspring pre-selection. We proposed to build Aggregated Surrogate Model of the original multi-objective problem in order to aggregate the information about the preferences between solutions in the objective space. Different sources of preference information can be used: Pareto dominance, Quality Indicators or Decision Maker based preferences. In contrast to ”multi-surrogate” approaches, the proposed ASM model represents a single objective which imitates an aggregated preference which the optimization algorithm should follow. In the procedure of model exploitation, \( \lambda_{pre} \) offspring per one parent are generated for pre-selection of the best candidate for its later evaluation on the expensive function. The experimental validation of ASM-assisted MO-CMA-ES and NSGA-II algorithms showed that ASM models significantly speedup the search (by a factor of about 2) in the phase of convergence toward the optimal Pareto front, but tends to lack of diversity in the phase of the optimal Pareto front approximation.

In the exploration part of the thesis, we presented a set of approaches which we developed in order to explore and extend the frontiers of Evolutionary Computation, and CMA-ES in particular, answering some questions which have arisen in the exploitation part of the thesis.

In Chapter 5, we proposed Adaptive Coordinate Descent answering the question whether we could efficiently exploit a potential separability of the decorrelated search space provided by Adaptive Encoding procedure. The final algorithm was shown to be competitive with (1+1)-CMA-ES, while being deterministic and based on simple dichotomy method. The extension of ACiD to large scale optimization, linear time ACiD (LACiD), demonstrated slightly worse performance than (1+1)-CMA-ES in terms of number of function evaluations (a loss of about a factor of 2 for \( n = 512 \) on Ellipsoid and Rosenbrock functions), but linear time complexity in contrast to quadratic time complexity of the fastest version of (1+1)-CMA-ES. In fact, LACiD scales with \( 10n \) elementary operations that is about \( 10^{-8}n \) seconds per function evaluation (see Figure 6.7). Thus, with this algorithm we explored the extreme opposite of surrogate-assisted search for expensive optimization - large scale optimization with cheap function evaluations.

Answering the question whether we could improve the results of multi-objective optimizers without using surrogate models, we proposed to look more carefully at the dynamic of success of parent individuals in offspring sampling. We showed that reward-based parent selection schemes can be highly beneficial for MO-CMA-ES allowing the algorithm to quickly catch fruitful directions of search.

Our observations of a relatively small speedup of BIPOP-∗∗ aACM-ES on multi-modal problems (due to difficult to learn the fitness landscapes) motivated the development of
simple restarts strategies for CMA-ES, NIPOP and NBIPOP, which were found to be competitive to the original restart schemes, IPOP and BIPOP. The new schemes are often faster than the original restarts schemes especially in higher dimensions, where the alternative exploration of $(\lambda, \sigma)$-space and the adaption of computational budgets of regimes of restarts are especially beneficial. In order to check against overfitting to BBOB benchmark problems, the proposed restarts schemes were tested on the real-world problem of interplanetary trajectory optimization, where they also showed good performance.

7.2 Future Directions

Here we would like to mention several directions of future research which look the most promising, of course, without pretending to cover everything that is important for EC, and CMA-ES in particular. Some of these ideas were already discussed in the corresponding Chapters and Sections.

There is a global goal to find an extremely cheap and fast (in terms of computational complexity and memory requirements and number of functions evaluations) optimization algorithm, suitable for different classes of optimization problems. It seems difficult to find such algorithm in a near future (if it exists) and we rather should think of a Pareto set of algorithms which better or worse satisfy the described above objectives.

A subspace of computationally cheap algorithms is covered by SEP-CMA-ES, Rank-One NES and linear time ACiD. They, however, have different performance and memory requirements. For example, the linear time ACiD is suitable for learning the full covariance matrix and performs well on Rotated Ellipsoid problem, but requires $O(n^2)$ memory space in order to store the covariance matrix. Thus, the primary interest in this direction would be to further explore these algorithms to find i). a version of linear time ACiD with linear space complexity; ii). a version of the original CMA-ES which uses linear time offspring sampling procedure and, thus, can be linear in time; iii). versions of SEP-CMA-ES and Rank-One NES which may learn few more principal components and, thus, performs better on fully non-separable problems.

Our preliminary results show that $*$aACM-ES algorithms can be further improved by a factor of up to 2 by more carefully looking at the surrogate model exploitation procedure. This would further shift to the left the curves of $*$aACM-ES algorithms presented in Figure 4.19. We also anticipate good results from the use of different surrogate learning techniques and their simultaneous adaptation to the current region of search. The primary interest in surrogate-assisted search would be to i). reduce the time and space complexity of the used surrogate learning procedure; ii). improve the accuracy of surrogates on multi-modal functions.

A natural perspective for $*$aACM-ES would be its extension to multi-objective optimization. This would require the rank-based formulation of multi-objective optimization and rank-based Hypervolume indicator (i.e., invariant with respect to the rank-preserving transformations of objectives) as discussed in Section 6.4.3. The surrogate model exploitation procedure, however, might be more complicated and less efficient since the accuracy of models for different objectives may differ significantly (e.g., from ideal to random pre-
diction). Alternatively, if we succeed to carefully replace PCA by Kernel PCA in Adaptive Encoding part of CMA-ES, then the use of ASM surrogate models for multi-objective optimization seems to be prospective. In this case, Kernel PCA-based $*\alpha$ACM-ES would learn the distribution of non-dominated points and sample offspring according to this distribution. We may evaluate these offspring on ASM model learned from the preference information, thus, unifying the single-objective and multi-objective surrogate-assisted optimization.
Bibliography


[Hansen et al., 2010b] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. Comparing results of 31 algorithms from the black-box optimization bench-

214


[Igel et al., 2007a] C. Igel, T. Suttrop, and N. Hansen. Steady-state Selection and Efficient Covariance Matrix Update in the Multi-objective CMA-ES. In S. Obayashi et al.,


Appendix A
Figure 1: Average surrogate model error in the space of model hyper-parameters measured in 10 runs of Ranking SVM with RBF kernel (First Row), Polynomial kernel with degree 2 (Second Row) and Polynomial kernel with degree 30 (Third Row) on Sphere (Left Column) and Rosenbrock (Right Column) functions in 10-D after 1000 \( \ell \) iterations of the SMO algorithm. The observed best pair of hyper-parameters is denoted as ‘*’.
Figure 2: Average surrogate model error in the space of model hyper-parameters measured in 10 runs of SVR with RBF kernel on (Left Column) Sphere and (Right Column) Rosenbrock functions in 10-D (First Row) and its scaled variants $f^2$ and $f^4$ (Second and Third Rows) after 100ℓ iterations of the SMO algorithm. The observed best pair of hyper-parameters is denoted as ‘*’.
Figure 3: Average surrogate model error in the space of model hyper-parameters measured in 10 runs of Ranking SVM (Left Column) and (1:N) rank-based SVR (Right Column) with RBF kernel and $\ell = 2048$ training points on Sphere function in 2-D (First Row), 10-D (First Row) and 40-D (First Row) after 1000$\ell$ (respectively, 100$\ell$) iterations of the SMO algorithm for Ranking SVM (respectively, SVR). The observed best pair of hyper-parameters is denoted as '*'.

236
Figure 4: Average surrogate model error in the space of model hyper-parameters measured in 10 runs of Ranking SVM (Left Column) and (1:N) rank-based SVR (Right Column) with RBF kernel and $\ell = 2048$ training points on Rosenbrock function in 2-D (First Row), 10-D (First Row) and 40-D (First Row) after $1000\ell$ (respectively, $100\ell$) iterations of the SMO algorithm for Ranking SVM (respectively, SVR). The observed best pair of hyper-parameters is denoted as ‘*’.
Appendix B

B.1 Dual Form of ASM Learning Problem

Introducing the non-negative Lagrangian multipliers $\alpha_i^{(*)}$ and $\beta_i^{(*)}$ for each constraint (5.2-5.7) respectively (where $^{(*)}$ is either $^{(up)}$ or $^{(low)}$), the dual form is:

$$L(w, \rho, \xi^{(*)}, \alpha^{(*)}, \beta^{(*)}) = \frac{1}{2}||w||^2 + C\sum_{i=1}^{\ell}(\xi_i^{up} + \xi_i^{low}) + C\sum_{i=\ell+1}^{m}\xi_i^{up} + \rho$$

Computing the KKT conditions leads to:

$$\frac{\partial L}{\partial w} = w + \sum_{i=1}^{\ell}(\alpha_i^{up} - \alpha_i^{low})\Phi(x_i) + \sum_{i=\ell+1}^{m}\alpha_i^{up}\Phi(x_i) = 0$$

$$\frac{\partial L}{\partial \rho} = 1 - \sum_{i=1}^{\ell}(\alpha_i^{up} - \alpha_i^{low}) - \sum_{i=\ell+1}^{m}\alpha_i^{up} = 0$$

$$\frac{\partial L}{\partial \xi_i^{up}} = C - \alpha_i^{up} - \beta_i^{up} = 0$$

$$\frac{\partial L}{\partial \xi_i^{low}} = C - \alpha_i^{low} - \beta_i^{low} = 0$$

$$\frac{\partial L}{\partial \xi_j^{up}} = C - \alpha_j^{up} - \beta_j^{up} = 0$$

Therefore, at the saddle point we have:

$$w = \sum_{i=1}^{\ell}\alpha_i^{low}\Phi(x_i) - \sum_{i=1}^{m}\alpha_i^{up}\Phi(x_j)$$

$$1 = \sum_{i=1}^{m}\alpha_i^{up} - \sum_{i=1}^{\ell}\alpha_i^{low}$$

$$C = \alpha_i^{up} + \beta_i^{up} = \alpha_i^{low} + \beta_i^{low}$$
Reporting these equalities, the Lagrangian becomes:

\[
L(w, \rho, \xi, \alpha, \beta) = -\frac{1}{2}||w||^2 - \epsilon \left( \sum_{i=1}^{\ell} (\alpha_i^{up} + \alpha_i^{low}) - \sum_{i=\ell+1}^{m} \alpha_i^{up} \right)
\]

\[
= -\frac{1}{2}||w||^2 - \epsilon \left( 2 \sum_{i=1}^{\ell} \alpha_i^{up} - 1 \right)
\]

Let us first define some additional notations. Denote

\[
K_{ij} = K(x_i, x_j) = \langle \Phi(x_i), \Phi(x_j) \rangle
\]

and

\[
\gamma_i = \begin{cases} 
(\alpha_i^{up} - \alpha_i^{low}) & \text{if } i = 1 \ldots \ell \\
\alpha_i^{up} & \text{otherwise}
\end{cases}
\]

Then:

\[
L(w, \rho, \xi, \alpha, \beta) = -\frac{1}{2} \sum_{i,j=1}^{m} \gamma_i \gamma_j K(x_i, x_j) - \epsilon \left( 2 \sum_{i=1}^{\ell} \alpha_i^{up} - 1 \right)
\]

Eliminating the \( \beta^{(*)} \) thanks to relations (8), the dual problem to solve in \( (\alpha^{(*)}) \) is:

Maximize

\[
\tilde{L}(\alpha^{(*)}) = -\frac{1}{2}||w||^2 - \epsilon \left( 2 \sum_{i=1}^{\ell} \alpha_i^{up} - 1 \right)
\]

subject to

\[
\sum_{i=1}^{m} \gamma_i = \sum_{i=1}^{m} \alpha_i^{up} - \sum_{i=1}^{\ell} \alpha_i^{low} = 1 \quad (10)
\]

\[
0 \leq \alpha_i^{(*)} \leq C \quad (11)
\]

B.2 Solving the Dual Problem

Following [Smola and Schölkopf, 2004], the idea is to iterate exact resolutions of the maximization problem by varying only two of the \( \alpha \)’s multipliers. Thanks to the sum constraint (Eq. (7) or (10)), one of the \( \alpha \) variables can be eliminated. As the resulting function, now depending on a single variable, is quadratic, its optimization can be solved analytically. It remains to choose the pair of \( \alpha \) indices; this choice has a large impact on the overall computational cost for large regression problems, and several heuristics have been proposed [Glasmachers and Igel, 2008]. It turns out that the best results in our problem were obtained for a uniform selection of the \( \alpha \) indices.
B.2 Solving the Dual Problem

B.2.1 The One-dimensional Maximization Problem

Using the same formulation as [13], let \( \varphi_i \) denote the model value at point \( x_i \), that is

\[
\varphi_i = \mathcal{F}(x_i) = \langle w, \Phi(x_i) \rangle = \sum_{j=1}^{m} \gamma_j K_{ij}
\]

Maximizing the Lagrangian as a function of variables \( i \) and \( j \) only, amounts to maximize (after removing terms that do not depend on those variables):

\[
\hat{L}_{ij}(\alpha_{i}^{(s)}, \alpha_{j}^{(s)}) = -\frac{1}{2} \left[ \gamma_i^2 K_{ii} + \gamma_j^2 K_{jj} + 2\gamma_i \gamma_j K_{ij} \right] - \gamma_i \left[ \sum_{k \neq i,j} \gamma_k K_{ki} \right] - \gamma_j \left[ \sum_{k \neq i,j} \gamma_k K_{kj} \right] - \epsilon \left( \hat{\gamma}_i + \hat{\gamma}_j \right)
\]

where \( \hat{\gamma}_i = 2 \alpha_{i}^{up} \) for Pareto points, and \( \hat{\gamma}_i = 0 \) for dominated points.

From Eq. (10), it comes:

\[
\gamma_i + \gamma_j = \gamma_i^{old} + \gamma_j^{old} \overset{\text{def}}{=} \Gamma
\]

Moreover, for all \( i \)

\[
\sum_{k \neq i,j} \gamma_k K_{ki} = \varphi_i - \gamma_i^{old} K_{ii} - \gamma_j^{old} K_{ij}
\]

Using the above in (12) for \( i \) and \( j \), and eliminating \( j \) thanks to the summation constraint (\( \gamma_i = \Gamma - \gamma_j \)) leads to a “single” variable maximization problem with unknown either \( \alpha_{i}^{*} \), if \( x_i \) is a dominated point, or \( \alpha_{i}^{up} / \alpha_{i}^{low} \), knowing that at most one is non-zero:

\[
\hat{L}(\alpha_{i}^{(s)}) = -\frac{1}{2} \left[ \gamma_i^2 K_{ii} + (\Gamma - \gamma_i)^2 K_{jj} + \gamma_i (\Gamma - \gamma_i) K_{ij} \right] - \gamma_i \left[ \varphi_i - \gamma_i^{old} K_{ii} - (\Gamma - \gamma_i^{old}) K_{ij} \right] - (\Gamma - \gamma_i) \left[ \varphi_j - (\Gamma - \gamma_j^{old}) K_{jj} - \gamma_i^{old} K_{ij} \right] - \epsilon \left( \hat{\gamma}_i + \hat{\gamma}_j \right)
\]

where

\[
\eta \overset{\text{def}}{=} K_{ii} + K_{jj} - 2 K_{ij} = \langle \Phi(x_i) - \Phi(x_j), \Phi(x_i) - \Phi(x_j) \rangle > 0 \text{ if } i \neq j
\]

and after having neglected with no loss of generality the terms that do not depend on any of the \( \alpha_{i}^{(s)} \).

We now need to distinguish the different cases to instantiate the \( \gamma_i^{(s)} \) and \( \hat{\gamma}_i \).
Dominated-Dominated

When both $x_i$ and $x_j$ are dominated, we have

$$\gamma_i = \alpha_i^{up} \text{ and } \hat{\gamma}_i = 0$$

and same holds for $j$. Thanks to the reduced sum-constraint (13), the one-dimensional Lagrangian becomes

$$\hat{L}(\alpha_i^{up}) = \frac{1}{2}(\alpha_i^{up})^2 \eta - \alpha_i^{up} (\varphi_i - \varphi_j - \alpha_i^{up,old} \eta)$$

The maximum is thus reached for

$$\alpha_i^{up} = \alpha_i^{up,old} + \frac{\varphi_j - \varphi_i}{\eta}$$

Because $\alpha_k^{up} \in [0, C]$ for all $k$, the bounds for $\alpha_i^{up}$ are now $[\max(0, \Gamma - C), \min(C, \Gamma)]$. Whenever one of the bounds is violated, $\alpha_i^{up}$ is reset to the bound, and in all cases, $\alpha_j^{up}$ is computed according to the sum-constraint (13).

Pareto-Dominated

Suppose that $x_i$ is non-dominated, and $x_j$ is dominated. Then

$$\gamma_i = \alpha_i^{up} - \alpha_i^{low} \quad \text{and} \quad \hat{\gamma}_i = 2 * \alpha_i^{up}$$

$$\gamma_j = \alpha_j^{up} \quad \text{and} \quad \hat{\gamma}_j = 0$$

We have to distinguish 2 cases, depending on which of $\alpha_i^{up,old}$ and $\alpha_i^{low,old}$ is not 0.

- $\alpha_i^{up,old} \neq 0$
  
  In this case, at least in some neighborhood of the current value $\alpha_i^{up,old}$,

  $$\gamma_i = \alpha_i^{up} \quad \text{and} \quad \hat{\gamma}_i = 2 * \alpha_i^{up}$$

  Thanks to the reduced sum-constraint (13), the one-dimensional Lagrangian becomes

  $$\hat{L}(\alpha_i^{up}) = -\frac{1}{2}(\alpha_i^{up})^2 \eta - \alpha_i^{up} (\varphi_i - \varphi_j - \alpha_i^{up,old} \eta + 2 \varepsilon)$$

  The maximum is thus reached for

  $$\alpha_i^{up} = \alpha_i^{up,old} + \frac{\varphi_j - \varphi_i - 2 \varepsilon}{\eta}$$

  From the reduced sum-constraint (13) we deduce the same bounds as in previous section for $\alpha_i^{up}$, i.e. $[\max(0, \Gamma - C), \min(C, \Gamma)]$. However, whereas the same action must be taken when the upper-bound is reached (set $\alpha_i^{up}$ to the upper bound, and compute $\alpha_j^{up}$ according to the sum-constraint (13), the situation is different if $\alpha_i^{up}$ violates its lower bound: if that bound is strictly positive (i.e. $\Gamma - C$ with $\Gamma > C$), again $\alpha_j^{up}$ should simply
be computed according to the sum-constraint (13). But if the lower bound is 0, it is likely that increasing \( \alpha^\text{low} \) might increase the Lagrangian further.

In the latter case, one has to consider again the reduced Lagrangian (12), but with now

\[
\gamma_i = -\alpha^\text{low}_i \quad \text{and} \quad \hat{\gamma}_i = 0 \quad \text{but with} \quad \gamma^\text{old}_i = \alpha^\text{up,old}_i
\]

which hence gives

\[
\hat{L}(\alpha^\text{up}_i) = -\frac{1}{2}(\alpha^\text{low}_i)^2 \eta + \alpha^\text{low}_i (\varphi_i - \varphi_j - \alpha^\text{up,old}_i \eta) \tag{20}
\]

whose maximum is reached for

\[
\alpha^\text{low}_i = -\alpha^\text{up,old}_i + \frac{\varphi_i - \varphi_j}{\eta} \tag{21}
\]

\( \alpha^\text{low,old}_i \neq 0 \)

In this case, at least in some neighborhood of the current value \( \alpha^\text{low,old}_i \),

\[
\gamma_i = -\alpha^\text{low}_i \quad \text{and} \quad \hat{\gamma}_i = 0
\]

The reduced sum constraint (13) becomes

\[
-\alpha^\text{low}_i + \alpha^\text{up}_j = \Gamma \tag{22}
\]

and the one-dimensional Lagrangian now reads

\[
\hat{L}(\alpha^\text{low}_i) = -\frac{1}{2}(\alpha^\text{low}_i)^2 \eta + \alpha^\text{low}_i (\varphi_i - \varphi_j + \alpha^\text{low,old}_i \eta) \tag{23}
\]

Its maximum is this reached for

\[
\alpha^\text{low}_i = \alpha^\text{low,old}_i + \frac{\varphi_i - \varphi_j}{\eta} \tag{24}
\]

The bounds for \( \alpha^\text{low}_i \) are now \([\max(0, -\Gamma), \min(C, C - \Gamma)]\). As in the previous case, if \( \alpha^\text{low}_i \) hits it upper bound, or its strictly positive lower bound, \( \alpha^\text{up}_j \) should simply be computed according to the constraint (22). But if it hits its 0 as its lower bound, here again maximization might be pursued, in a similar way than in previous section: the solution is given by

\[
\alpha^\text{up}_i = -\alpha^\text{low,old}_i + \frac{\varphi_j - \varphi_i - 2\varepsilon}{\eta} \tag{25}
\]

\( \alpha^\text{up,old}_i = \alpha^\text{low,old}_i = 0 \)

In this case, that could probably enter in both previous cases, both options should be explored: looking into the domain where \( \alpha^\text{up}_i > 0 \), it comes from equation (19) that the maximum value is reached for \( \alpha^\text{up}_i = \frac{\varphi_j - \varphi_i - 2\varepsilon}{\eta} \) while in the domain \( \alpha^\text{low}_i > 0 \), equation (24) gives that the maximum is reached for \( \alpha^\text{low}_i = \frac{\varphi_i - \varphi_j}{\eta} \). If both quantities are negative (i.e. \( 0 < \phi_j - \phi_i < 2\varepsilon \)), the solution is \( \alpha^\text{up}_i = \alpha^\text{low}_i = 0 \). Otherwise, only one of these quantities is positive, and should be retained – with the same truncations than in sections B.2.2 or B.2.1.
Pareto-Pareto

Suppose now that both \( x_i \) and \( x_j \) belong to the Pareto front. Then

\[
\gamma_i = \alpha_i^{up} - \alpha_i^{low} \quad \text{and} \quad \gamma_j = \alpha_j^{up} - \alpha_j^{low}
\]

As in previous section, we have to distinguish different cases depending on which is non zero from \( \alpha_i^{up} \) and \( \alpha_i^{low} \), and from \( \alpha_j^{up} \) and \( \alpha_j^{low} \).

\( \alpha_i^{up,old} > 0 \) and \( \alpha_j^{up,old} > 0 \)

Thanks to the reduced sum-constraint \( \alpha_i^{up} + \alpha_j^{up} = \Gamma \), the one-dimensional Lagrangian now reads

\[
\hat{\mathcal{L}}(\alpha_i^{up}) = -\frac{1}{2}(\alpha_i^{up})^2 \eta - \alpha_i^{up} (\varphi_i - \varphi_j - \alpha_i^{up,old} \eta)
\]

Its maximum is this reached for

\[
\alpha_i^{up} = \alpha_i^{up,old} + \frac{\varphi_j - \varphi_i}{\eta}
\]

The bounds for \( \alpha_i^{low} \) are, again, \([\max(0, \Gamma - C), \min(C, \Gamma)]\). If either \( \alpha_i^{up} \) or \( \alpha_j^{up} \) hits the 0 lower bound, maximization should be continued in the corresponding domain (either \( \alpha_i^{up} = 0, \alpha_i^{low} > 0 \) or \( \alpha_j^{up} = 0, \alpha_j^{low} > 0 \)).

\( \alpha_i^{low,old} > 0 \) and \( \alpha_j^{up,old} > 0 \)

The reduced sum-constraint now reads \(-\alpha_i^{low} + \alpha_j^{up} = \Gamma\), and the one-dimensional Lagrangian becomes

\[
\hat{\mathcal{L}}(\alpha_i^{low}) = -\frac{1}{2}(\alpha_i^{low})^2 \eta + \alpha_i^{low} (\varphi_i - \varphi_j + \alpha_i^{low,old} \eta - 2\varepsilon)
\]

Its maximum is this reached for

\[
\alpha_i^{low} = \alpha_i^{low,old} + \frac{\varphi_i - \varphi_i - 2\varepsilon}{\eta}
\]

The bounds for \( \alpha_i^{low} \) are here \([\max(0, -\Gamma), \min(C, C - \Gamma)]\). Again, if one of the \( \alpha \)'s hits the 0 bound, maximization should be continued in the other domain.

\( \alpha_i^{up,old} > 0 \) and \( \alpha_j^{low,old} > 0 \)

Similarly, the reduced sum-constraint now reads \( \alpha_i^{up} - \alpha_j^{low} = \Gamma \), and the one-dimensional Lagrangian becomes

\[
\hat{\mathcal{L}}(\alpha_i^{up}) = -\frac{1}{2}(\alpha_i^{up})^2 \eta - \alpha_i^{up} (\varphi_i - \varphi_j - \alpha_i^{up,old} \eta + 2\varepsilon)
\]

Its maximum is this reached for

\[
\alpha_i^{up} = \alpha_i^{up,old} + \frac{\varphi_j - \varphi_i - 2\varepsilon}{\eta}
\]
B.2 Solving the Dual Problem

The bounds for $\alpha_i^{up}$ are here $[\max(0, \Gamma), \min(C, C + \Gamma)]$. Again, if one of the $\alpha$’s hits the 0 bound, maximization should be continued in the other domain.

$\alpha_{i,\text{low,old}} > 0$ and $\alpha_{j,\text{low,old}} > 0$

The reduced sum-constraint now reads $\alpha_{i,\text{low}} + \alpha_{j,\text{up}} = -\Gamma$, the term in $\epsilon$ hence vanishes, and the onedimensional Lagrangian becomes

$$\hat{\mathcal{L}}(\alpha_{i,\text{low}}) = -\frac{1}{2}(\alpha_{i,\text{low}})^2 \eta + \alpha_{i,\text{low}} (\varphi_i - \varphi_j + \alpha_{i,\text{low,old}} \eta)$$ (32)

Its maximum is this reached for

$$\alpha_{i,\text{low}} = \alpha_{i,\text{low,old}} + \frac{\varphi_i - \varphi_j}{\eta}$$ (33)

The bounds for $\alpha_{i,\text{low}}$ are here $[\max(0, -C - \Gamma), \min(C, -\Gamma)]$, and again, if one of the $\alpha$’s hits the 0 bound, maximization should be continued in the other domain.