Searching for Compact Hierarchical Structures in DNA by means of the Smallest Grammar Problem

Matthias Gallé

François Coste
Symbiose Project
INRIA/IRISA
France

Gabriel Infante-López
NLP Group
U. N. de Córdoba
Argentina

Université de Rennes 1
February, 15th 2011

Motivation: Deciphering a Text

Motivation: Deciphering a Text

Twas brillig, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

©leninimports.com

Motivation: Deciphering a Text

Twas **brillig**, and the slithy toves
Did gyre and gimble in the wabe;
All mimsy were the borogoves,
And the mome raths outgrabe.

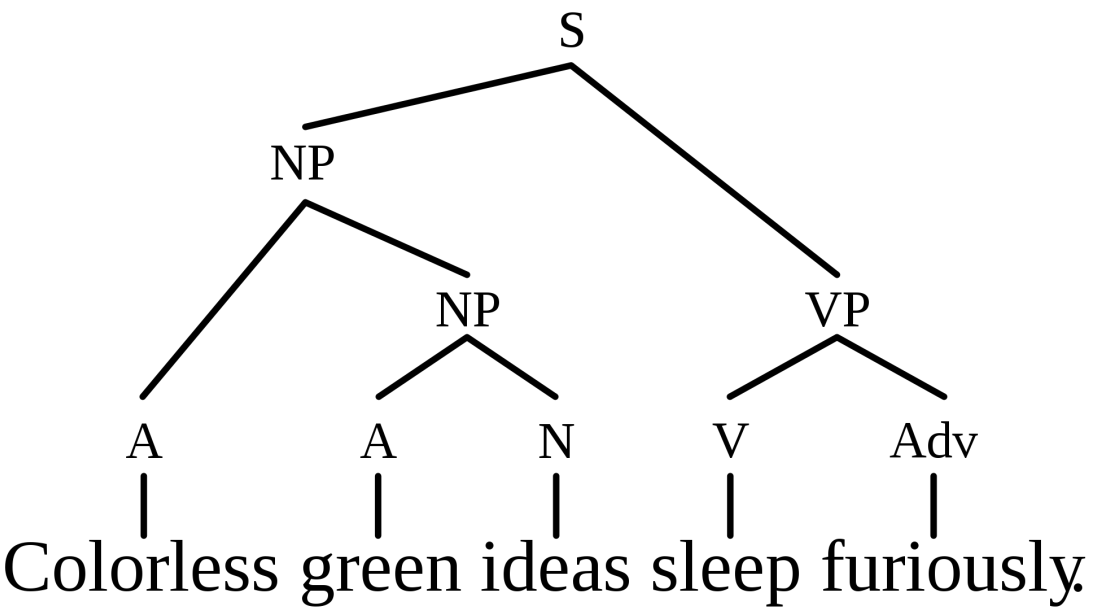
"That's enough to begin with", Humpty Dumpty interrupted: "there are plenty of hard words there. 'BRILLIG' means four o'clock in the afternoon – the time when you begin BROILING things for dinner."

Motivation: Deciphering a Text

Colorless green ideas sleep furiously

©J. Soares, chomsky.info

Motivation: Deciphering a Text



©wikipedia

©J. Soares, chomsky.info

Motivation: Deciphering a Text

ATGGCCCGGACGAAGCAGACAGCTCGCAAGTCTACCGGC

GGCAAGGCACCGCGGAAGCAGCTGGCCACCAAGGCAGCG

CGCAAAAGCGCTCCAGCGACTGGCGGTGTGAAGAAGCCC

CACCGCTACAGGCCAGGCACCGTGGCCTTGCGTGAGATC

CGCCGTTATCAGAAGTCGACTGAGCTGCTCATCCGCAAA

CTGCCATTTCAGCGCCTGGTGCGAGAAATCGCGCAGGAT

TTCAAAACCGACCTTCGTTTCCAGAGCTCGGCGGTGATG

GCGCTGCAAGAGGCGTGCGAGGCCTATCTGGTGGGTCTC

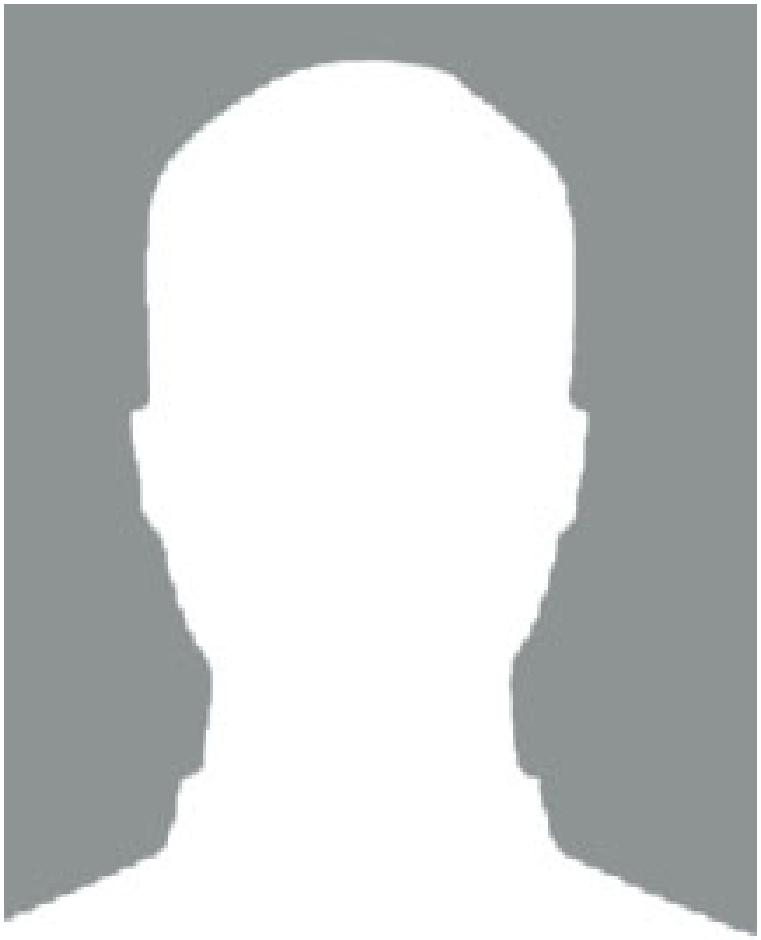
TTTGAAGACACCAACCTCTGTGCTATTCACGCCAAGCGT

GTCACTATTATGCCTAAGGACATCCAGCTTGCGCGTCGT

ATCCGTGGCGAGCGAGCATAATCCCCTGCTCTATCTTGG

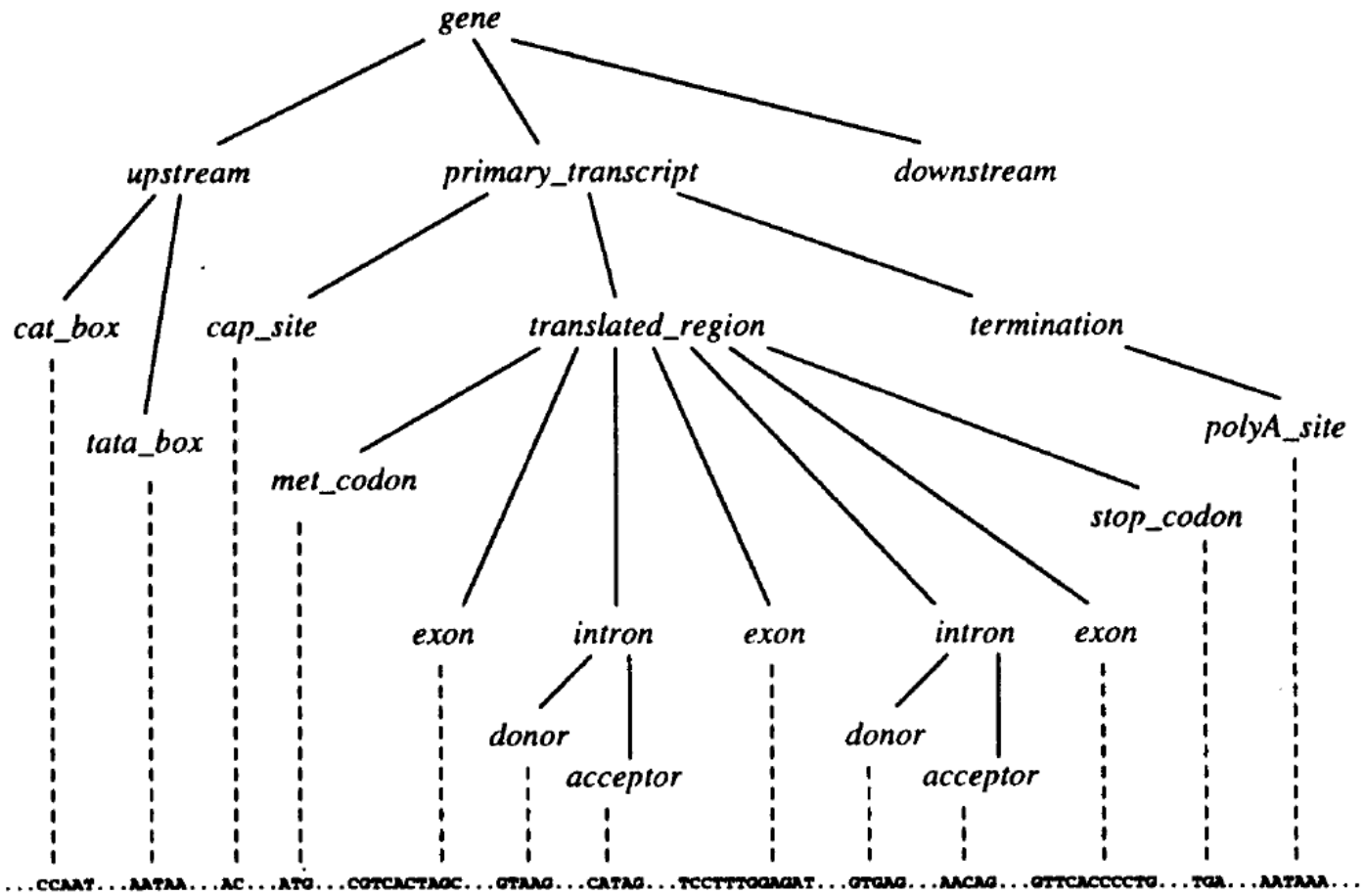
GTTTCTTAATTGCTTCCAAGCTTCCAAAGGCTCTTTTC

AGAGCCACTTA



©You (HIST1H3J, chromosome 6)

Structuring DNA



Linguistics of DNA

- A good metaphor ("transcription", "translation"), but also more than that

Linguistics of DNA

- A good metaphor ("transcription", "translation"), but also more than that
- What can linguistic models reveal about DNA?
  Ex: "A linguistic model for the rational design of antimicrobial peptides".
  Loose, Jensen, Rigoutsos, Stephanopoulos. Nature 2003

Linguistics of DNA

- A good metaphor ("transcription", "translation"), but also more than that
- What can linguistic models reveal about DNA?
  Ex: "A linguistic model for the rational design of antimicrobial peptides".
  Loose, Jensen, Rigoutsos, Stephanopoulos. Nature 2003
- Use of Formal Grammars

Learning the Linguistics of DNA

At Symbiose [Kerbellec, Coste 08] obtained good results modelling families of proteins with non-deterministic finite automata

$$\Downarrow$$

Choice 1 Go up to context-freeness (long-range correlations, memory), on DNA sequences

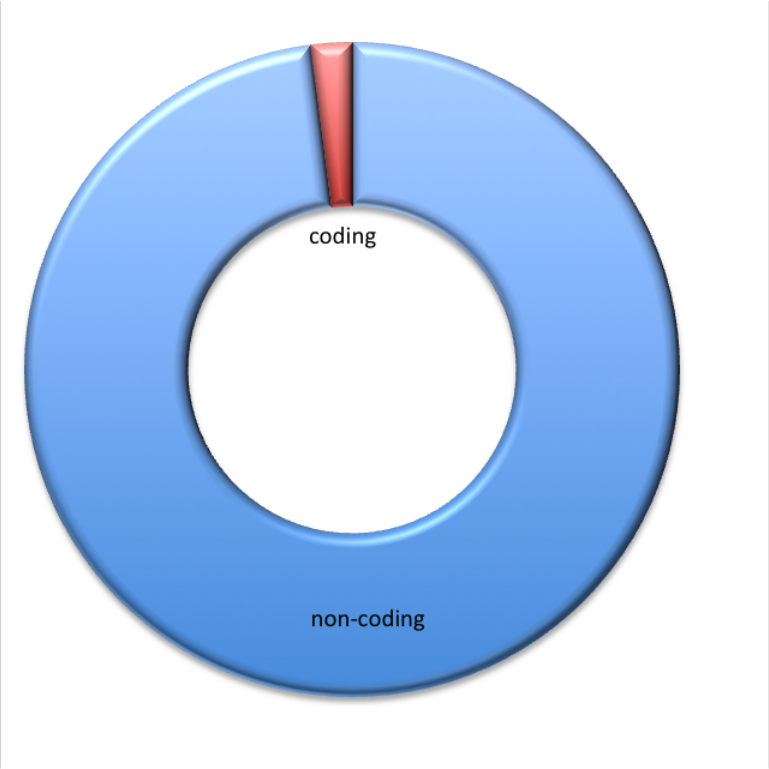
What is a good context-free grammar

What is a good context-free grammar: Stay generic

We don't want to introduce any domain-specific learning bias

What is a good context-free grammar: Stay generic

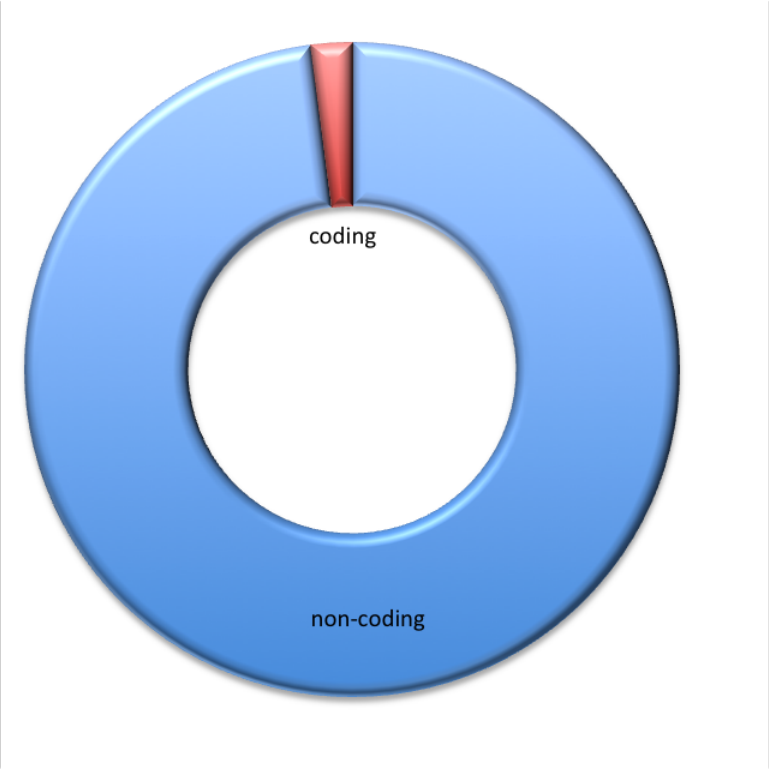
We don't want to introduce any domain-specific learning bias



Proportion in Human Genome

What is a good context-free grammar: Stay generic

We don't want to introduce any domain-specific learning bias



Proportion in Human Genome

⇒ **Choice 2** Use Occam's Razor and search for the smallest grammar

Formalisation of our Problem

Motivation Unveil hierarchical structure in DNA

Choice 1 Model: Context-free grammar

\+ **Choice 2** Goodness: Occam's Razor

= The Smallest Grammar Problem: finding the <u>smallest context-free grammar</u> that generates exactly one sequence

Formalisation of our Problem

Motivation Unveil hierarchical structure in DNA

Choice 1 Model: Context-free grammar

\+ **Choice 2** Goodness: Occam's Razor

= The Smallest Grammar Problem: finding the smallest context-free grammar that generates exactly one sequence

Remark

On the way, don't forget to be feasible enough to apply on DNA

Smallest Grammar Problem

Problem Definition

Given a sequence s, find a grammar $G(s)$ of smallest size that generates only s.

Smallest Grammar Problem

An Example

Problem Definition

Given a sequence s, find a grammar $G(s)$ of smallest size that generates only s.

Example

$s =$ "`how much wood would a woodchuck chuck if a woodchuck could chuck wood?`", a possible $G(s)$ (not necessarily smallest) is

$$
\begin{array}{lcl}
S & \longrightarrow & \text{how much } N_2 \text{ w} N_3\ N_4\ N_1 \text{ if } N_4 \text{ c} N_3\ N_1\ N_2 \text{ ?} \\
N_1 & \longrightarrow & \text{chuck} \\
N_2 & \longrightarrow & \text{wood} \\
N_3 & \longrightarrow & \text{ould} \\
N_4 & \longrightarrow & \text{a } N_2 N_1
\end{array}
$$

Smallest Grammar Problem

Straight-line grammars

Problem Definition

Given a sequence s, find a straight-line context-free grammar $G(s)$ of smallest size that generates s.

Remark

Grammars that do not branch (one and only one production rule for every non-terminal) nor loop (no recursion)

Smallest Grammar Problem

Definition of $|G|$

Problem Definition

Given a sequence s, find a straight-line context-free grammar $G(s)$ of smallest size that generates s.

Size of a Grammar

$$|G| = \sum_{N \to \omega \in \mathcal{P}} (|\omega| + 1)$$

Smallest Grammar Problem

Definition of $|G|$

Problem Definition

Given a sequence s, find a straight-line context-free grammar $G(s)$ of smallest size that generates s.

Size of a Grammar

$$|G| = \sum_{N \to \omega \in \mathcal{P}} (|\omega| + 1)$$

$$\begin{array}{lcl} S & \to & \text{how much } N_2 \text{ w} N_3 \ N_4 \ N_1 \text{ if } N_4 \text{ c} N_3 \ N_1 \ N_2 \text{ ?} \\ N_1 & \to & \text{chuck} \\ N_2 & \to & \text{wood} \\ N_3 & \to & \text{ould} \\ N_4 & \to & \text{a } N_2 N_1 \end{array}$$

$$\Downarrow$$

$$\text{how much } N_2 \text{ w} N_3 \ N_4 \ N_1 \text{ if } N_4 \text{ c} N_3 \ N_1 \ N_2 \mid \text{chuck} \mid \text{wood} \mid \text{ould} \mid \text{a } N_2 \ N_1 \mid$$

Smallest Grammar Problem

Hardness

Problem Definition

Given a sequence s, find a straight-line context-free grammar $G(s)$ of smallest size that generates s.

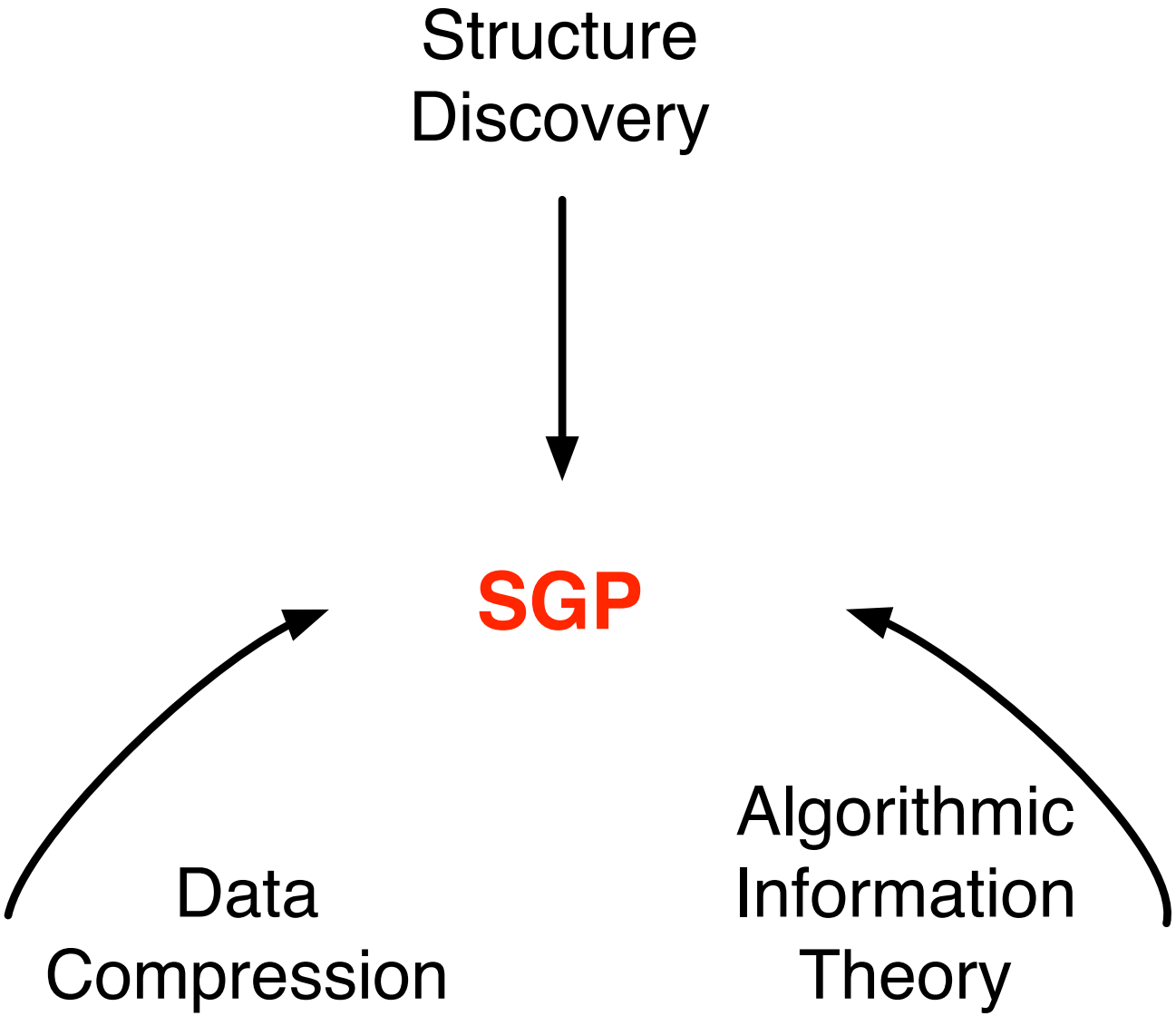
Hardness

This is a NP-Hard problem[a]

[a] Storer & Szymanski. "Data Compression via Textual Substitution" J of ACM

Charikar, et al. "The smallest grammar problem" 2005. IEEE Transactions on Information Theory

A Generic Problem



SGP: 3 Applications

Structure Discovery (SG)

Find the explanation of a coherent body of data.
SGP: The smallest parse tree is the one that captures the best all regularities

Data Compression (DC)

Encoding information using fewer bits than the original representation.
SGP: Instead of encoding a sequence, encode a smallest grammar for this sequence

Algorithmic Information Theory (AIT)

Relationship between information theory and computation. Kolmogorov Complexity of s = size of smallest Turing Machine that outputs s.
SGP: Change unrestricted grammar by context-free grammar to go from uncomputable to intractable

Timeline

1972 Structural Information Theory AIT

Klix, Scheidereiter, Organismische Informationsverarbeitung

1975 SD in Natural Language

Wolff, An algorithm for the segmentation of an artificial language analogue

1980 Complexity of bio sequences AIT

Ebeling, Jiménez-Montaño, On grammars, complexity, and information measures of biological macromolecules

1982 Macro-schemas DC

Storer & Szymanski, Data Compression via Textual Substitution

1996 Sequitur SD

Nevill-Manning & Witten, Compression and Explanation using Hierarchical Grammars

1998 Greedy offline algorithm DC

Apostolico & Lonardi, Off-line compression by greedy textual substitution

2000 Grammar-based Codes DC

Kieffer & Yang, Grammar-based codes: a new class of universal lossless source codes

2002 The SGP AIT

Charikar, Lehman, et al., The smallest grammar problem

2006 Sequitur for Grammatical Inferece SD

Eyraud, Inférence Grammaticale de Langages Hors-Contextes

2007 MDLcompress SD

Evans,et al., MicroRNA Target Detection and Analysis for Genes Related to Breast Cancer Using MDLcompress

2010 Normalized Compression Distance AIT

Cerra & Datcu, A Similarity Measure Using Smallest Context-Free Grammars

2010 Compressed Self-Indices DC

Claude & Navarro Self-indexed grammar-based compression.

Bille, et at. Random access to grammar compressed strings

Algorithmic Information Theory

1972 Structural Information Theory AIT

Klix, Scheidereiter, Organismische Informationsverarbeitung

1975 SD in Natural Language

Wolff, An algorithm for the segmentation of an artificial language analogue

1980 Complexity of bio sequences AIT

Ebeling, Jiménez-Montaño, On grammars, complexity, and information measures of biological macromolecules

1982 Macro-schemas DC

Storer & Szymanski, Data Compression via Textual Substitution

1996 Sequitur SD

Nevill-Manning & Witten, Compression and Explanation using Hierarchical Grammars

1998 Greedy offline algorithm DC

Apostolico & Lonardi, Off-line compression by greedy textual substitution

2000 Grammar-based Codes DC

Kieffer & Yang, Grammar-based codes: a new class of universal lossless source codes

2002 The SGP AIT

Charikar, Lehman, et al., The smallest grammar problem

2006 Sequitur for Grammatical Inferece SD

Eyraud, Inférence Grammaticale de Langages Hors-Contextes

2007 MDLcompress SD

Evans,et al., MicroRNA Target Detection and Analysis for Genes Related to Breast Cancer Using MDLcompress

2010 Normalized Compression Distance AIT

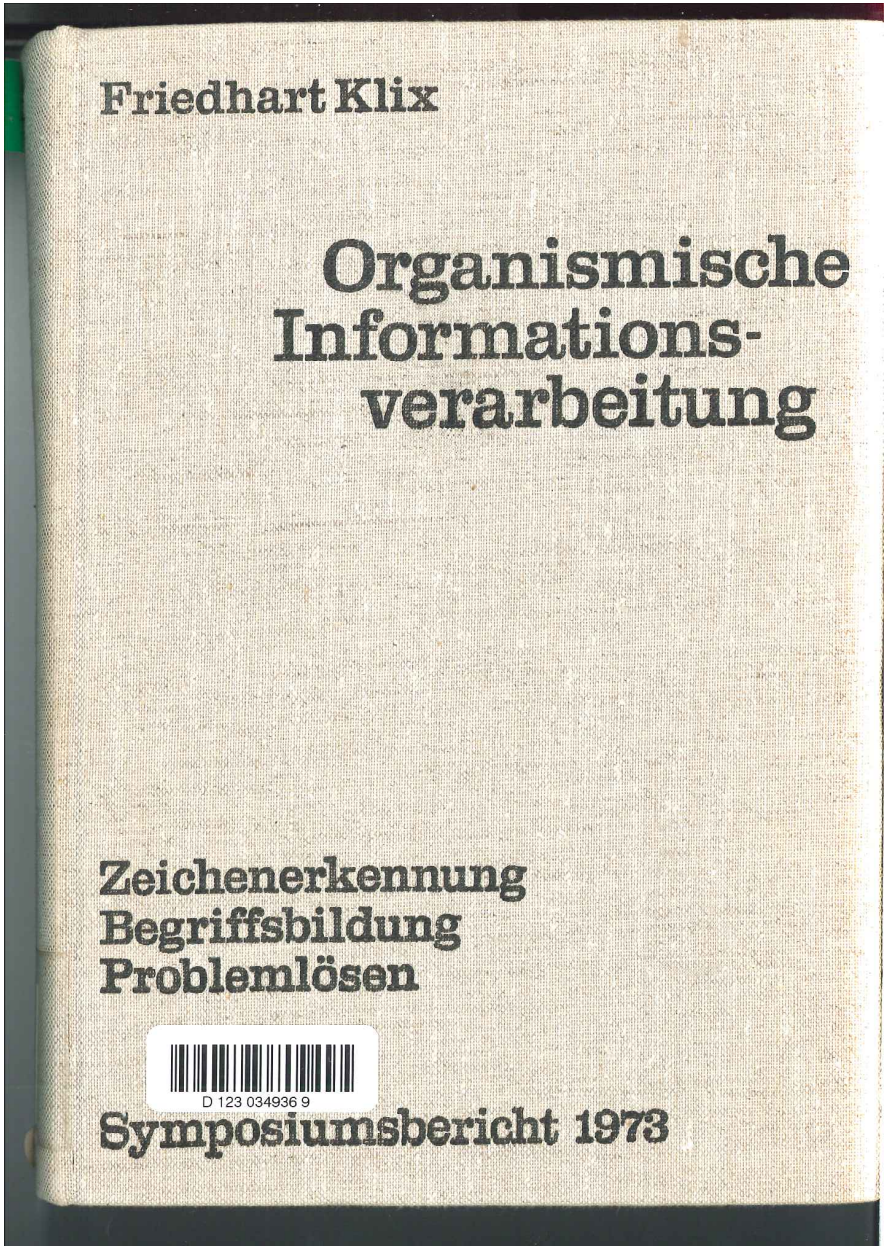
Cerra & Datcu, A Similarity Measure Using Smallest Context-Free Grammars

2010 Compressed Self-Indices DC

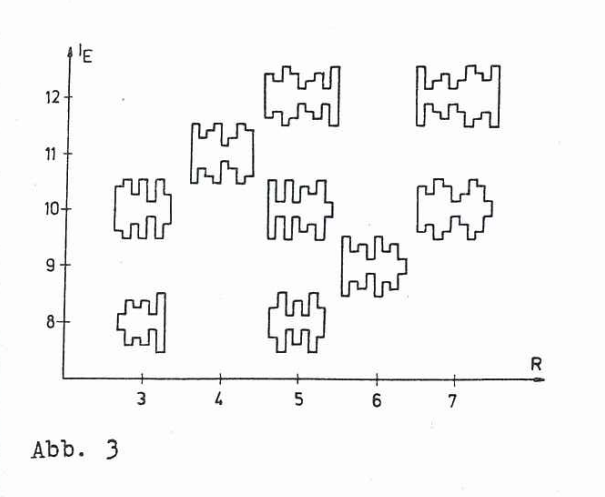
Claude & Navarro Self-indexed grammar-based compression.

Bille, et at. Random access to grammar compressed strings

Structural Information Theory



Klix, “Struktur, Strukturbeschreibung und Erkennungsleistung”



Scheidereiter, “Zur Beschreibung strukturierter Objeckte

mit kontextfreien Grammatiken”

Eine generative Grammatik ist ein Quadrupel

$G = (V_T, V_H, R, S)$

mit der Menge der terminalen Symbole (Alphabet) V_T, der Menge der Hilfssymbole (Variablen) V_H, der Menge der Ersetzungsregeln R und dem Startsymbol $S \in V_H$. Wir wollen hier den Typ des Ersetzungsregeln einschränken, indem wir nur kontextfreie Regeln zulassen, d.h. nur Regeln der Art

$$\sigma \rightarrow q \text{ mit } \sigma \in V_H,\ q \in (V_T \cup V_H) \quad .$$

Mit dieser Einschränkung kann man die Kompliziertheit einer Regel durch die Wortlänge der rechten Seite definieren:

$$K(\sigma \rightarrow q) \underset{Df.}{=} |q| \quad , \qquad (1)$$

wobei $|q|$ die Wortlänge von q bedeutet. Die Kompliziertheit eines Wortes p über dem Alphabet V_T definieren wir dann als Summe der Kompliziertheit derjenigen Regeln, die zur Ableitung des Wortes p benutzt werden, unabhängig davon, wieoft:

$$K_G(p) \underset{Df.}{=} \sum_{\sigma \in V_H} K(\sigma \rightarrow q) \qquad (2)$$

Dabei soll V_H genau die Variablen enthalten, die in der Ableitung von p vorkommen, und es soll zu jeder Variablen genau eine Regel existieren.

Information Measures of Biological Macromolecules

Ebeling, Jiménez-Montaño, "On grammars, complexity, and information measures of biological macromolecules". Mathematical Biosciences. 1980

p = TGGTGGTGGGGGAAGGATTCGAACCTTCGAAGTCGAT-
GACGGCAGATTT

AGAGTCTGCTCCCTTTGGCCGCTCGGGAACCCCACCACGG-
GTAATGCTT

TTACTGGCCTGCTCCCTTATCGGGAAGC). (14)

The alphabet consists of four letters $X = \{ACGT\}$ denoting the four bases, and the length is $l(p) = 126$.

The shortest program we were able to find for this sequence reads:

$$S \to \sigma_5\sigma_5\sigma_5\sigma_{10}G\sigma_9\sigma_{10}A\sigma_3\sigma_8\sigma_3G\sigma_6A\sigma_{11}AC\sigma_{10}CAGA\sigma_4GAG\sigma_7\sigma_1$$

$$\sigma_5\sigma_8GC\sigma_2\sigma_8\sigma_8A\sigma_8AC\sigma_{10}GT\sigma_9\sigma_{11}CT\sigma_4C\sigma_5\sigma_8\sigma_1A\sigma_2GC,$$

$$\begin{array}{ll} \sigma_1 \to \sigma_{11}C\sigma_7\sigma_8TT, & \sigma_7 \to TC, \\ \sigma_2 \to \sigma_6\sigma_{10}\sigma_9, & \sigma_8 \to CC, \\ \sigma_3 \to T\sigma_6\sigma_9, & \sigma_9 \to AA, \\ \sigma_4 \to TTTA, & \sigma_{10} \to GG, \\ \sigma_5 \to \sigma_{11}G, & \sigma_{11} \to TG, \\ \sigma_6 \to \sigma_7G. & \end{array} \qquad (15)$$

The maximal complexity is obtained from the trivial representation of the string by 63 pairs produced from 16 rules, i.e. $K_G^{\max} = 63 + 32 = 95$. We find then

$$K_G = 85, \qquad R_G = 0.105, \qquad L_G = 37.$$

Algorithmic Information Theory

1972 Structural Information Theory AIT

Klix, Scheidereiter, Organismische Informationsverarbeitung

1975 SD in Natural Language

Wolff, An algorithm for the segmentation of an artificial language analogue

1980 Complexity of bio sequences AIT

Ebeling, Jiménez-Montaño, On grammars, complexity, and information measures of biological macromolecules

1982 Macro-schemas DC

Storer & Szymanski, Data Compression via Textual Substitution

1996 Sequitur SD

Nevill-Manning & Witten, Compression and Explanation using Hierarchical Grammars

1998 Greedy offline algorithm DC

Apostolico & Lonardi, Off-line compression by greedy textual substitution

2000 Grammar-based Codes DC

Kieffer & Yang, Grammar-based codes: a new class of universal lossless source codes

2002 The SGP AIT

Charikar, Lehman, et al., The smallest grammar problem

2006 Sequitur for Grammatical Inferece SD

Eyraud, Inférence Grammaticale de Langages Hors-Contextes

2007 MDLcompress SD

Evans,et al., MicroRNA Target Detection and Analysis for Genes Related to Breast Cancer Using MDLcompress

2010 Normalized Compression Distance AIT

Cerra & Datcu, A Similarity Measure Using Smallest Context-Free Grammars

2010 Compressed Self-Indices DC

Claude & Navarro Self-indexed grammar-based compression.

Bille, et at. Random access to grammar compressed strings

Data Compression

1972 Structural Information Theory AIT

Klix, Scheidereiter, Organismische Informationsverarbeitung

1975 SD in Natural Language

Wolff, An algorithm for the segmentation of an artificial language analogue

1980 Complexity of bio sequences AIT

Ebeling, Jiménez-Montaño, On grammars, complexity, and information measures of biological macromolecules

1982 Macro-schemas DC

Storer & Szymanski, Data Compression via Textual Substitution

1996 Sequitur SD

Nevill-Manning & Witten, Compression and Explanation using Hierarchical Grammars

1998 Greedy offline algorithm DC

Apostolico & Lonardi, Off-line compression by greedy textual substitution

2000 Grammar-based Codes DC

Kieffer & Yang, Grammar-based codes: a new class of universal lossless source codes

2002 The SGP AIT

Charikar, Lehman, et al., The smallest grammar problem

2006 Sequitur for Grammatical Inferece SD

Eyraud, Inférence Grammaticale de Langages Hors-Contextes

2007 MDLcompress SD

Evans,et al., MicroRNA Target Detection and Analysis for Genes Related to Breast Cancer Using MDLcompress

2010 Normalized Compression Distance AIT

Cerra & Datcu, A Similarity Measure Using Smallest Context-Free Grammars

2010 Compressed Self-Indices DC

Claude & Navarro Self-indexed grammar-based compression.

Bille, et at. Random access to grammar compressed strings

Structure Discovery

1972 Structural Information Theory AIT

Klix, Scheidereiter, Organismische Informationsverarbeitung

1975 SD in Natural Language

Wolff, An algorithm for the segmentation of an artificial language analogue

1980 Complexity of bio sequences AIT

Ebeling, Jiménez-Montaño, On grammars, complexity, and information measures of biological macromolecules

1982 Macro-schemas DC

Storer & Szymanski, Data Compression via Textual Substitution

1996 Sequitur SD

Nevill-Manning & Witten, Compression and Explanation using Hierarchical Grammars

1998 Greedy offline algorithm DC

Apostolico & Lonardi, Off-line compression by greedy textual substitution

2000 Grammar-based Codes DC

Kieffer & Yang, Grammar-based codes: a new class of universal lossless source codes

2002 The SGP AIT

Charikar, Lehman, et al., The smallest grammar problem

2006 Sequitur for Grammatical Inferece SD

Eyraud, Inférence Grammaticale de Langages Hors-Contextes

2007 MDLcompress SD

Evans,et al., MicroRNA Target Detection and Analysis for Genes Related to Breast Cancer Using MDLcompress

2010 Normalized Compression Distance AIT

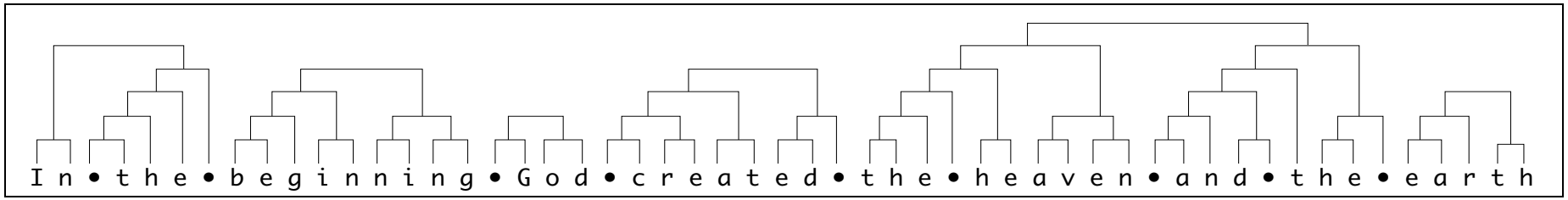
Cerra & Datcu, A Similarity Measure Using Smallest Context-Free Grammars

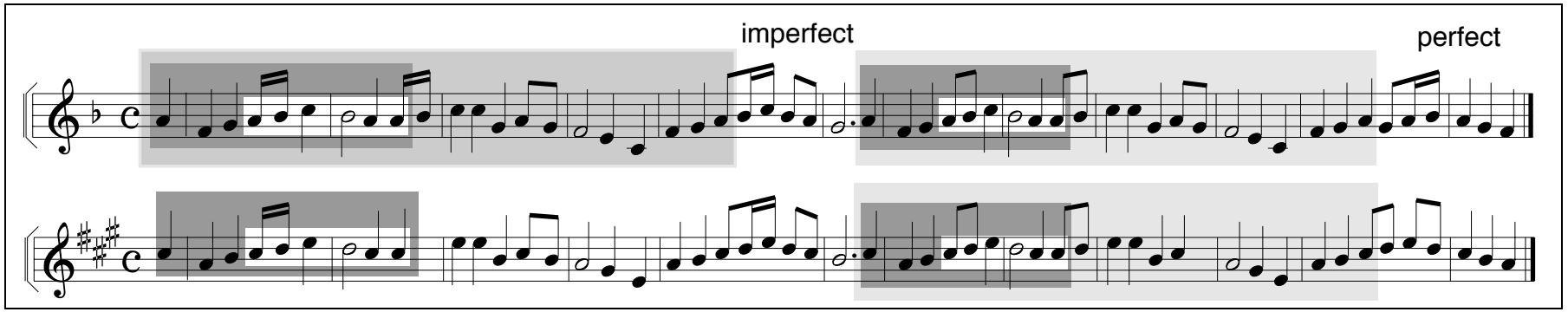
2010 Compressed Self-Indices DC

Claude & Navarro Self-indexed grammar-based compression.

Bille, et at. Random access to grammar compressed strings

Sequitur for SD





Nevill-Manning, "Inferring Sequential Structure". PhD Thesis. 1996

Used in Grammatical Inference [Eyraud, 2006]

Contributions

1. Comparison of Practical Algorithms

2. Attacking the Smallest Grammar Problem
   - What is a Word? Efficiency Issues
   - Choice of Occurrences
   - Choice of Set of Words

3. Applications: DNA Compression

Contributions

1. Comparison of Practical Algorithms

2. Attacking the Smallest Grammar Problem
   - What is a Word? Efficiency Issues
   - Choice of Occurrences
   - Choice of Set of Words

3. Applications: DNA Compression

Previous Algorithms

Previous Algorithms

- The theoretical ones Charikar,et al.05; Rytter03; Sakamoto03,04; Gagie&Gawrychowski10

Previous Algorithms

- The theoretical ones Charikar,et al.05; Rytter03; Sakamoto03,04; Gagie&Gawrychowski10
- The on-line ones : read from left to right. Ex: LZ78, Sequitur, ...
- The off-line ones : have access to the whole sequence

Off-line algorithms

An Example

$S \quad \rightarrow \quad$ how_much_wood_would_a_woodchuck_chuck_
if_a_woodchuck_could_chuck_wood?

Off-line algorithms

An Example

$S \quad \rightarrow \quad$ how_much_wood_would_a_woodchuck_chuck_
if_a_woodchuck_could_chuck_wood?

Off-line algorithms

An Example

$$
\begin{array}{lcl}
S & \longrightarrow & \text{how\_much\_wood\_would\_a\_woodchuck\_chuck\_} \\
 & & \text{if\_a\_woodchuck\_could\_chuck\_wood?} \\
 & \Downarrow & \\
S & \longrightarrow & \text{how\_much\_wood\_would}\,N_1\,\text{huck\_if}\,N_1\,\text{ould\_chuck\_wood?} \\
N_1 & \longrightarrow & \text{\_a\_woodchuck\_c}
\end{array}
$$

Off-line algorithms

An Example

$$
\begin{array}{lcl}
S & \longrightarrow & \text{how\_much\_wood\_would\_a\_woodchuck\_chuck\_} \\
 & & \text{if\_a\_woodchuck\_could\_chuck\_wood?} \\
 & \Downarrow & \\
S & \longrightarrow & \text{how\_much\_wood\_would}\,N_1\,\text{huck\_if}\,N_1\,\text{ould\_chuck\_wood?} \\
N_1 & \longrightarrow & \text{\_a\_woodchuck\_c}
\end{array}
$$

Off-line algorithms

An Example

$$
\begin{array}{lcl}
S & \longrightarrow & \text{how\_much\_wood\_would\_a\_woodchuck\_chuck\_} \\
 & & \text{if\_a\_woodchuck\_could\_chuck\_wood?} \\
 & \Downarrow & \\
S & \longrightarrow & \text{how\_much\_wood\_would}\,N_1\,\text{huck\_if}\,N_1\,\text{ould\_chuck\_wood?} \\
N_1 & \longrightarrow & \text{\_a\_woodchuck\_c} \\
 & \Downarrow & \\
S & \longrightarrow & \text{how\_much\_wood\_would}\,N_1\,\text{huck\_if\_}N_1\,\text{ould\_}N_2\,\text{wood?} \\
N_1 & \longrightarrow & \text{\_a\_wood}\,N_2\,\text{c} \\
N_2 & \longrightarrow & \text{chuck\_}
\end{array}
$$

Previous Algorithms

- The theoretical ones Charikar,et al.05; Rytter03; Sakamoto03,04; Gagie&Gawrychowski10
- The on-line ones : read from left to right. Ex: LZ78, Sequitur, ...
- The off-line ones : have access to the whole sequence :
  - **Most Frequent (MF)**: take most frequent repeat, replace all occurrences with new symbol, iterate. $f(w) = \mathrm{occ}(w)$

    Wolff "An algorithm for the segmentation of an artificial language analogue". British J of Psychology. 1975

    Jiménez-Montaño "On the syntactic structure of protein sequences and the concept of grammar complexity". B. Mathematical Biology. 1984

    Larsson & Moffat. "Offline Dictionary-Based Compression". DCC. 1999

Previous Algorithms

- The theoretical ones Charikar,et al.05; Rytter03; Sakamoto03,04; Gagie&Gawrychowski10
- The on-line ones : read from left to right. Ex: LZ78, Sequitur, ...
- The off-line ones : have access to the whole sequence :
  - **Most Frequent (MF)**: take most frequent repeat, replace all occurrences with new symbol, iterate. $f(w) = \text{occ}(w)$
  - **Maximal Length (ML)**: take longest repeat, replace all occurrences with new symbol, iterate. $f(w) = |w|$

    Bentley & McIlroy "Data compression using long common strings". DCC. 1999.

    Nakamura, et al. "Linear-Time Text Compression by Longest-First Substitution". MDPI Algorithms. 2009
  - **Most Compressive (MC)**: take repeat that compresses the best, replace with new symbol, iterate. $f(w) = (\text{occ}(w) - 1) * (|w| - 1) - 2$

    Apostolico & Lonardi. "Off-line compression by greedy textual substitution" Proceedings of IEEE. 2000

A General Framework: IRR

IRR (Iterative Repeat Replacement) framework

Input: a sequence s, a score function f

1. Initialize Grammar by $S \rightarrow s$
2. take repeat ω that maximizes f over G
3. **if** replacing ω would yield a bigger grammar than G **then**
   a. **return** G

   else
   a. replace all (non-overlapping) occurrences of ω in G by new symbol N
   b. add rule $N \rightarrow \omega$ to G
   c. goto 2

Complexity: $\mathcal{O}(n^3)$

Relative size on Canterbury Corpus

	On-line	Off-line		
sequence	Sequitur	IRR-ML	IRR-MF	IRR-MC (ref.)
alice29.txt	19.9%	37.1%	8.9%	41,000
asyoulik.txt	17.7%	37.8%	8.0%	37,474
cp.html	22.2%	21.6%	10.4%	8,048
fields.c	20.3%	18.6%	16.1%	3,416
grammar.lsp	20.2%	20.7%	15.1%	1,473
kennedy.xls	4.6%	7.7%	0.3%	166,924
lcet10.txt	24.5%	45.0%	8.0%	90,099
plrabn12.txt	14.9%	45.2%	5.8%	124,198
ptt5	23.4%	26.1%	6.4%	45,135
sum	25.6%	15.6%	11.9%	12,207
xargs.1	16.1%	16.2%	11.8%	2,006
average	*19.0%*	*26.5%*	*9.3%*	

Extends and confirms partial results of Nevill-Manning & Witten "On-Line and Off-Line Heuristics for Inferring Hierarchies of Repetitions in Sequences". 2000. Proc. of the IEEE. 80 (11)

Contributions

1. Comparison of Practical Algorithms

2. Attacking the Smallest Grammar Problem
   - What is a Word? Efficiency Issues
   - Choice of Occurrences
   - Choice of Set of Words

3. Applications: DNA Compression

Contributions

1. Comparison of Practical Algorithms

2. Attacking the Smallest Grammar Problem
   - What is a Word? Efficiency Issues
   - Choice of Occurrences
   - Choice of Set of Words

3. Applications: DNA Compression

What is a word?

Something repeated

$$S \rightarrow \text{how\_much\_wood\_would\_a\_woodchuck\_chuck\_if\_a\_woodchuck\_could\_chuck\_wood?}$$

A Taxonomy of Repeats

- **simple repeats**: a string that occurs more than 2 times
- **maximal repeats**: a repeat that cannot be extended

$$MR(s) = \{w : \nexists\, w' \in \mathcal{R}(s) : \forall o \in Occ(w) : \forall o' \in Occ(w') : o \nsubseteq o'\}$$

- **super-maximal repeats**: a MR that is not contained in another one

$$SMR(s) = \{w : \nexists\, w' \in \mathcal{R}(s) : \exists o \in Occ(w) : \forall o' \in Occ(w') : o \nsubseteq o'\}$$

$$= \{w : \forall\, w' \in \mathcal{R}(s) : \nexists o \in Occ(w) : \forall o' \in Occ(w') : o \nsubseteq o'\}$$

- **largest-maximal repeats**: a MR that has at least one *occurrence* not covered by another one:

$$LMR(s) = \{w : \exists\, w' \in \mathcal{R}(s) : \nexists o \in Occ(w) : \forall o' \in Occ(w') : o \nsubseteq o'\}$$

What we like of $[\epsilon|L|S]MR$

Worst Case Behavior

	$\#$	$\sum \#Occ$
r	$\Theta(n^2)$	$\Theta(n^2)$
mr	$\Theta(n)$	$\Theta(n^2)$
lmr	$\Theta(n)$	$\Omega(n^{\frac{3}{2}})$
smr	$\Theta(n)$	$\Theta(n)$

Efficiency: Accelerating IRR

- IRR computes score on each word in each iteration
- Score functions: $f = f(|w|, \mathrm{occ}(w))$

Efficiency: Accelerating IRR

- IRR computes score on each word in each iteration
- Score functions: $f = f(|w|, \mathrm{occ}(w))$

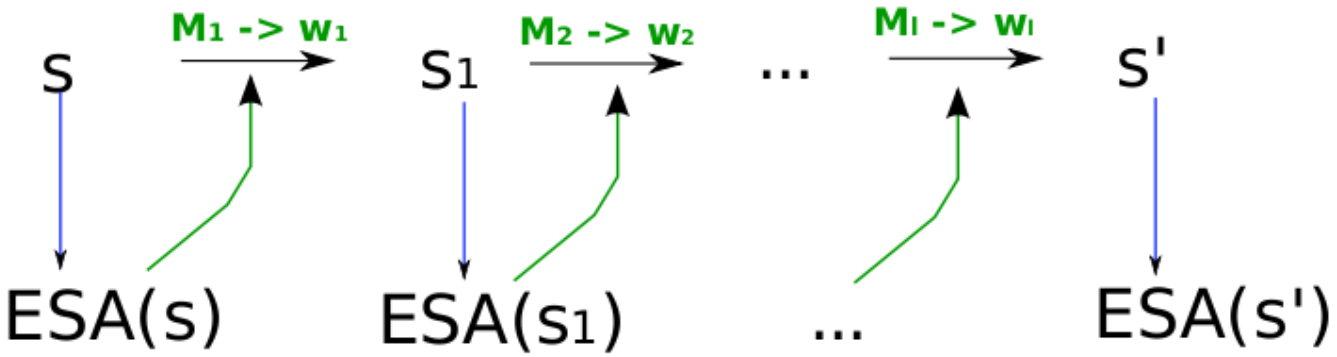
1. by using maximal repeats we reduce IRR from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ with equivalent final grammar size
2. We use an Enhanced Suffix Array to compute these scores

Efficiency: Accelerating IRR

- IRR computes score on each word in each iteration
- Score functions: $f = f(|w|, \mathrm{occ}(w))$

1. by using maximal repeats we reduce IRR from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ with equivalent final grammar size
2. We use an Enhanced Suffix Array to compute these scores

Inplace update of enhanced suffix array[1]



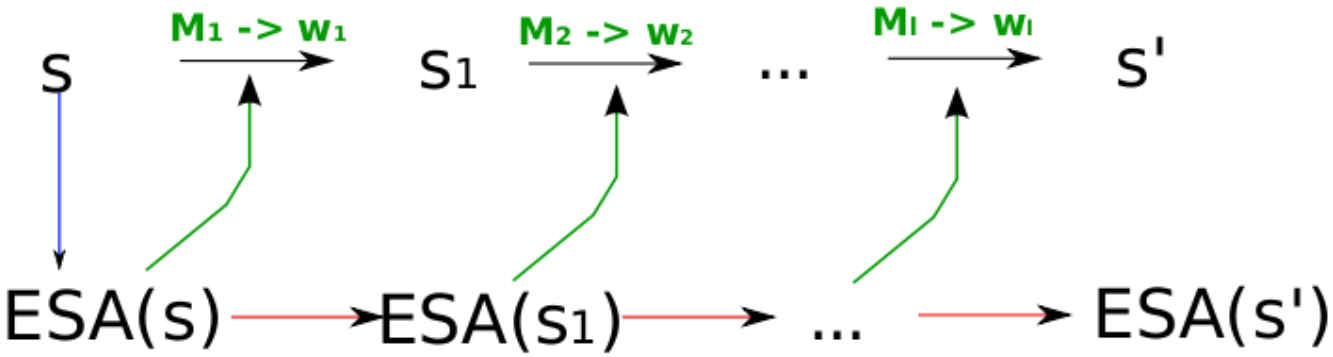
[1] "In-Place Update of Suffix Array While Recoding Words" 2009. IJFCS 20 (6)

Efficiency: Accelerating IRR

- IRR computes score on each word in each iteration
- Score functions: $f = f(|w|, \mathrm{occ}(w))$

1. by using maximal repeats we reduce IRR from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$ with equivalent final grammar size
2. We use an Enhanced Suffix Array to compute these scores

Inplace update of enhanced suffix array[1]



Up to 70x speed-up (depending on the score function) More

[1] "In-Place Update of Suffix Array While Recoding Words" 2009. IJFCS 20 (6)

Contributions

1. Comparison of Practical Algorithms

2. Attacking the Smallest Grammar Problem
   - What is a Word? Efficiency Issues
   - Choice of Occurrences
   - Choice of Set of Words

3. Applications: DNA Compression

A General Framework: IRR

IRR (Iterative Repeat Replacement) framework

Input: a sequence s, a score function f

1. Initialize Grammar by $S \rightarrow s$
2. take repeat ω that maximizes f over G
3. **if** replacing ω would yield a bigger grammar than G **then**
   a. **return** G

   else
   a. replace all (non-overlapping) occurrences of ω in G by new symbol N
   b. add rule $N \rightarrow \omega$ to G
   c. goto 2

Choice of Occurrences

The Minimal Grammar Parsing (MGP) Problem

Given a sequence s and a set of words C, find a smallest straight-line grammar for s whose constituents (words) are C.

Choice of Occurrences

The Minimal Grammar Parsing (MGP) Problem

Given a sequence s and a set of words C, find a smallest straight-line grammar for s whose constituents (words) are C.

- $\neq$ Smallest Grammar Problem: in MGP words are given
- $\neq$ Static Dictionary Parsing [Schuegraf 74]: in MGP words have also to be parsed

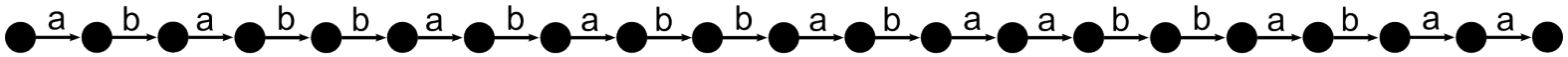
MGP: Solution

Given sequences $s = ababbababbabaabbabaa$, $C = \{abbaba, bab\}$

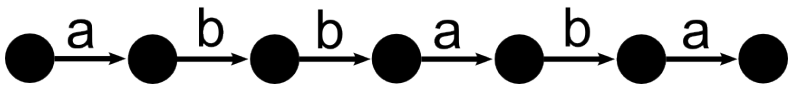
MGP: Solution

Given sequences $s = ababbababbabaabbabaa$, $C = \{abbaba, bab\}$

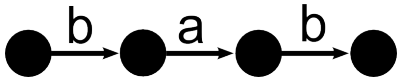
N_0



N_1



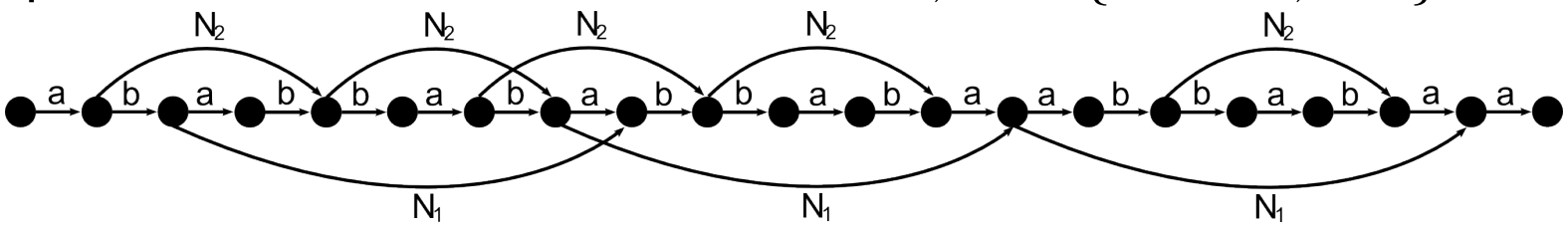
N_2



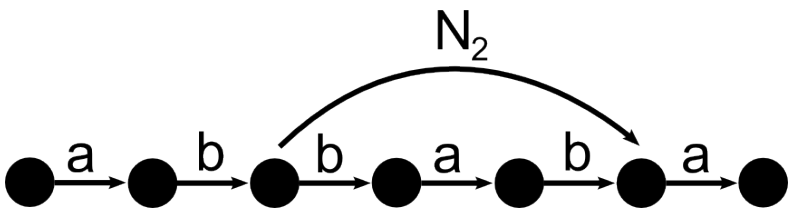
MGP: Solution

Given sequences $s = ababbababbabaabbabaa$, $C = \{abbaba, bab\}$

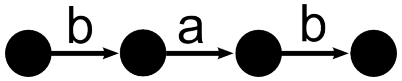
N_0



N_1



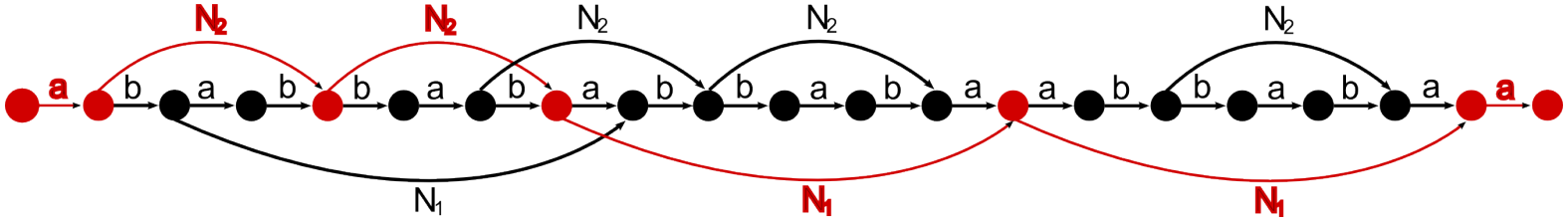
N_2



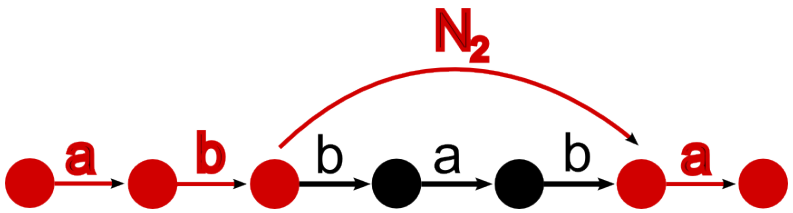
MGP: Solution

Given sequences $s = ababbababbabaabbabaa$, $C = \{abbaba, bab\}$

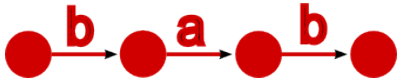
N_0



N_1



N_2



A minimal grammar for $\langle s, C \rangle$ is

$$
\begin{aligned}
N_0 &\rightarrow aN_2N_2N_1N_1a \\
N_1 &\rightarrow abN_2a \\
N_2 &\rightarrow bab
\end{aligned}
$$

Choice of Occurrences

The Minimal Grammar Parsing (MGP) Problem

Given a sequence s and a set of words C, find a smallest straight-line grammar for s whose constituents (words) are C.

- $\neq$ Smallest Grammar Problem: in MGP words are given
- $\neq$ Static Dictionary Parsing [Schuegraf 74]: in MGP words have also to be parsed

Complexity

mgp can be computed in $\mathcal{O}(n^3)$

Split the Problem

$$SGP = \begin{cases} 1. \text{ Find an optimal set of words C} \\ 2. \ \textit{mgp} \text{ (s,C)} \end{cases}$$

Split the Problem

$$SG(s) = mgp\left(\underset{C \subseteq \mathcal{R}(s)}{\text{argmin}}\left(|mgp(s, C)|\right)\right)$$

Contributions

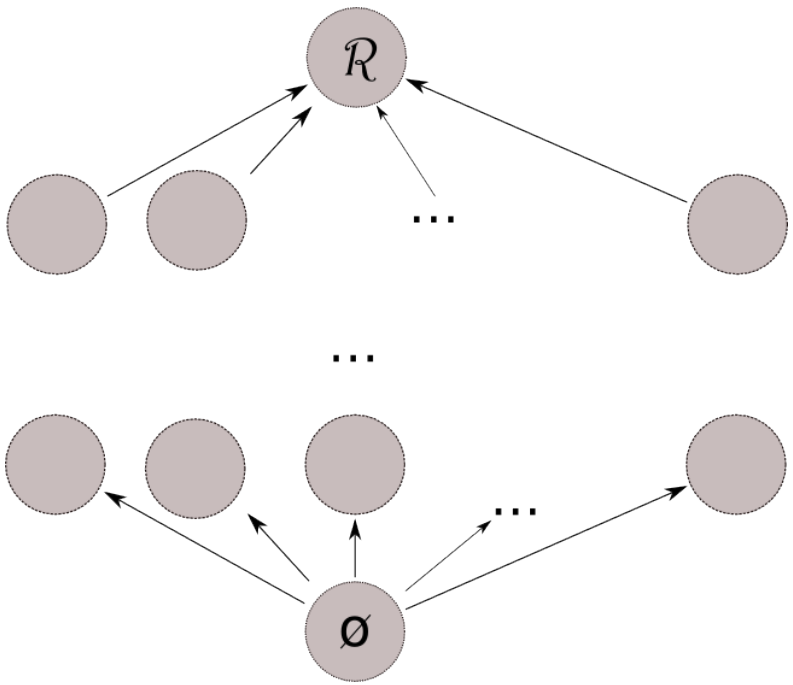
1. Comparison of Practical Algorithms

2. Attacking the Smallest Grammar Problem
   - What is a Word? Efficiency Issues
   - Choice of Occurrences
   - Choice of Set of Words

3. Applications: DNA Compression

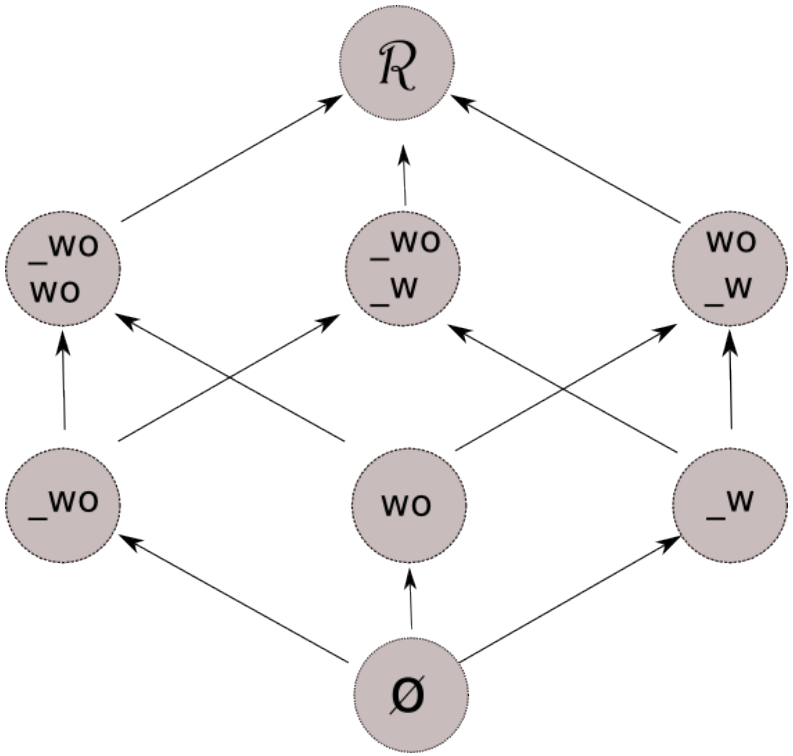
A Search Space for the SGP

Given s, take the lattice $\langle 2^{\mathcal{R}(s)}, \subseteq \rangle$ and associate a score to each node C: the size of the grammar $mgp(s, C)$.



A Search Space for the SGP: Example

Given $s =$ "how_much_wood_would", $\mathcal{R}(s) = \{\_wo, \_w, wo\}$



Lattice is a good search space

Theorem

The general SGP cannot be solved by IRR.

There exists a sequence s such that for any score function f, $IRR(s, f)$ does not return a smallest grammar. Example

Theorem

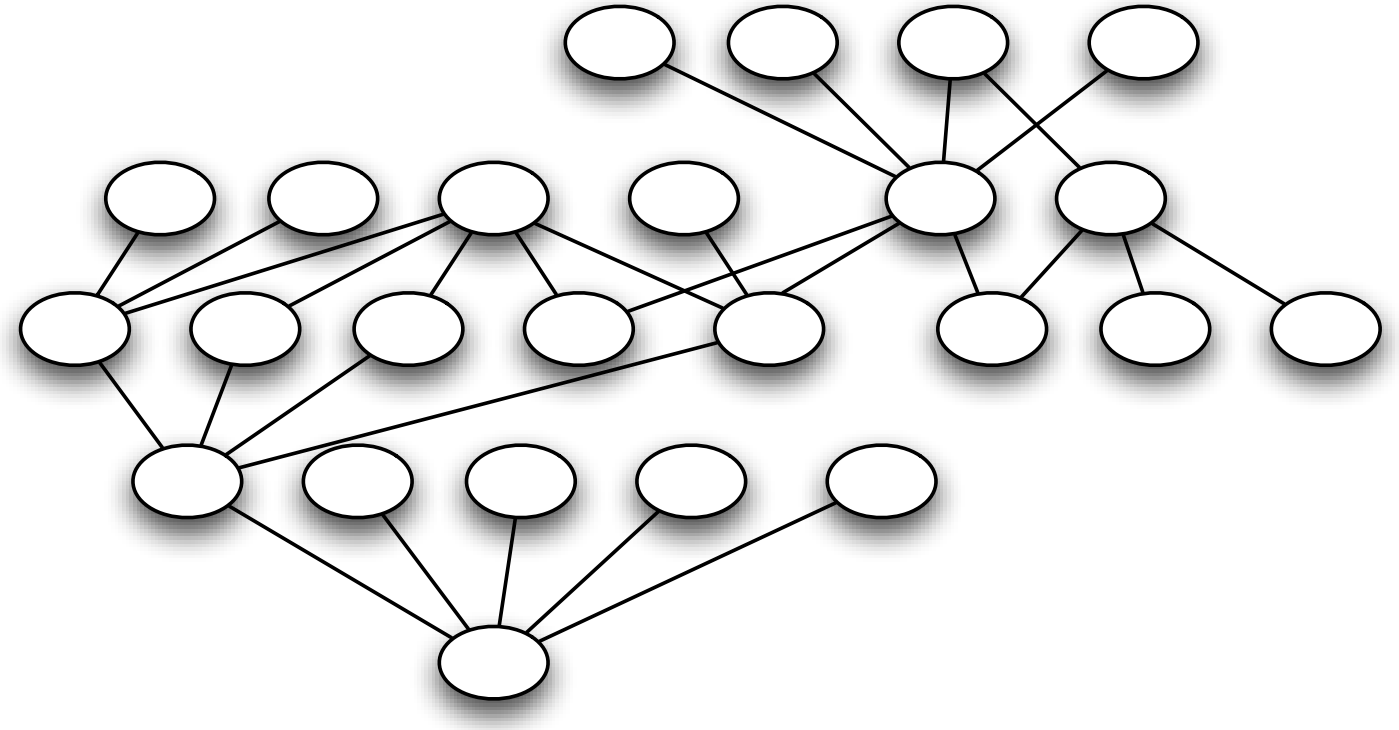
$\langle 2^{\mathcal{R}(s)}, \subseteq \rangle$ is a complete and correct search space for the SGP[a]

$$\mathcal{SG}(s) = \bigcup_{C : C \text{ is global minimum of } \langle 2^{\mathcal{R}(s)}, \subseteq \rangle} \mathcal{MGP}(s, C)$$

[a] "The Smallest Grammar Problem as Constituents Choice and Minimal Grammar Parsing" 2011 *Submitted*

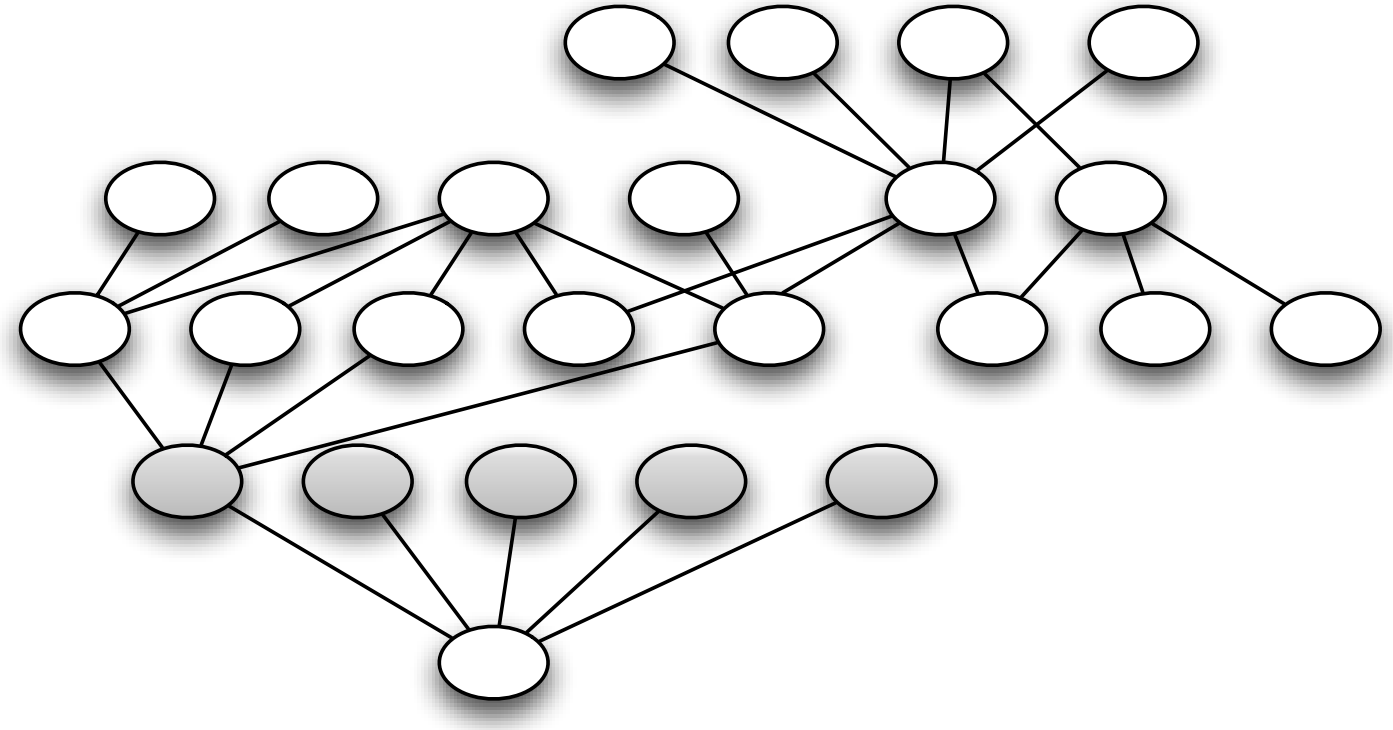
Choice of Words: Hill-climbing

Hill Climbing: given node C, compute scores of nodes $C \cup \{w_i\}$ and take node with smallest score.



Choice of Words: Hill-climbing

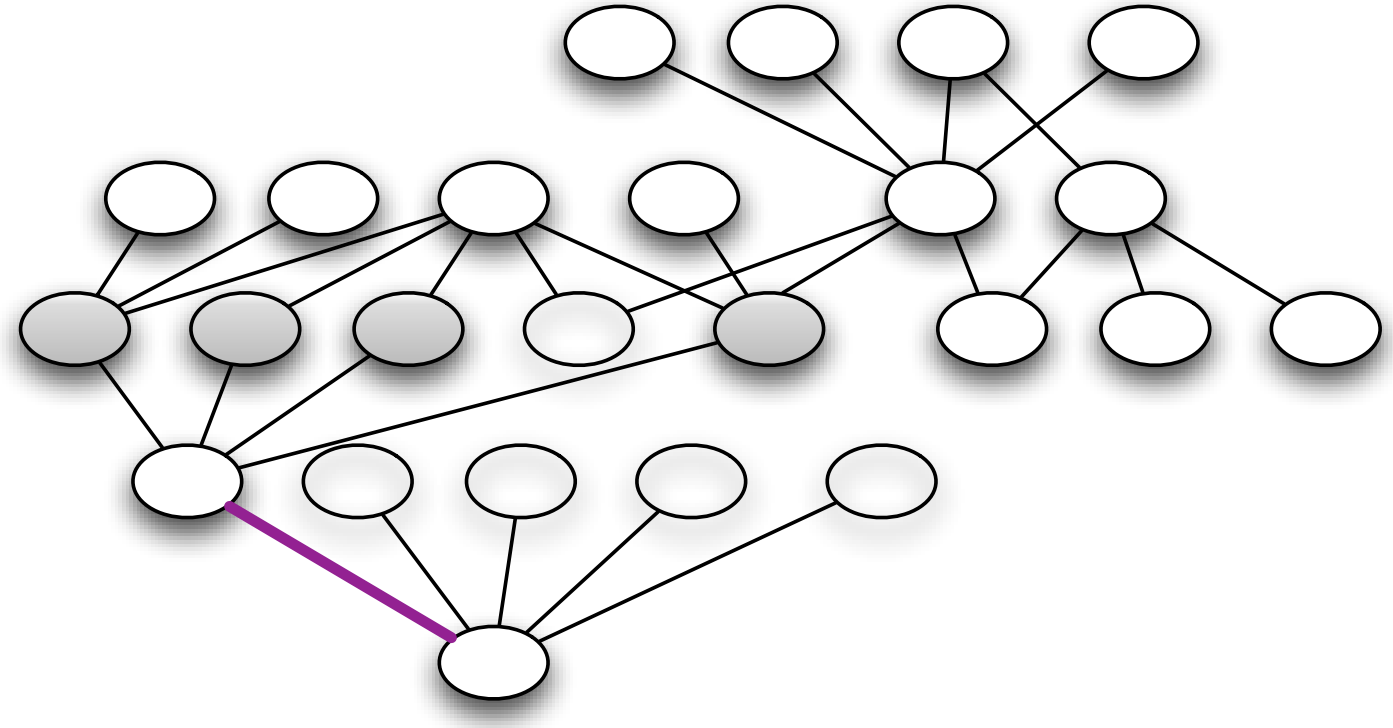
Hill Climbing: given node C, compute scores of nodes $C \cup \{w_i\}$ and take node with smallest score.



: *mgp*

Choice of Words: Hill-climbing

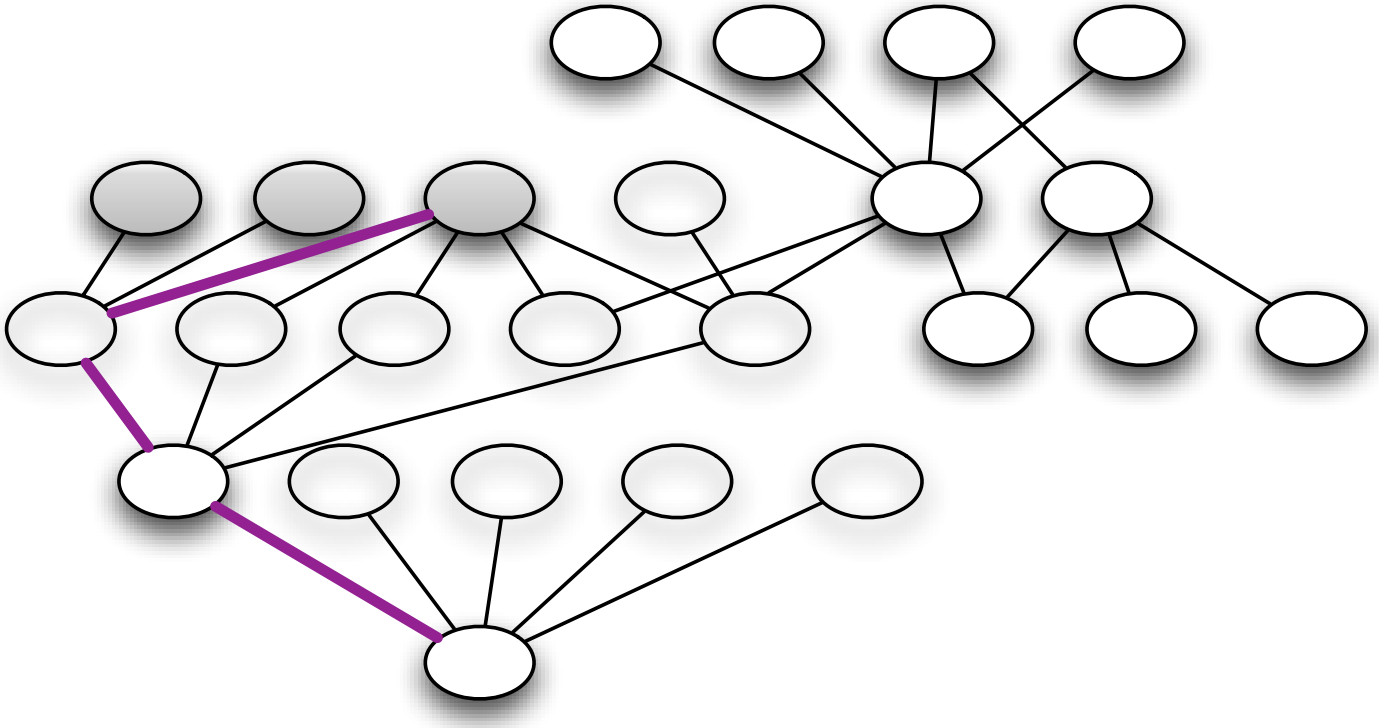
Hill Climbing: given node C, compute scores of nodes $C \cup \{w_i\}$ and take node with smallest score.



: *mgp*

Choice of Words: Hill-climbing

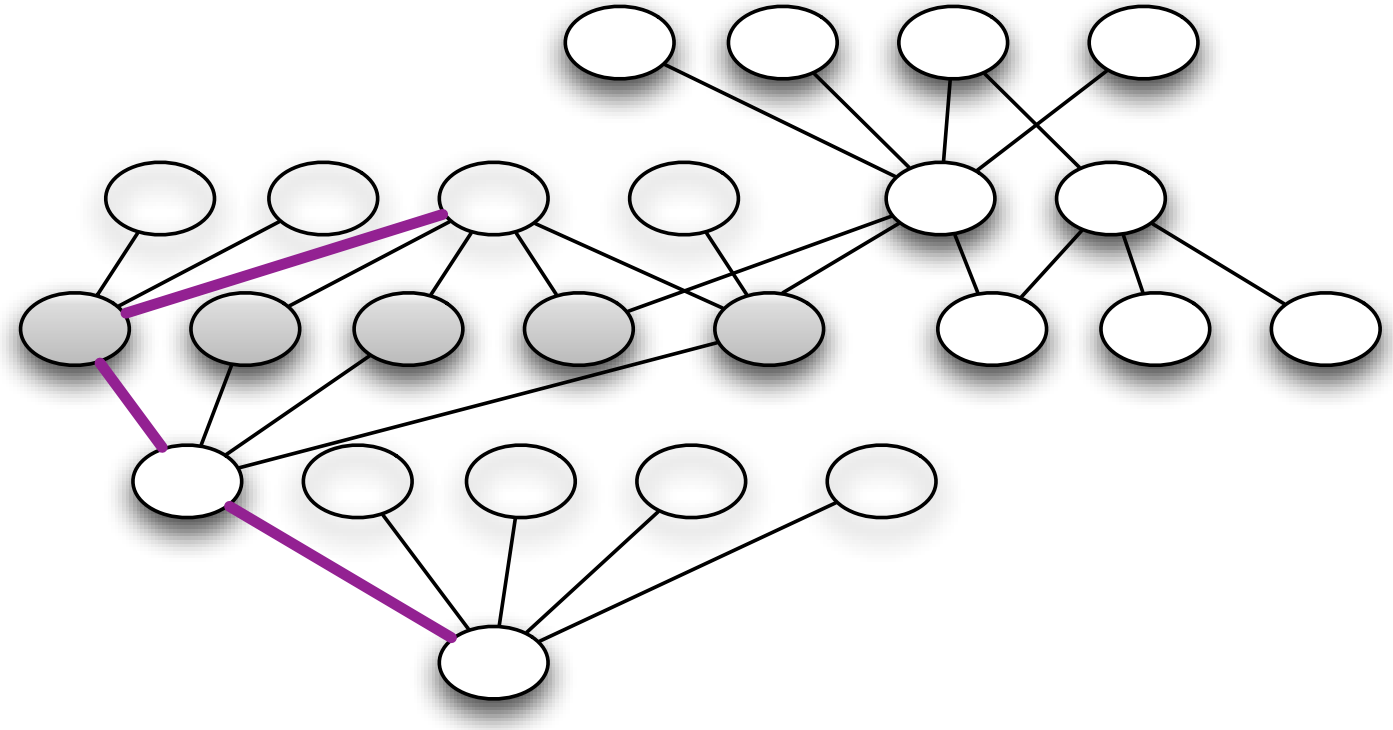
Hill Climbing: given node C, compute scores of nodes $C \cup \{w_i\}$ and take node with smallest score.



: *mgp*

Choice of Words: Hill-climbing

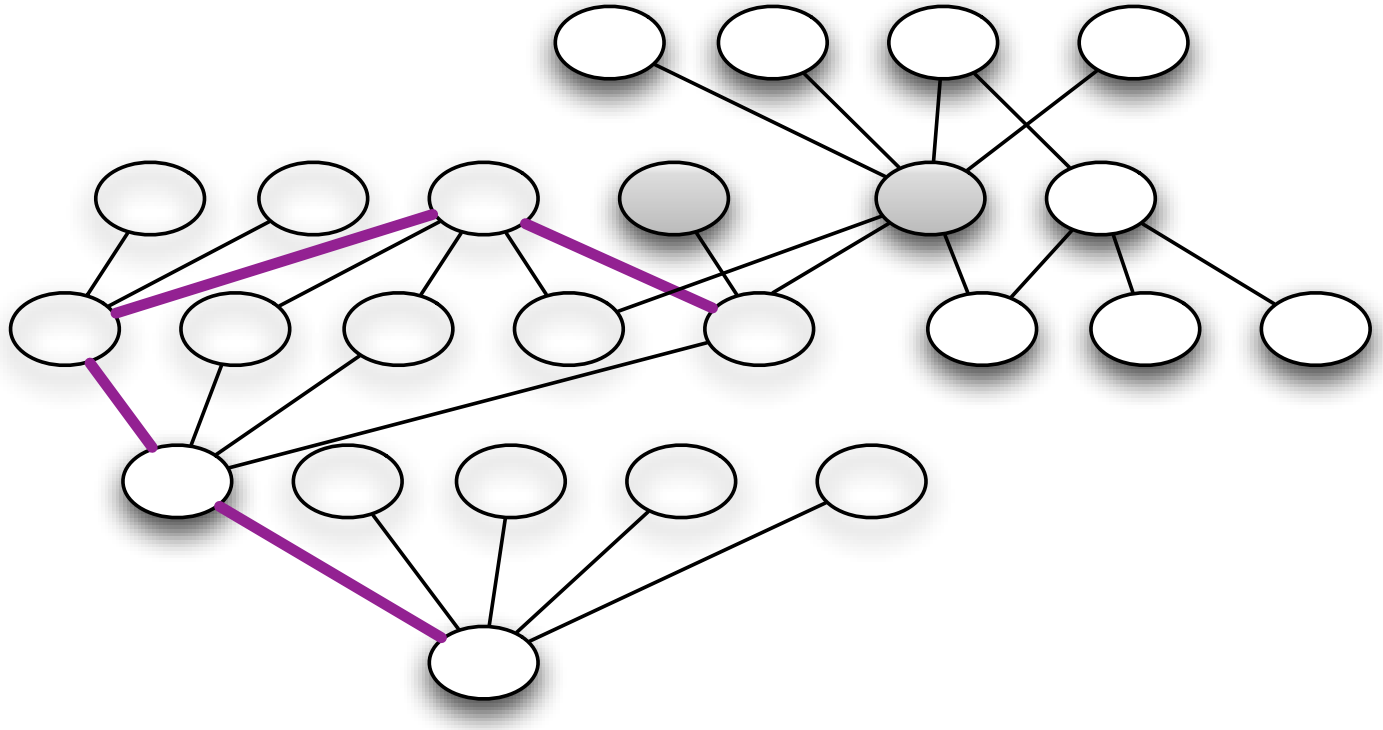
Hill Climbing: given node C, compute scores of nodes $C \cup \{w_i\}$ and take node with smallest score.
We can also go down: given node C, compute scores of nodes $C \setminus \{w_i\}$ and take node with smallest score



: *mgp*

Choice of Words: Hill-climbing

Hill Climbing: given node C, compute scores of nodes $C \cup \{w_i\}$ and take node with smallest score.
We can also go down: given node C, compute scores of nodes $C \setminus \{w_i\}$ and take node with smallest score



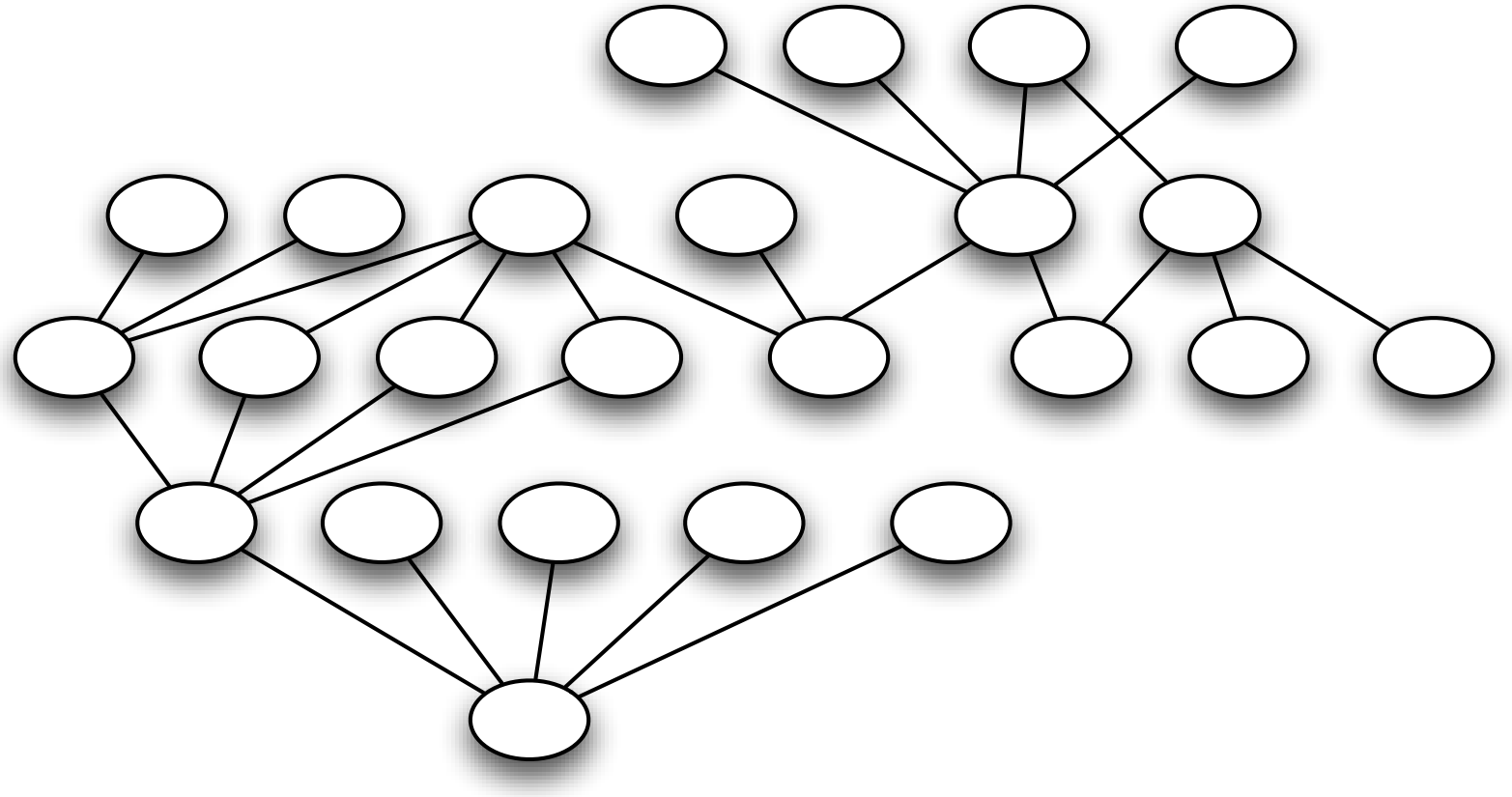
ZZ: succession of both phases. Is in $\mathcal{O}(n^7)$

Results of ZZ wrt IRR-MC

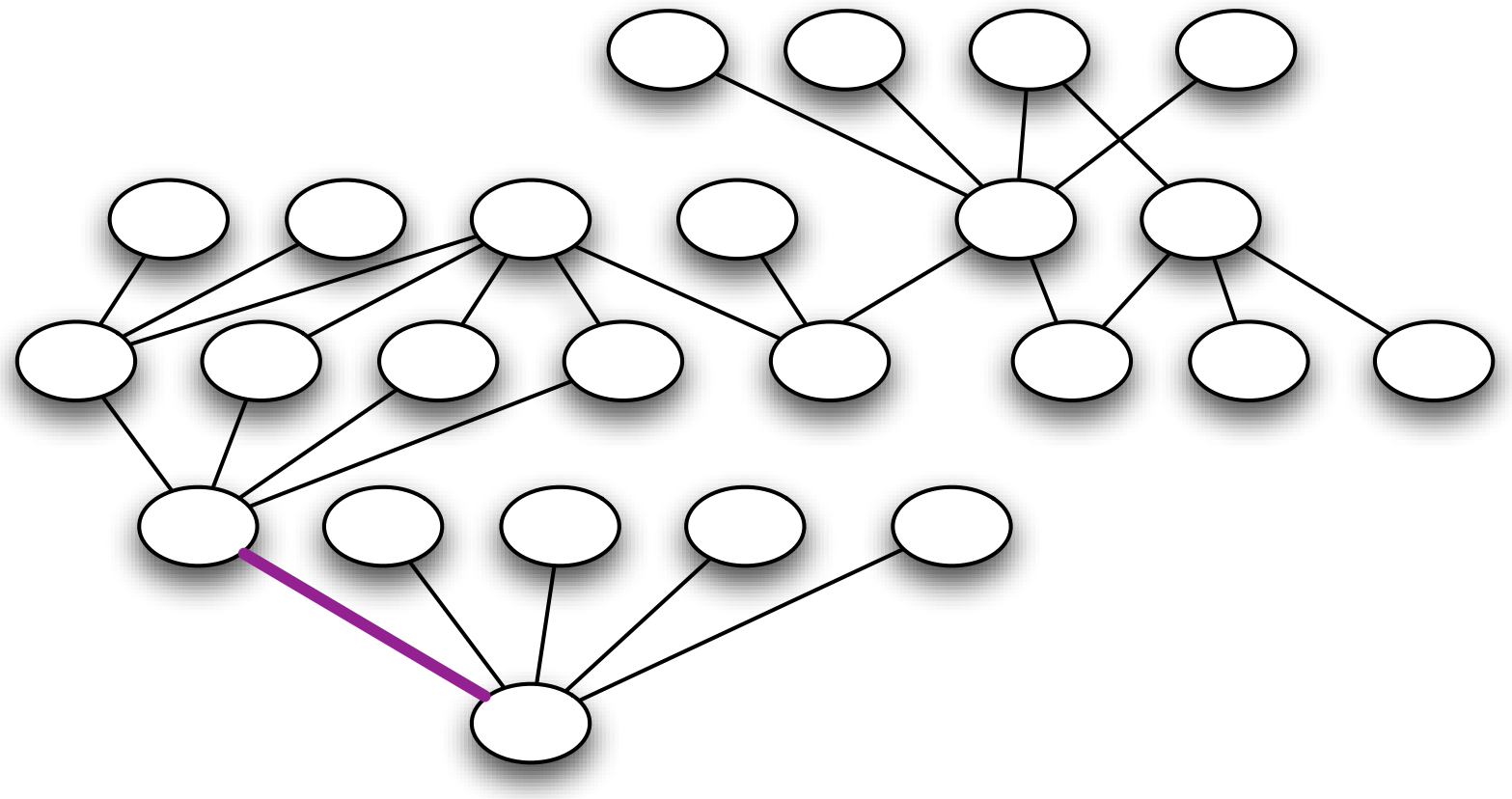
sequence	size	IRR-MC	ZZ
chmpxx	121Knt	28,706	-9.35%
chntxx	156Knt	37,885	-10.41%†
hehcmv	156Knt	53,696	-10.07%
humdyst	39Knt	11,066	-8.93%
humghcs	229Knt	12,933	-6.97%
humhbb	39Knt	18,705	-8.99%
humhdab	66Knt	15,327	-8.7%
humprtb	73Knt	14,890	-8.27%
mpomtcg	59Knt	44,178	-9.66%
mtpacga	57Knt	24,555	-9.64%
vaccg	192Knt	43,701	-10.08%†
average			-9.19%

†: partial result (execution of ZZ was interrupted)

Choice of Words: Size-Efficiency Tradeoff

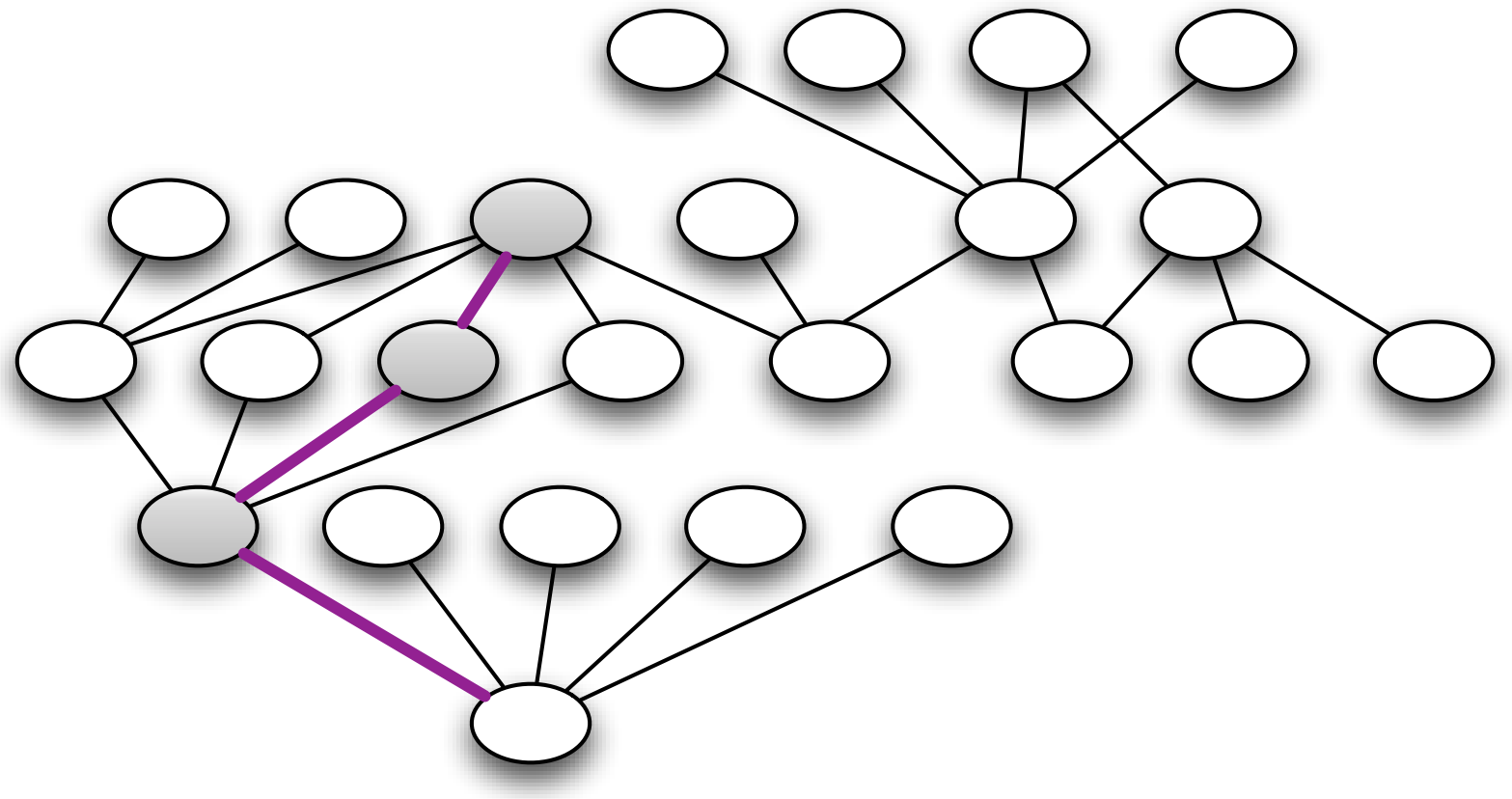


Choice of Words: Size-Efficiency Tradeoff



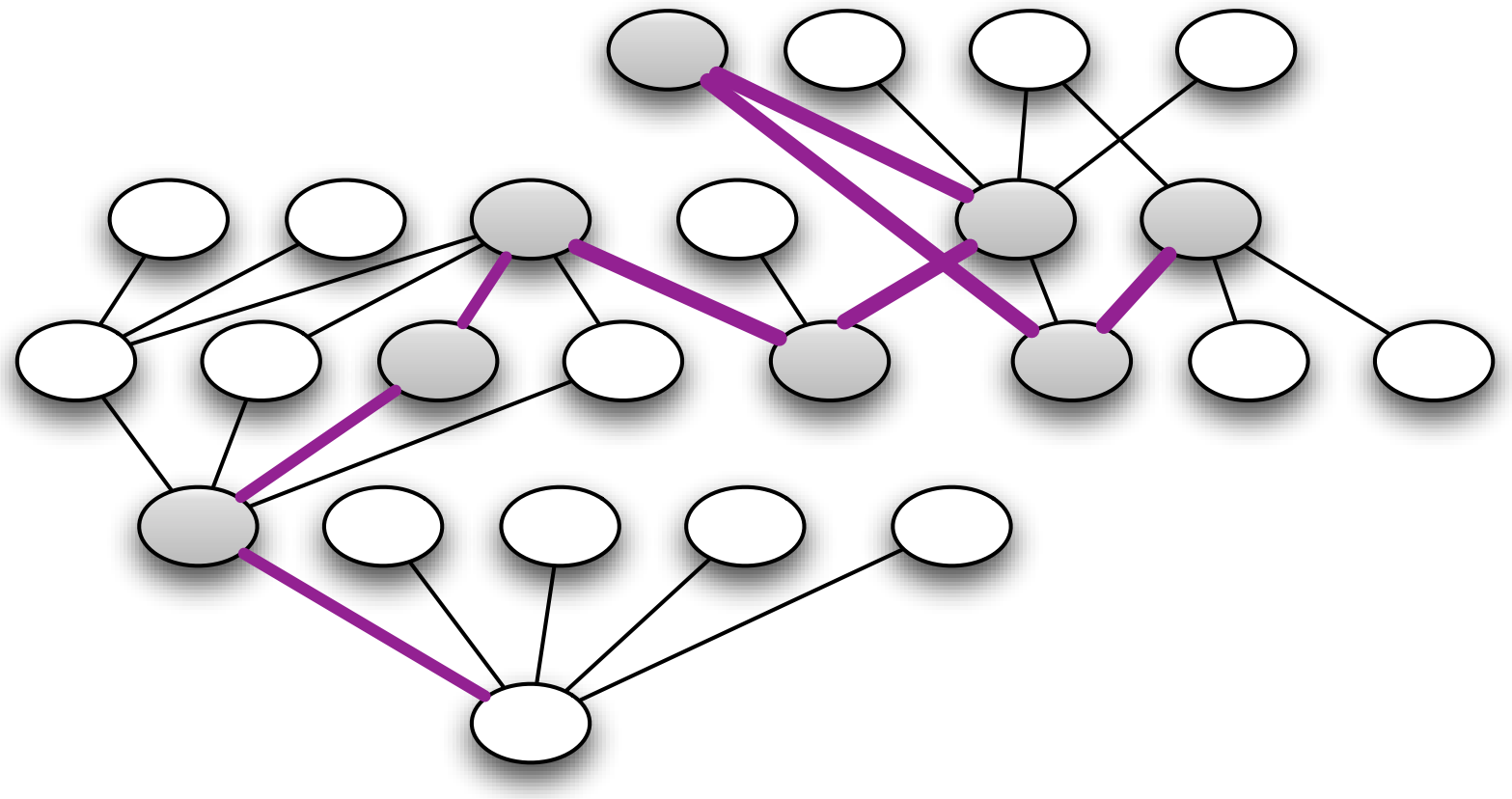
IRRCOO: uses only current state to chose next node

Choice of Words: Size-Efficiency Tradeoff



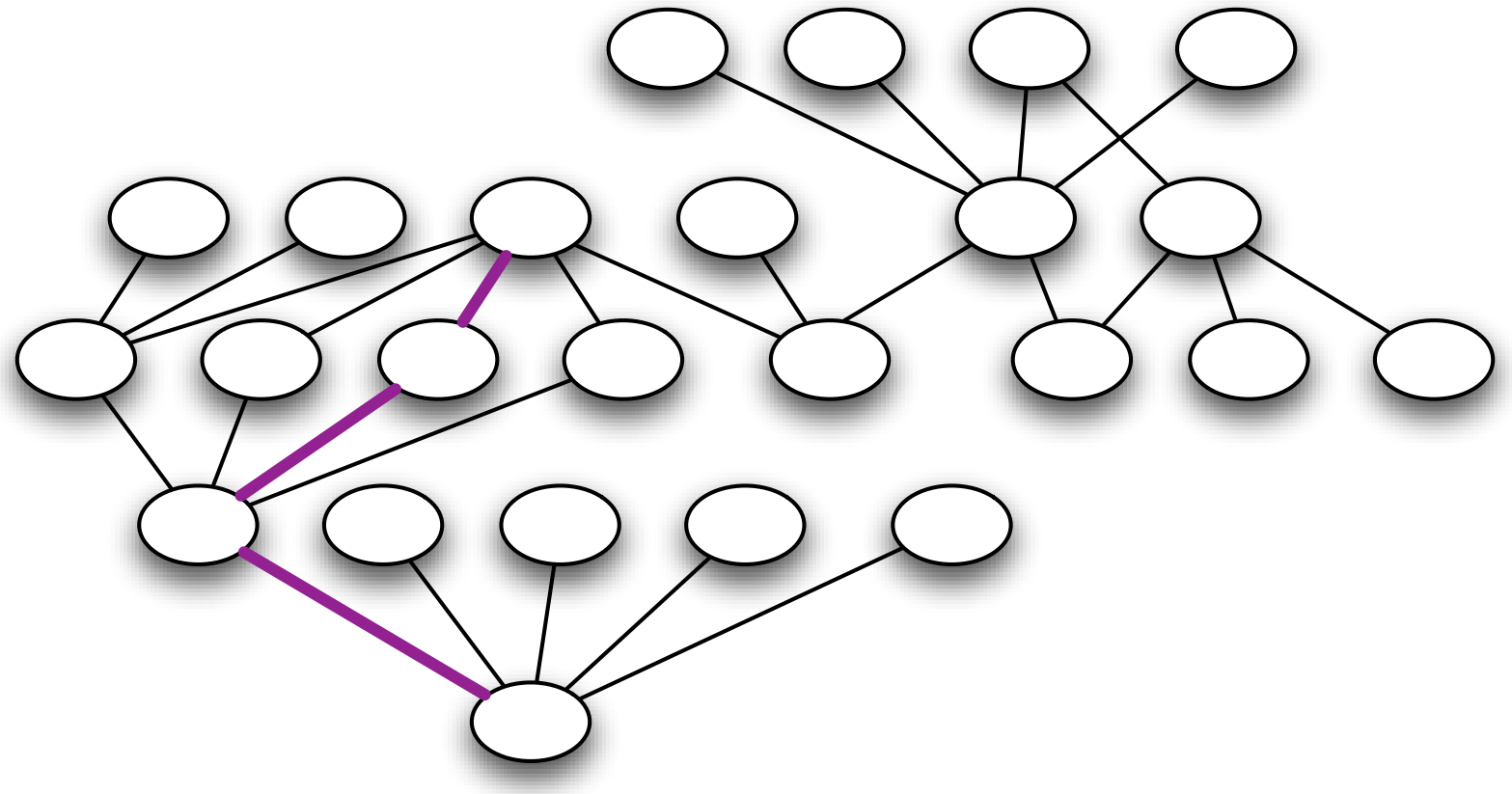
IRRCOO: uses only current state to chose next node

Choice of Words: Size-Efficiency Tradeoff



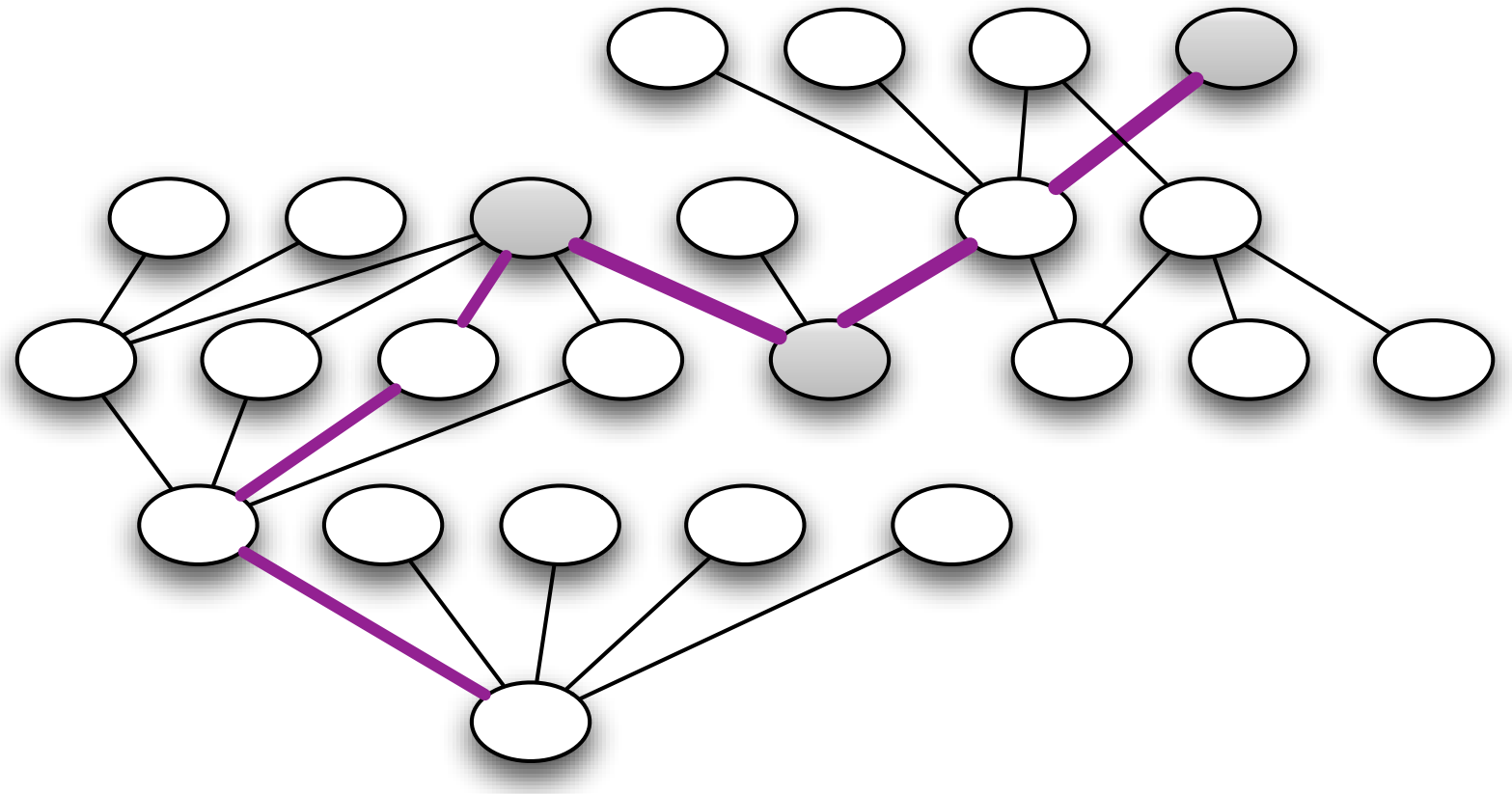
IRRCOOC: IRRCOO + clean-up

Choice of Words: Size-Efficiency Tradeoff



IRRMGP* = (IRR-MC + MGP + cleanup)*

Choice of Words: Size-Efficiency Tradeoff



IRRMGP* = (IRR-MC + MGP + cleanup)*

Results: IRRMGP* on big sequences

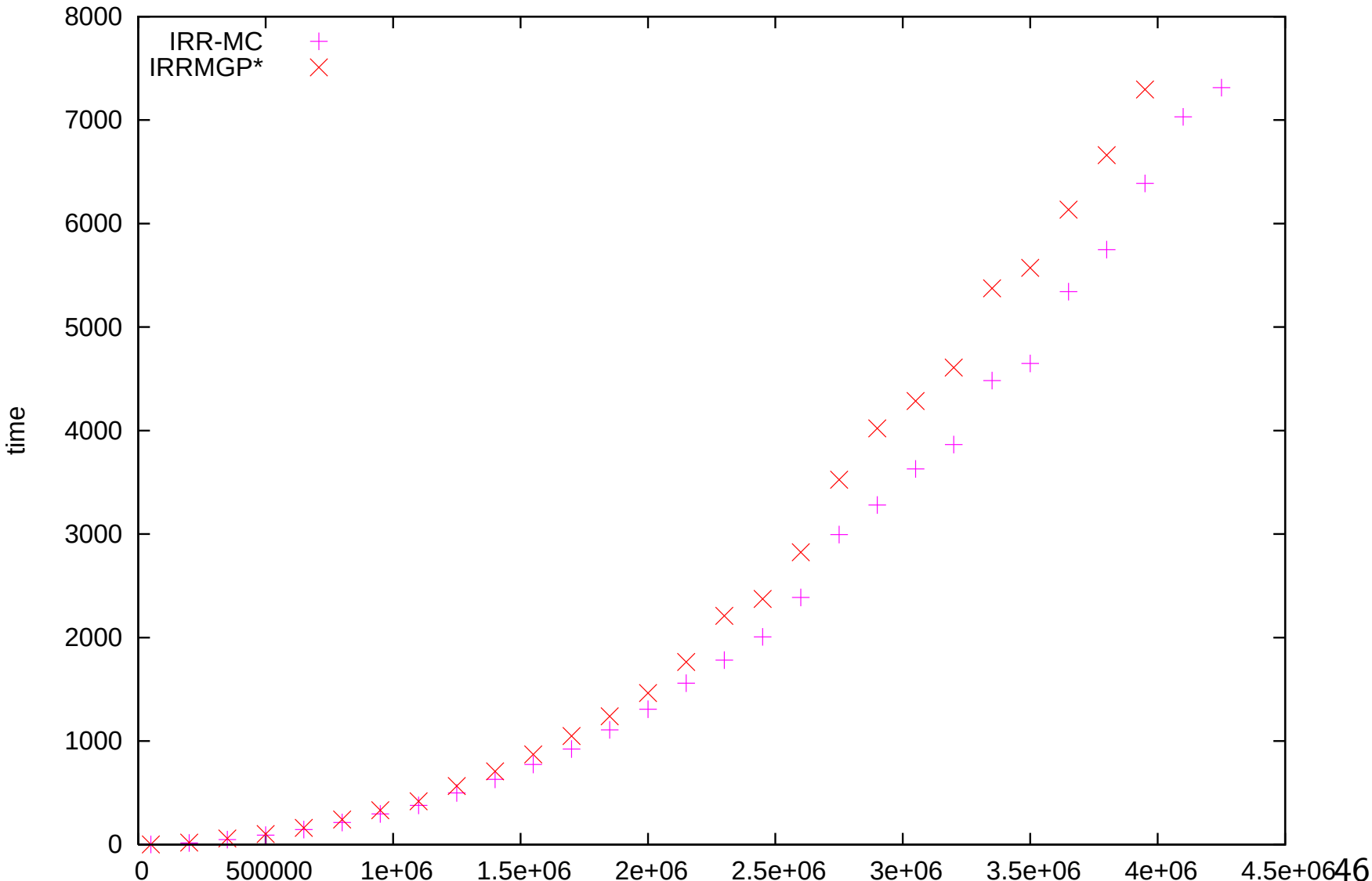
Classi-fication	sequence name	length	IRRMGP*[2]	size im-provement
Virus	P. lambda	48 Knt	13,061	-4.25%
Bacterium	E. coli	4.6 Mnt	741,435	-8.82%
Protist	T. pseudonana chrI	3 Mnt	509,203	-8.15%
Fungus	S. cerevisiae	12.1 Mnt	1,742,489	-9.68%
Alga	O. tauri	12.5 Mnt	1,801,936	-8.78%
Plant	A. Thal. chrIV	18.6 Mnt	2,561,906	-9.94%
Nematoda	C. Eleg. chrIII	13.8 Mnt	1,897,290	-9.47%

IRRMGP* scales up on bigger sequence finding close to 10% smaller grammars than state of the art.

[2] "Searching for Smallest Grammars on DNA Sequences" 2011 JDA

More Results

bytes vs. seconds



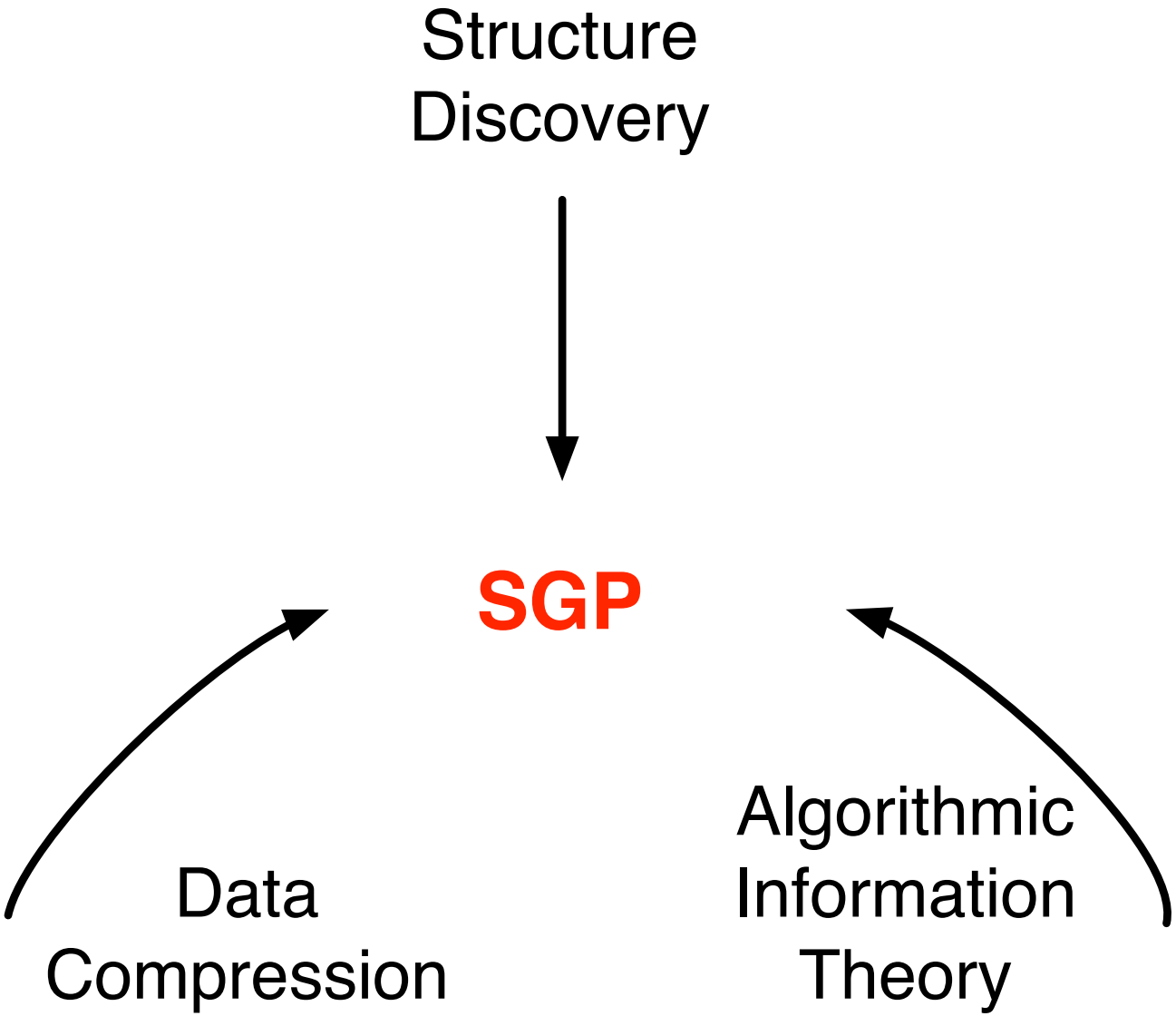
Contributions

1. Comparison of Practical Algorithms

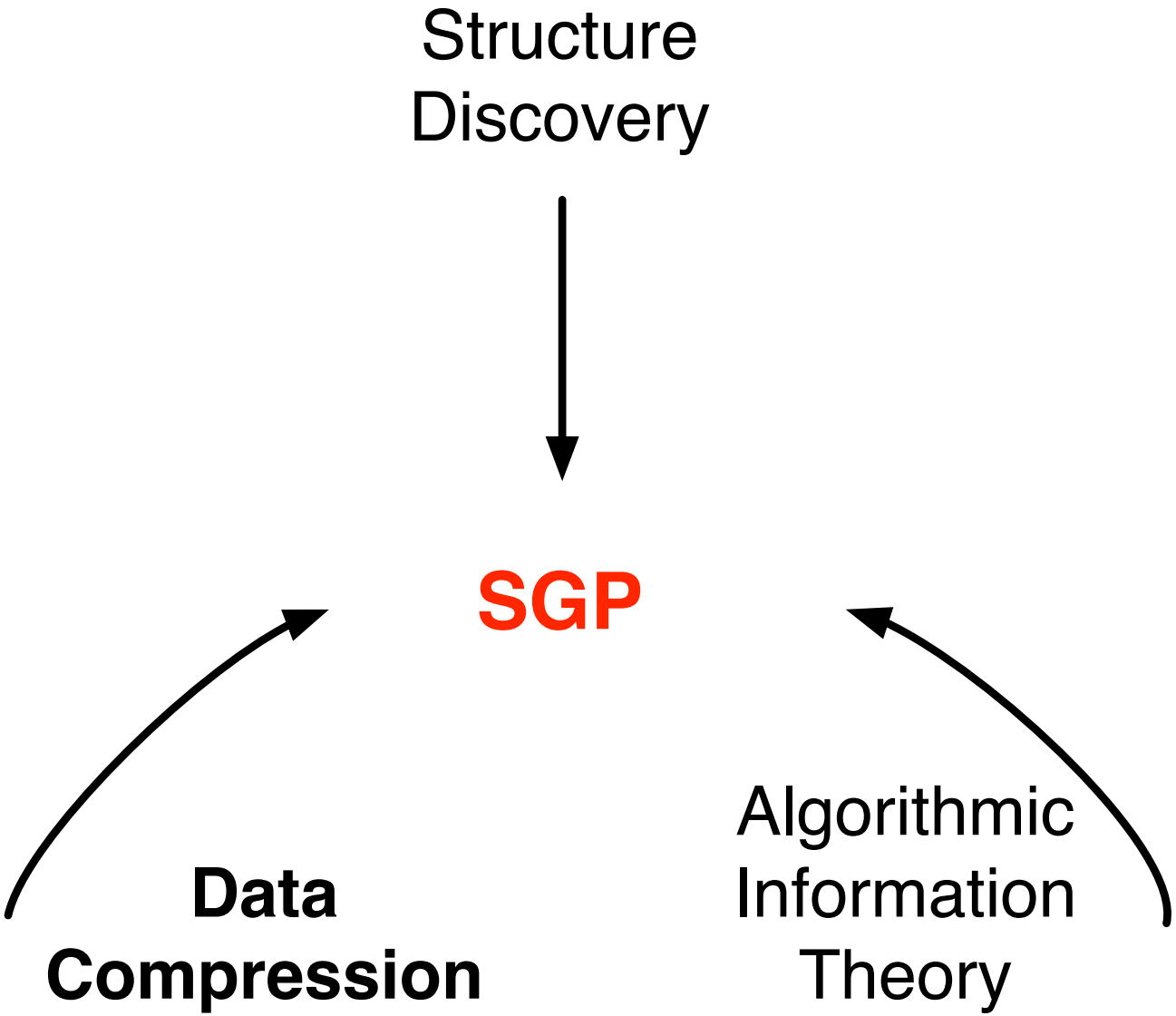
2. Attacking the Smallest Grammar Problem
   - What is a Word? Efficiency Issues
   - Choice of Occurrences
   - Choice of Set of Words

3. Applications: DNA Compression

A Generic Problem



A Generic Problem



Grammar-Based Codes [Kieffer & Yang 00]

$$s \quad \Longrightarrow G_s \Longrightarrow \quad R_s \Longrightarrow \quad B_s$$

Grammar-Based Codes [Kieffer & Yang 00]

$s \Longrightarrow G_s \Longrightarrow R_s \Longrightarrow B_s$

"how much wood would a woodchuck...

$$
\begin{array}{lcl}
S & \longrightarrow & \text{how much } N_2 \text{ w} N_3 \ldots \\
N_1 & \longrightarrow & \text{chuck} \\
N_2 & \longrightarrow & \text{wood} \\
N_3 & \longrightarrow & \text{ould} \\
N_4 & \longrightarrow & \text{a } N_2 N_1
\end{array}
$$

how much N_2 wN_3... | chuck | wood |...

10011...

Grammar-Based Codes [Kieffer & Yang 00]

$s \Longrightarrow G_s \Longrightarrow R_s \Longrightarrow B_s$

"how much wood would a woodchuck...

S	$\longrightarrow$	how much N_2 wN_3...
N_1	$\longrightarrow$	chuck
N_2	$\longrightarrow$	wood
N_3	$\longrightarrow$	ould
N_4	$\longrightarrow$	a $N_2 N_1$

how much N_2 wN_3... | chuck | wood |...

10011...

Combine macro schema with statistical schema

Grammar-Based Codes [Kieffer & Yang 00]

$$s \Longrightarrow G_s \Longrightarrow R_s \Longrightarrow B_s$$

"how much wood would a woodchuck...

S	$\longrightarrow$	how much N_2 wN_3...
N_1	$\longrightarrow$	chuck
N_2	$\longrightarrow$	wood
N_3	$\longrightarrow$	ould
N_4	$\longrightarrow$	a $N_2 N_1$

how much N_2 wN_3... | chuck | wood |...

10011...

Combine macro schema with statistical schema

Kieffer and Yang showed universality for such *Grammar-Based Codes*[3]

[3] Kieffer and Yang "Grammar-based codes: a new class of universal lossless source codes". 2000. IEEE TIT

Application: DNA Compression

- DNA difficult to compress better than the baseline of 2 bits per symbol
- $\geq$ 20 algorithms in the last 18 years
- Four Grammar-based specific DNA compressor:
  - Greedy Apostolico, Lonardi. "Compression of Biological Sequences by Greedy off-line Textual Substitution". 2000
  - GTAC Lanctot, Li, Yang. "Estimating DNA sequence entropy". 2000
  - DNASequitur Cherniavsky, Lander. "Grammar-based compression of DNA sequences". 2004
  - MDLcompress Evans, Kourtidis, et al. "MicroRNA Target Detection and Analysis for Genes Related to Breast Cancer Using MDLcompress" 2007

Grammar-based DNA compressor

bits per symbol

sequence	DNA Sequitur	GTAC[4]	Greedy	MDL Compress	AAC-2	DNA Light
chmpxx	2.12	3.1635	1.9022	-	1.8364	1.6415
chntxx	2.12	3.0684	1.9986	1.95	1.9333	1.5971
hehcmv	2.12	3.8455	2.0158	-	1.9647	1.8317
humdyst	2.16	4.3197	2.3747	1.95	1.9235	1.8905
humghcs	1.75	2.2845	1.5994	1.49	1.9377	0.9724
humhbb	2.05	3.4902	1.9698	1.92	1.9176	1.7416
humhdab	2.12	3.4585	1.9742	1.92	1.9422	1.6571
humprt	2.14	3.5302	1.9840	1.92	1.9283	1.7278
mpomtcg	2.12	3.7140	1.9867	-	1.9654	1.8646
mtpacga	-	3.4955	1.9155	-	1.8723	1.8442
vaccg	2.01	3.4782	1.9073	-	1.9040	1.7542

[4] our implementation

Special characteristics of DNA

- Complementary strand

Special characteristics of DNA

- Complementary strand
- Inexact repeats:
  - We used rigid patterns / partial words: motifs of fixed size that may contain a special don't care / joker symbol ($\bullet$)
  - "$\_\bullet$ *ould*" matches "$\_$*would*" and "$\_$*could*"
  - Exceptions are cheap to encode (no need of specifying position)

Straight-line Grammars with Don't Cares

$$
\begin{array}{lcl}
S & \longrightarrow & hN_1hN_2N_3a\_woN_1k\_chuck\_if\_a\_woN_1kN_3chuckN_2? \\
N_1 & \longrightarrow & o \bullet \bullet \bullet uc \\
N_2 & \longrightarrow & \_wood \\
N_3 & \longrightarrow & \_ \bullet ould\_ \\
E & \longrightarrow & w\_mwdchdchc
\end{array}
$$

Classes of rigid patterns

- repeated
- simple, maximal, irredundant[5] ($\approx$ largest-maximal repeats) motifs

[5] Parida,et al. "Pattern Discovery on character sets and real-valued data: linear bound on irredundant motifs and polynomial time algorithms" SODA 00

Classes of rigid patterns

- repeated
- simple, maximal, **irredundant**[5] ($\approx$ largest-maximal repeats) motifs
- but they are not *dense* enough, have mostly two occurrences which overlap

[5] Parida,et al. "Pattern Discovery on character sets and real-valued data: linear bound on irredundant motifs and polynomial time algorithms" SODA 00

Classes of rigid patterns

- repeated
- simple, maximal, **irredundant**[5] ($\approx$ largest-maximal repeats) motifs
- but they are not *dense* enough, have mostly two occurrences which overlap
- our heuristic: start from a (maximal) repeat r, use it as a seed to find its *occurrence-equivalent maximal motif*[6]: $extension(r)$

[5] Parida,et al. "Pattern Discovery on character sets and real-valued data: linear bound on irredundant motifs and polynomial time algorithms" SODA 00

[6] Ukkonen, "Maximal and minimal representations of gapped and non-gapped motifs of a string" Theoretical CS 2009

Iterative Motif Replacement

- IMR: an algorithm that computes a straight-line grammar with don't cares
- IRR-like:
  1. select in each iteration a maximal repeat r that reduces the most $\hat{H}(G)$ (empirical entropy)
     $$\hat{H}(G) = -\sum_{x \in \Sigma \cup \mathcal{N} \cup \{|\}} occ_G(x) * \log \frac{occ_G(x)}{|G|}$$
  2. Use it as a seed to compute $m = extension(r)$
  3. Recover the submotif of m that reduces the most $\hat{H}(G)$

More details

Iterative Motif Replacement: Results

bits per symbol

sequence	DNA Sequitur	Greedy	MDL Compress	**IMR^c**	AAC-2	DNA Light
chmpxx	2.12	1.9022	-	1.6793	1.8364	1.6415
chntxx	2.12	1.9986	1.95	1.6196	1.9333	1.5971
hehcmv	2.12	2.0158	-	1.8542	1.9647	1.8317
humdyst	2.16	2.3747	1.95	1.9331	1.9235	1.8905
humghcs	1.75	1.5994	1.49	1.1820	1.9377	0.9724
humhbb	2.05	1.9698	1.92	1.8313	1.9176	1.7416
humhdab	2.12	1.9742	1.92	1.8814	1.9422	1.6571
humpr	2.14	1.9840	1.92	1.8839	1.9283	1.7278
mpomtcg	2.12	1.9867	-	1.9157	1.9654	1.8646
mtpacga	-	1.9155	-	1.8571	1.8723	1.8442
vaccg	2.01	1.9073	-	1.7743	1.9040	1.7542

IMR^c encodes explicitly with the structure.

The grammars is encoded with a standard adaptive arithmetic encoder.

Conclusions

Summary: The general SGP

- We studied the Smallest Grammar Problem from the motivation of finding meaningful hierarchical structure in DNA sequencs
- Approach: to split SGP into two:
  1. Choice of Words
     - ★ Classes of maximality of repeats; algorithms and bounds
     - ★ Efficiency: IRR from $\mathcal{O}(n^3)$ to $\mathcal{O}(n^2)$
     - ★ Efficiency: Inplace update of an enhanced suffix array
  2. Choice of Occurrences
     - ★ MGP Problem and its solution
     - ★ Lattice as a search space
     - ★ Algorithms that find smaller grammars ($\approx 10\%$) than state of the art

Summary: Applications

- Data Compression: compress with structure. First competitive grammar-based DNA compressor by extending the notion of straight-line grammar to rigid motifs
- AIT: consistent results using IRRMGP* in a Normalised Compression Distance framework
- Structure Discovery: analysis of number of smallest grammar and their similarity

Perspectives: Beyond the SGP

- Smallest grammar $\neq$ most compressible
- SGP does not care about the size of the alphabet
- Experiments: huge number of smallest grammar seems to come from the presence of small words
- Back to Structure Discovery:
  - "better" grammars with rigid motifs
  - go beyond rigid motifs

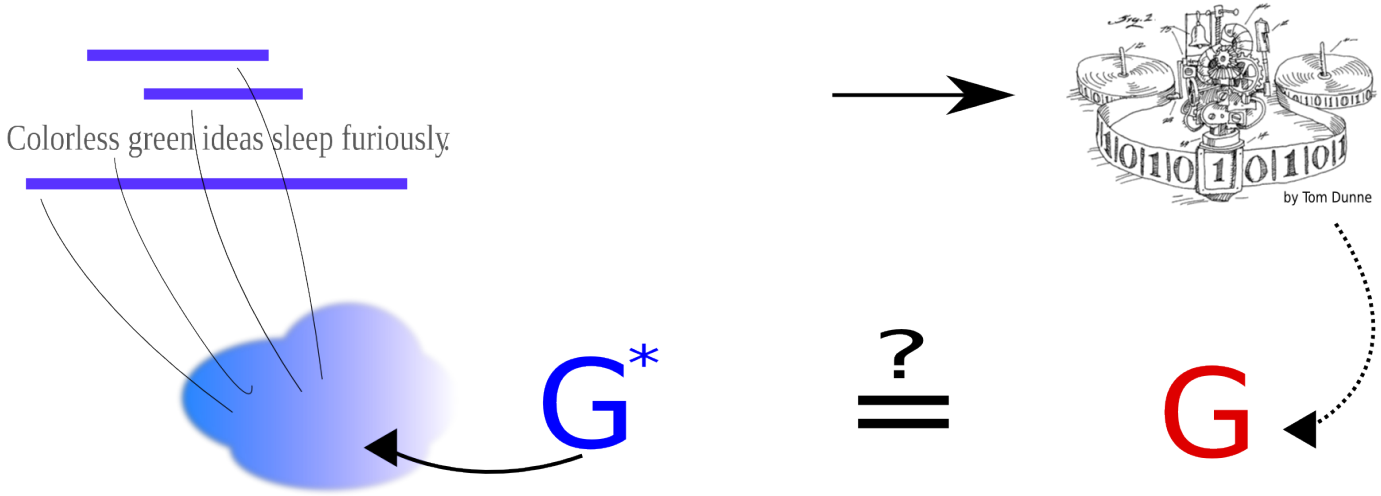
Perspectives: Beyond the SGP

- The SGP overfits by design. "To learn you have to forget"
- Generalise the final grammar. SLG with don't cares is a first step in this direction.

- Links to Grammatical Inference

Learn a General Grammar

- Class of CF Languages are not learnable [Gold 67]
- Class of CF Languages can be learnt from positive examples + parse trees [Sakakibara, 92]
- Several algorithms that work well in practice based on substitutability, mutual information, frequency, etc.



Acknowledgments

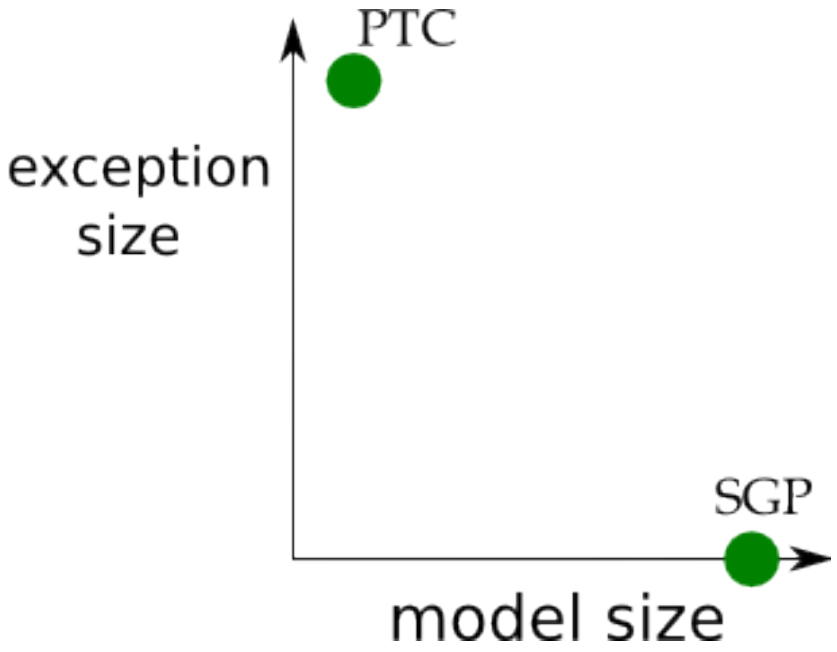
- €: CORDIS contract; MINCyT / INRIA / CNRS collaboration
- François Coste, Gabriel Infante-López
- Symbiose, pln
- Pierre Peterlongo (INRIA Rennes), Rafael Carrascosa (U Córdoba)
- Matthieu Perrin (ENS Cachan Bretagne), Tania Roblot (U Auckland)
- IST INRIA Staff (Pascale, Anne, Agnès)

The End

$$
\begin{array}{lcl}
S & \longrightarrow & \text{thDkAforBr\_attenC.\_DoAhave\_Dy\_quesCs?} \\
A & \longrightarrow & \text{B\_} \\
B & \longrightarrow & \text{\_you} \\
C & \longrightarrow & \text{tion} \\
D & \longrightarrow & \text{an}
\end{array}
$$

Appendix

- Parse Tree Compression and SGP are two extremes
- PTC: model is (very) general. Grammar is given to both encoder and decoder, only derivation is send.



- Find the MDL-inspired golden mean

Heuristic for Selecting a Good Motif

1. Select exact repeat that minimises

$$H(G) = -\sum_{x \in \Sigma \cup \mathcal{N} \cup \{|\}} occ_G(x) * \log \frac{occ_G(x)}{|G|}$$

`..._od_would_a_wo...chuck_could_c_...`

ould_

Back

Heuristic for Selecting a Good Motif

1. Select exact repeat that minimises
$$H(G) = -\sum_{x \in \Sigma \cup \mathcal{N} \cup \{|\}} occ_G(x) * \log \frac{occ_G(x)}{|G|}$$
2. extend it to the left minimising $H(G)$

$\longleftarrow$ $\longleftarrow$

`...`_`od`_`would`_`a`_`wo...chuck`_`could`_`c`_`...`

ould_

Back

Heuristic for Selecting a Good Motif

1. Select exact repeat that minimises
$$H(G) = -\sum_{x \in \Sigma \cup \mathcal{N} \cup \{|\}} occ_G(x) * \log \frac{occ_G(x)}{|G|}$$
2. extend it to the left minimising $H(G)$

$\longleftarrow$ $\longleftarrow$

..._od_would_a_wo...chuck_could_c_...

•ould_

Back

Heuristic for Selecting a Good Motif

1. Select exact repeat that minimises
$$H(G) = -\sum_{x \in \Sigma \cup \mathcal{N} \cup \{|\}} occ_G(x) * \log \frac{occ_G(x)}{|G|}$$
2. extend it to the left minimising $H(G)$

← ←

..._od_would_a_wo...chuck_could_c_...

•ould

Back

Heuristic for Selecting a Good Motif

1. Select exact repeat that minimises
$$H(G) = -\sum_{x \in \Sigma \cup \mathcal{N} \cup \{|\}} occ_G(x) * \log \frac{occ_G(x)}{|G|}$$
2. extend it to the left minimising $H(G)$

$\longleftarrow$ $\longleftarrow$

`...`_`od`_`would`_`a`_`wo...chuck`_`could`_`c`_`...`

...●●_●ould_

Back

Heuristic for Selecting a Good Motif

1. Select exact repeat that minimises
$$H(G) = -\sum_{x \in \Sigma \cup \mathcal{N} \cup \{|\}} occ_G(x) * \log \frac{occ_G(x)}{|G|}$$
2. extend it to the left minimising $H(G)$
3. extend it to the right minimising $H(G)$

$\longrightarrow$ $\longrightarrow$

. . . _od_would_a_wo. . . chuck_could_c_. . .

. . . • • _•ould_• • . . .

Back

Heuristic for Selecting a Good Motif

1. Select exact repeat that minimises
$$H(G) = -\sum_{x \in \Sigma \cup \mathcal{N} \cup \{|\}} occ_G(x) * \log \frac{occ_G(x)}{|G|}$$
2. extend it to the left minimising $H(G)$
3. extend it to the right minimising $H(G)$

`...`_`od`_`would`_`a`_`wo...chuck`_`could`_`c`_`...`

•ould

Back

Enhanced Suffix Array [Abouelhoda, Kurtz, et al 2004]

ABRACADABRA → ABRACADABRA$

Enhanced Suffix Array [Abouelhoda, Kurtz, et al 2004]

sarr + *lcp* + *isa* = *ESA*

i	isa	lcp	sarr	suffix
0	3	0	11	$
1	7	0	10	A$
2	11	1	7	ABRA$
3	4	4	0	ABRACADABRA$
4	8	1	3	ACADABRA$
5	5	1	5	ADABRA$
6	9	0	8	BRA$
7	2	3	1	BRACADABRA$
8	6	0	4	CADABRA$
9	10	0	6	DABRA$
10	1	0	9	RA$
11	0	2	2	RACADABRA$

Our update algorithm

i	isa	lcp	sa	suffix
0	1	0	25	
1	14	0	0	ACGCATCTCCATCGCGCATATCATC
2	18	1	17	ATATCATC
3	11	2	22	ATC
4	6	3	19	ATCATC
5	25	3	10	ATCGCGCATATCATC
6	16	3	4	ATCTCCATCGCGCATATCATC
7	23	0	24	C
8	12	1	16	CATATCATC
9	10	3	21	CATC
10	5	4	9	CATCGCGCATATCATC
11	24	4	3	CATCTCCATCGCGCATATCATC
12	15	1	8	CCATCGCGCATATCATC
13	19	1	14	CGCATATCATC
14	13	5	1	CGCATCTCCATCGCGCATATCATC
15	17	3	12	CGCGCATATCATC
16	8	1	6	CTCCATCGCGCATATCATC
17	2	0	15	GCATATCATC
18	20	4	2	GCATCTCCATCGCGCATATCATC
19	4	2	13	GCGCATATCATC
20	22	0	18	TATCATC
21	9	1	23	TC
22	3	2	20	TCATC
23	21	2	7	TCCATCGCGCATATCATC
24	7	2	11	TCGCGCATATCATC
25	0	2	5	TCTCCATCGCGCATATCATC

- Enhanced Suffix array for *ACGCATCTCCATCGCGCATATCATC*

Our update algorithm

i	isa	lcp	sa	suffix
0	1	0	25	
1	14	0	0	ACGCATCTCCATCGCGCATATCATC
2	18	1	17	ATATCATC
3	11	2	22	ATC
4	6	3	19	ATCATC
5	25	3	10	ATCGCGCATATCATC
6	16	3	4	ATCTCCATCGCGCATATCATC
7	23	0	24	C
8	12	1	16	CATATCATC
9	10	3	21	CATC
10	5	4	9	CATCGCGCATATCATC
11	24	4	3	CATCTCCATCGCGCATATCATC
12	15	1	8	CCATCGCGCATATCATC
13	19	1	14	CGCATATCATC
14	13	5	1	CGCATCTCCATCGCGCATATCATC
15	17	3	12	CGCGCATATCATC
16	8	1	6	CTCCATCGCGCATATCATC
17	2	0	15	GCATATCATC
18	20	4	2	GCATCTCCATCGCGCATATCATC
19	4	2	13	GCGCATATCATC
20	22	0	18	TATCATC
21	9	1	23	TC
22	3	2	20	TCATC
23	21	2	7	TCCATCGCGCATATCATC
24	7	2	11	TCGCGCATATCATC
25	0	2	5	TCTCCATCGCGCATATCATC

- Enhanced Suffix array for $ACGCATCTCCATCGCGCATATCATC$
- Replace each occurrence of $w = CAT$ by M.

Our update algorithm

i	isa	lcp	sa	suffix
0	1	0	25	
1	14	0	0	ACGCATCTCCATCGCGCATATCATC
2	18	1	17	ATATCATC
3	11	2	22	ATC
4	6	3	19	ATCATC
5	25	3	10	ATCGCGCATATCATC
6	16	3	4	ATCTCCATCGCGCATATCATC
7	23	0	24	C
8	12	1	16	CATATCATC
9	10	3	21	CATC
10	5	4	9	CATCGCGCATATCATC
11	24	4	3	CATCTCCATCGCGCATATCATC
12	15	1	8	CCATCGCGCATATCATC
13	19	1	14	CGCATATCATC
14	13	5	1	CGCATCTCCATCGCGCATATCATC
15	17	3	12	CGCGCATATCATC
16	8	1	6	CTCCATCGCGCATATCATC
17	2	0	15	GCATATCATC
18	20	4	2	GCATCTCCATCGCGCATATCATC
19	4	2	13	GCGCATATCATC
20	22	0	18	TATCATC
21	9	1	23	TC
22	3	2	20	TCATC
23	21	2	7	TCCATCGCGCATATCATC
24	7	2	11	TCGCGCATATCATC
25	0	2	5	TCTCCATCGCGCATATCATC

Steps of the algorithm

1. Delete positions
2. Move some lines
3. Update LCP

Efficiency

$$\frac{\text{recreating from scratch}}{\text{our update}} = \begin{array}{c} \uparrow \\ 1.0 \\ \downarrow \end{array} \begin{array}{l} \textit{better} \\ \\ \textit{worse} \end{array}$$

sequence	size	Φ lcp	random		max length		max comp.	
			K&S	L&S	K&S	L&S	K&S	L&S
bible.txt	4MB	13,0	66,8	22,9	64,4	22,5	15,4	3,7
E.coli	4.6MB	23,0	69,1	27,4	53,5	24,0	9,5	2,1
world192	2.5MB	17,4	65,0	21,8	60,7	21,1	16,3	4,5

Back

Problems of IRR-like algorithms

Example

$xaxbxcx|_1xbxcxax|_2xcxaxbx|_3xaxcxbx|_4xbxaxcx|_5xcxbxax|_6xax|_7xbx|_8xcx$

Problems of IRR-like algorithms

Example

$xaxbxcx|_1 xbxcxax|_2 xcxaxbx|_3 xaxcxbx|_4 xbxaxcx|_5 xcxbxax|_6 xax|_7 xbx|_8 xcx$

A smallest grammar is:

$$
\begin{aligned}
S &\rightarrow Ab\,C|_1 B\,c\,A|_2 C\,a\,B|_3 A\,c\,B|_4 B\,a\,C|_5 C\,b\,A|_6 A|_7 B|_8 C \\
A &\rightarrow xax \\
B &\rightarrow xbx \\
C &\rightarrow xcx
\end{aligned}
$$

Problems of IRR-like algorithms

Example

$xaxbxcx|_1xbxcxax|_2xcxaxbx|_3xaxcxbx|_4xbxaxcx|_5xcxbxax|_6xax|_7xbx|_8xcx$

But what IRR can do is like:

$$
\begin{array}{lcl}
S & \longrightarrow & Abxcx|_1xbxcA|_2xcAbx|_3Acxbx|_4xbAcx|_5xcxbA|_6A|_7xbx|_8xcx \\
A & \longrightarrow & xax \\
 & \Downarrow & \\
S & \longrightarrow & Abxcx|_1BcA|_2xcAbx|_3AcB|_4xbAcx|_5xcxbA|_6A|_7B|_8xcx \\
A & \longrightarrow & xax \\
B & \longrightarrow & xbx \\
 & \Downarrow & \\
S & \longrightarrow & AbC|_1BcA|_2xcAbx|_3AcB|_4xbAcx|_5CbA|_6A|_7B|_8C \\
A & \longrightarrow & xax \\
B & \longrightarrow & xbx \\
C & \longrightarrow & xcx
\end{array}
$$

Back

Non-Uniqueness of SG

Lemma

There can be an exponential number of global minima in the lattice.

Lemma

Given a fixed node C, there can be an exponential number of minimal grammars with these constituents.

Stability of Small Grammars

Measure

UF_1: harmonic mean between precision and recall of brackets given by the parse tree / grammar.

Stability 1 (of 3)

Given a node C (chosen by ZZ), pick up two random minimal grammar parsing with these constituents.

Stability 1 (of 3)

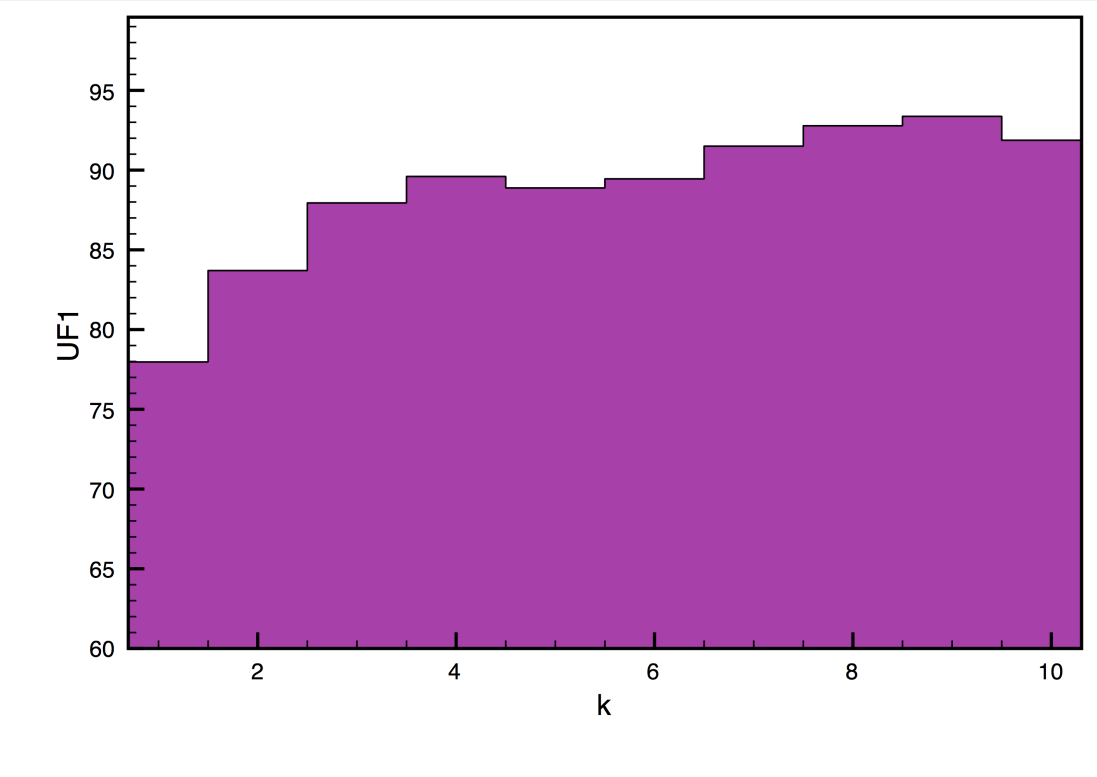
Given a node C (chosen by ZZ), pick up two random minimal grammar parsing with these constituents.
$UF_1 = 77.81\%$ (`alice29.txt`, with 1000 samples)

Stability 2 (of 3)

Consider only brackets of size $> k$

Stability 2 (of 3)

Consider only brackets of size $> k$

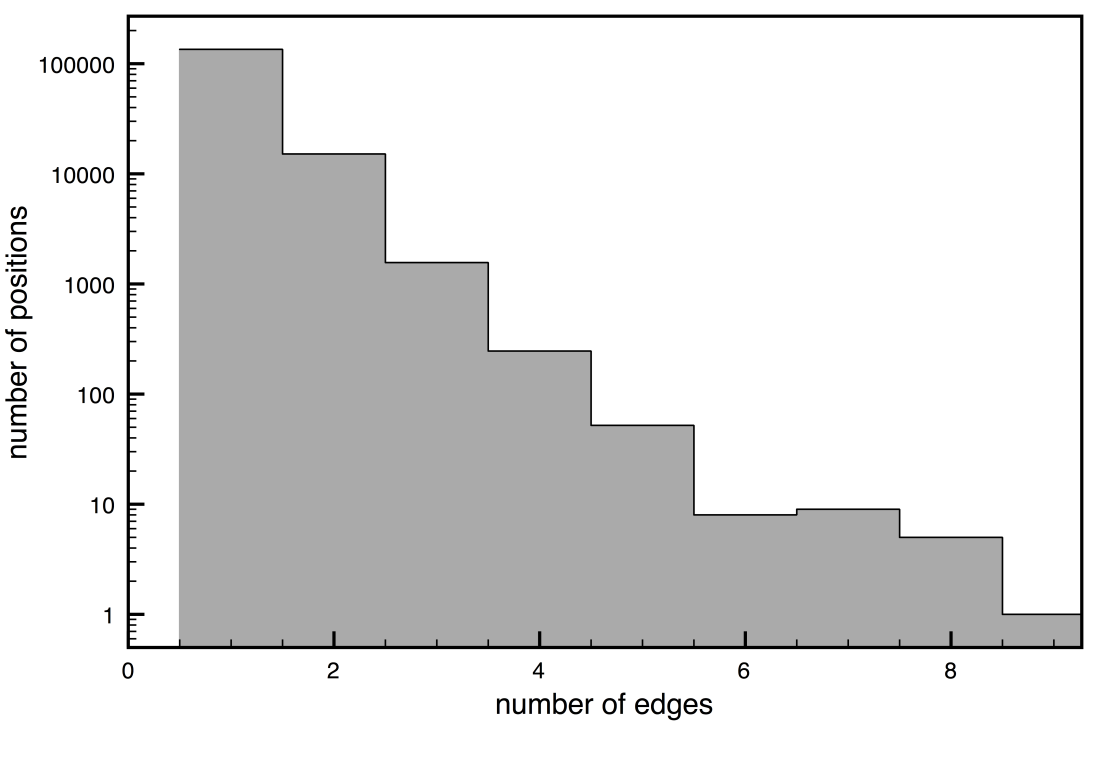


Stability 3 (of 3)

Consider number of possible parses given a position

Stability 3 (of 3)

Consider number of possible parses given a position



Stability 3 (of 3)

Consider number of possible parses given a position

Ex: A really *unstable* zone corresponds to:

```
              'Fury said to a
             mouse, That he
           met in the
         house,
      "Let us
        both go to
          law:  I will
            prosecute
              YOU.  --Come,
                 I'll take no
                  denial; We
               must have a
           trial:  For
        really this
     morning I've
    nothing
   to do."
     Said the
       mouse to the
         cur, "Such
           a trial,
             dear Sir,
```

```
                With
            no jury
         or judge,
       would be
     wasting
    our
     breath."
      "I'll be
        judge, I'll
          be jury,"
                Said
            cunning
              old Fury:
             "I'll
              try the
                 whole
                  cause,
                     and
               condemn
              you
             to
              death."'
```

Results on Penn Treebank (POS)

strategy	number of brackets	UP	UR	UNCP	UNCR
mc	934338	22.5	21.5	43.7	45.2
ml	990109	9.2	9.3	23.2	30.1
mo	965277	21.4	21.1	42.1	43.9
key	960027	12.6	12.3	29.2	33.7
pc	960603	13.0	12.7	29.7	34.2
sequitur	961660	14.0	13.0	31.4	35.4

Results of bracketing the POS tags of the Penn Treebank IRR algorithm, compared to the gold standard (977205 brackets)

Results on Penn Treebank

strategy	number of brackets	UP	UR	UNCP	UNCR
rbranch		46.7	42.8	64.9	74.3
mc	31652	38.7	30.2	57.8	68.7
ml	33710	27.1	22.6	43.4	57.6
mo	33084	38.0	31.0	56.9	67.6
key	32738	24.4	19.7	41.0	56.3
pc	32792	23.8	19.3	40.8	55.6
sequitur	33112	29.5	24.1	47.1	61.0

Results of bracketing the POS tags of the Penn Treebank 10 (up to 10 words, without punctuation) IRR algorithm, compared to the gold standard (40535 brackets)

Structural Information Theory with Grammars

Eine generative Grammatik ist ein Quadrupel

$G = (V_T, V_H, R, S)$

mit der Menge der terminalen Symbole (Alphabet) V_T, der Menge der Hilfssymbole (Variablen) V_H, der Menge der Ersetzungsregeln R und dem Startsymbol $S \in V_H$. Wir wollen hier den Typ des Ersetzungsregeln einschränken, indem wir nur kontextfreie Regeln zulassen, d.h. nur Regeln der Art

$$\sigma \rightarrow q \text{ mit } \sigma \in V_H,\ q \in (V_T \cup V_H) \quad .$$

Mit dieser Einschränkung kann man die Kompliziertheit einer Regel durch die Wortlänge der rechten Seite definieren:

$$K(\sigma \rightarrow q) \underset{Df.}{=} |q| \quad , \qquad (1)$$

wobei $|q|$ die Wortlänge von q bedeutet. Die Kompliziertheit eines Wortes p über dem Alphabet V_T definieren wir dann als Summe der Kompliziertheit derjenigen Regeln, die zur Ableitung des Wortes p benutzt werden, unabhängig davon, wieoft:

$$K_G(p) \underset{Df.}{=} \sum_{\sigma \in V_H} K(\sigma \rightarrow q) \qquad (2)$$

Dabei soll V_H genau die Variablen enthalten, die in der Ableitung von p vorkommen, und es soll zu jeder Variablen genau eine Regel existieren.

Scheidereiter, "Zur Beschreibung strukturierter Objeckte mit kontextfreien Grammatiken" 1973

Structural Information Theory with Grammars

Es ist leicht einzusehen, daß es mehrere Grammatiken gibt, die das Wort p erzeugen. Das Optimalitätsproblem besteht jetzt darin, eine solche zu finden, bei der die Kompliziertheit von p minimal ist. Da es unter relativ einfachen Bedingungen nur endlich viele solcher Grammatiken gibt, könnte man durch Probieren eine optimale finden. Wir wollen einen anderen Weg gehen.

"Under relatively simple condition, there exists only a finite number of such grammars, one could find an optimal one by exhaustive search"

Scheidereiter, "Zur Beschreibung strukturierter Objeckte mit kontextfreien Grammatiken" 1973