A model simulation for trip planning recommendation system in Tourism

Uyanga Sukhbaatar

Communauté
UNIVERSITÉ Grenoble Alpes

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

préparée dans le cadre d'une cotutelle entre *Communauté Université de Grenoble Alpes* et *L'Universite de Science et Technologie de la Mongolie*

Spécialité :**Mathematique et Informatique**
Arrêté ministériel : le 6 janvier 2005 -7 août 2006

Présentée par

Uyanga SUKHBAATAR

Thèse dirigée par **Ahmed LBATH**
codirigée par **Altangerel AYUSH**

préparée au sein des **Laboratoires d'Informatique de Grenoble** et de L'**Ecole Doctorale Mathematiques, Sciences et de Technologies l'Information, Informatique (MSTII)**

Un modele de simulation pour une infrastructure logistique dediee a la fourniture de services mobiles sensibles au contexte: application au Tourism

Thèse soutenue publiquement le« **07 Octobre, 2016** »,
devant le jury composé de :

Prof. Sebti Foufou
Universite de Bourgogne, President
Prof. Enkhbat RENTSEN
Institut of Mathematics, National University of Mongolia, Examinateur
Prof. Mend-Amar Majig
National University of Mongolia, Examinateur
Prof. Altangerel AYUSH
Mongolian University of Science and Technology,Co-directeur de these
Prof. Ahmed LBATH
Joseph Fourier University of Grenoble, Directeur de these
Prof. Pradorn Sureephong
Univercite de Chiang Mai, Rapporteur
Prof. Santichai Wicha
Univercite de Chiang Mai, Examinateur

Abstract

When tourists visit a city for one or multiple days, they are unlikely to visit every tourist attraction places. But they have to deal with the dilemma of which points of interest (POI) would be worth to visit. These choices are normally based on information gathered by tourists via the Internet, magazines, printed tourist guides, etc. After deciding of which sights to visit, tourists have to consider on which route to take with respect to the visiting time required for each place, the POI's visiting days/hours and the timetable for sites, entrance fees and other constraints. All these kinds of requirements and questions are represented as a Tourist Trip Planning Problem in the field of Operations Research.

This research work aims to investigate the several variants of trip planning problems and develop efficient technique to solve that optimization problem. In order to model this kind of problem the Orienteering Problem became the promising starting point which has originated from the group sport game so called “Orienteering”. The main objective of the OP is to collect the score which is associated to the each point. According to different type of optimization problems, this basic model is extended to several variants in order to enable additional tourist functionalities. In this thesis, we will discuss about one of the latest and hardest versions of the OP which we called the Time Dependent Multi Constraint Team Orienteering Problem with Time Windows. The simple MCTOPTW takes into account money budget limitation as multiple constraints in addition to time window and associated satisfaction score while the TDMCTOPTW considers the integration of urban public transportation network into the MCTOPTW. Based on the algorithm that is successfully applied to the certain version of problem, we proposed the Iterated Local Search Algorithm to tackle the Time Dependent Multi Constraint Team Orienteering Problem with Time Windows.

Finally we applied the model and algorithm to mobile tourist tour recommendation system that enables to plan a tour for Ulaanbaatar city, so called UBTourPlanner. The real life test set of Ulaanbaatar city is experimented and implemented.

Résumé

Quand les touristes visitent une ville pour un ou plusieurs jours, ils sont peu susceptibles de visiter tous les lieux d'attraction touristique. Mais ils doivent faire face au dilemme de laquelle les points d'intérêt (POI) serait intéressant à visiter. Ces choix sont normalement basés sur des informations recueillies par les touristes via Internet, magazines, guides touristiques imprimés, etc. Après avoir décidé qui sites à visiter, les touristes doivent prendre en compte sur la route à prendre par rapport au temps de visite requis pour chaque place, visite jours / heures du POI et le calendrier pour les sites, les frais d'entrée et d'autres contraintes. Tous ces types d'exigences et les questions sont représentés comme un voyage touristique Planification problème dans le domaine des opérations de recherche.

Ce travail de recherche vise à étudier les différentes variantes de problèmes de planification de voyage et de développer la technique efficace pour résoudre ce problème d'optimisation. Afin de modéliser ce genre de problème le problème Orienteering est devenu le point de départ prometteur qui a son origine dans le jeu de sport de groupe que l'on appelle "Orienteering". L'objectif principal de l'OP est de recueillir le score qui est associé à l'chaque point. Selon le type de problèmes d'optimisation différents, ce modèle de base est étendue à plusieurs variantes afin de permettre aux fonctionnalités touristiques supplémentaires. Dans cette thèse, nous allons discuter de l'une des versions les plus récentes et les plus difficiles de l'OP que nous avons appelé l'heure à charge multi Constraint équipe Orienteering un problème de temps Windows. Le MCTOPTW simple, prend en compte la limitation du budget de l'argent comme de multiples contraintes, en plus de la fenêtre de temps et le score de satisfaction associée tandis que le TDMCTOPTW considère l'intégration du réseau de transport public urbain dans le MCTOPTW. Sur la base de l'algorithme qui est appliqué avec succès à la certaine version du problème, nous avons proposé la Iterated Local Search Algorithme pour resoudre le problem de TDMCTOPTW.

Enfin, nous avons appliqué le modèle et l'algorithme de système de recommandation de circuit touristique mobile qui permet de planifier une visite de la ville d'Oulan-

Bator, ainsi appelé UBTourPlanner. L'ensemble de test de vie réelle de la ville d'Ulaanbaatar est expérimenté et mis en œuvre.

Acknowledgements

This thesis is a result of exciting four years of research. I want to express my gratitude for the people who have helped me and gave me continuous support until the end of this work. First of all, I wish to thank my supervisor, Professor Ahmed LBATH, for giving me this opportunity to pursue a PhD, for supervising me during these beautiful four years in Grenoble and in Ulaanbaatar by his broad knowledge and experience in this field. It was my honor to work with him and was a real pleasure. I always appreciate his generosity for giving me an opportunity to work with the very kind team MRIM of LIG, University of Grenoble (Modélisation et Recherche d'Information Multimédia of Laboratoire Informatique de Grenoble). I am also thankful to all Professors and colleagues from the MRIM team.

I am so grateful to my co-supervisor, Prof Altangerel Ayush and examiner Dr Mend-Amar Majig. This research work would have been much poorer without their inestimable advises and efficacious guidance. They influenced and supported this work in many ways. I also wish to thank the European Commission. This research work could not be done without great help and financial support of the European Union, E-tourism Program. I am really blessed that I had a very rare opportunity to study and live in a European country. I could not imagine my PhD life in Grenoble without my friends Pathathai Na Lumpoon, Mu lei, Teerawat Kamnerdsiri, Isaac Caicedo Castro and all my friends of the LIG. Also, I appreciate great help of my collegues doctorante Ariunbayr S. and Dr. Naranbaatar D.

I wish to thank to my family, my mother, my father, brothers and sisters. Especially thank my grandmother Khumbaa Bat and my wonderful husband Ulziisaikhan Chuluunbaatar, for his care, unlimited support and effort by keep motivating me.

Also, I express my gratitude to the members of my examination committee, Prof. Nopasit Chakpitak, Prof. Pradorn Sureephong and Prof Enkhbat Rentsen, generously gave useful remarks and carefully reviewed this work. This final manuscript is definitely improved from the previous version based on their great advises and remarks.

Contents

List of Figures

List of Tables

List of Symbols

A_{id}	The arrival time at point I, in tour d
C_i	Closing hour of the point i
C_{ij}	Distance from city i to city j
e_{izd}	Variable equal to 1 if location i is category z in tour d, 0 otherwise
E_z	Maximun number of visits of the particular category
f_{id}	Spent money (entrance fee) to visit point i in tour d
F_{max}	Total money budget for each tour
F(S)	Value of the objective function in Local Search
m	Number of tours
$MaxArrivalDelay_i$	How much the currently scheduled arrival time at a particular attraction point can be delayed without making any visit unfeasible
maxDelay	The maximum time a departure from an attraction point i can be delayed
$maxshift_{id}$	Maximum time the service completion of a given visit of point i can be delayed in tour d
$MaxTrans_i$	The maximum time the leave time of a visit can be delayed without changing the bus between attraction points i and i+1
$MinTrans_i$	The maximum time the leave time can be decreased without changing the bus between attraction points i and i+1.
n	Number of point

O_i	Opening hour of point i
$Ratio_{ijkd}$	Evaluation of each visit that will be selected for insertion
S’	Neighbor of the current solution S
S_i	Satisfaction score of point i
$Shift_{ijkd}$	The total time consumption to insert an extra visit j between visits i and k in tour d
t_i	Time duration to visit point i
t_{ij}	Travel time between point i to point j
T_{max}	Total time budget of each tour
$TransPeriod_i$	The period of the bus service travelling between attraction points i and i+1.
u_i	An artificial variable
V_{id}	Start of the visit i in tour d
V_{idt}	Start of the visit at point i in tour d, started at the time t
w_i	Weight of itemsin Knapsack Problem
$wait_{id}$	Waiting time at point i in tour d
X_{ij}	Variable equal to 1 if visit i is followed by visit j, otherwise equal to 0
X_{ijd}	Variable equal to1 if a visit point i is followed by a visit point j in a tour d, 0 Otherwise
X_{ijdt}	Variable equal to 1 if a visit point i is followed by a visit point j in tour d at the time period t, 0 otherwise
Y_{id}	Variable equal to1 if point i is visited in tour d, 0 otherwise
z_n	Number of items in Knapsack Problem

List of Abbreviations

ACO	**A**nt **C**olony **O**ptimization
AGA	**A**dapted **G**enetic **A**lgorithms
AHP	**A**nalytical **H**ierarchy **P**rocess
AMP	**A**daptive **M**emory **P**rocedure
BC	**B**utt and **C**avalier
ELS	**E**volutionary **L**ocal **S**earch
FPR	**F**ast variant of the **P**ath **R**elinking
FVN	**F**ast **V**ariable **N**eighborhood **S**earch
GA	**G**enetic **A**lgorithm
GLS	**G**uided **L**ocal **S**earch
GRASP	**G**reedy **R**andomized **A**daptive **S**earch **P**rocedure
ILS	**I**terated **L**ocal **S**earch
MCTOPTW	**M**ulti **C**onstraint **T**eam **O**rienteering **P**roblem with **T**ime **W**indows
OP	**O**rienteering **P**roblem
OPTW	**O**rienteering **P**roblem with **T**ime **W**indows
POI	**P**oint **O**f **I**nterest
SPR	**S**low variant of the **P**ath **R**elinking
STSP	**S**elective **T**ravelling **S**alesman **P**roblem

SVN	**S**low **V**ariable **N**eighborhood **S**earch
SVNS	**S**kewed **V**ariable **N**eighborhood **S**earch
SVRPTW	**S**elective **V**ehicle **R**outing **P**roblem with **T**ime **W**indows
TDTOPTW	**T**ime **D**ependent **T**eam **O**rienteering **P**roblem with **T**ime **W**indows
TDMCTOPTW	**T**ime **D**ependent **M**ulti **C**onstraint **T**eam **O**rienteering **P**roblem with **T**ime **W**indows
TOP	**T**eam **O**rienteering **P**roblem
TOPTW	**T**eam **O**rienteering **P**roblem with **T**ime **W**indows
TSP	**T**raveling **S**alesman **P**roblem
TTPP	**T**ourist **T**our **P**lanning **P**roblem
UBTP	**U**laanbaatar **T**our **P**lanner
VNS	**V**ariable **N**eighborhood **S**earch

Chapter 1

Introduction

Contents

Abstract. In this opening Chapter of the thesis we discuss an overview of the trip planning recommendation system regarding to its functionalities and related problem definition. In this research work, we introduce one of the classes of tour planning problem so called Orienteering Problem which is used as the starting point to model the trip planning problem. In this chapter, we also present the overall view of the thesis objectives, motivations and contributions which are presented in detail on the following chapters.

1.1 Mobile Trip Planning Recommendation system

In our everyday life we face the decision making problem in any field. But making a decision without any suggestions or recommendations is always hard task to solve. Especially if we are in unknown place in the middle of unknown people it is hard to say where to go, where to eat and where to stay. These kinds of problem always occur to tourists since he/she has never been in that country.

Based on that need, tourism related recommendation systems are offered by many resources to support tourists managing their long, medium and short tours. Basically, tourist recommendation systems can select and filter the relevant results to the user from the large database of tourist services starting by transportations, accommodations, attractions, Point of Interests and even fixed tour package. From the 1990's number of researchers presented a classification of different approaches and recommendations techniques based on their target applications, the way they formulate recommendations and the algorithms they implement in their paper such as collaborative filtering, content-based filtering, knowledge-based filtering and hybrid system. Recently, authors in (Gavalas, D. et al., 2014) presented different types of recommendation systems including:

Collaborative filtering: This type of the recommendation system was mostly used in social media and e-commerce. Recommend similar items which were chosen by other users with similar interest and preferences to user.

Content-based filtering: These recommendation systems based on content items that the user has selected before.

Knowledge-based filtering: This type of recommendation system filters by reasoning about what match the user's interest and requirements. In that type, the knowledge about user created by asking to provide user's preferences and choices.

Hybrid system: Hybrid system can combine any systems of the previously mentioned techniques.

Nowadays, there are plenty of tourist mobile recommendation systems that exist on the market and some of them operate in major tourism portals. We can categorize them by main features and their recommending items. In the fast developing computing era, we are using number of recommending and supporting applications. They are becoming more and more intelligent and effective thanks to developers' wide knowledge and great experiments. Below, we present most common services which are offered by tourism related recommendation systems that we use in our everyday life.

- Single tour planning recommendations
- Multiple tour planning recommendations
- Points of Interest recommendation
- Services recommendation (accommodation, transport, restaurant etc.)

1.1.1 Recommendation system functionality

Most of the tourism related recommendation systems provide several main functions including selection of attractions, POIs, hotel & restaurants, information recommendations and complete tour planning recommendations. In this section, we focused on tour planning recommendation systems. Figure 1.1 shows tour planning functionalities presented in the recommendation systems.

Single day tour planningfunction gives an option to plan one day tour in a city. Most of the tourist recommendation systems have that function but recently Multiple day tour planning function is developed due to user requirement and real time data. Some of the recommendation systems only suggest Accommodation and Restaurant

Selection which enables to select appropriate hotel and place to eat based on users preferences and budget.

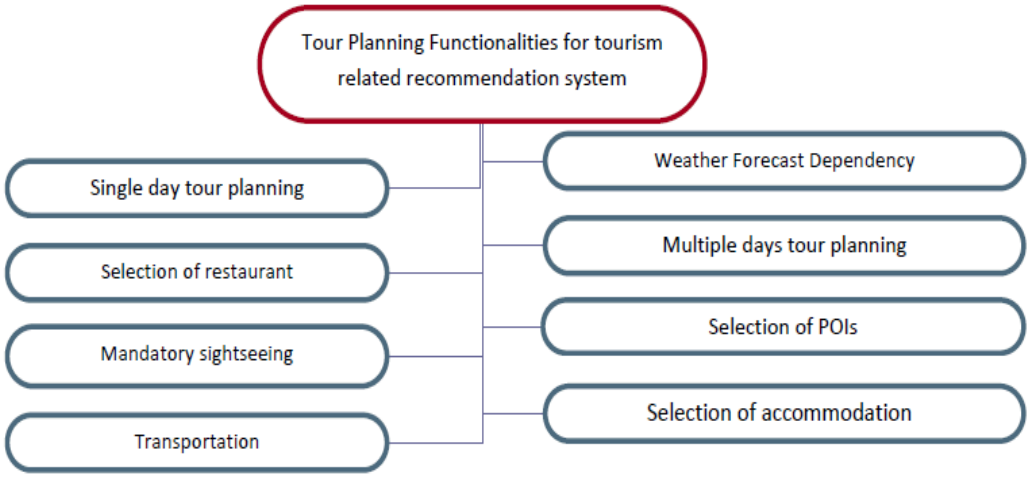


Figure1.1: Tour Planning Functionalities

Almost, all of the tourist recommendation systems offer selection of tourist attraction places and routing while very few of the take into account public transportation integration. The weather forecast dependency function can help to manage a tour i.e. if it is rainy outside, tourist can enjoy indoor visits. In this thesis, we propose a new tourist recommendation system which offers multiple day tour planning, selection of attraction places and integration of public transportation functionalities.

1.1.2 Existing recommendation systems for tour planning

Shiraishi.T et al., proposed P-Tour personal navigation system which provides near best schedule to visit multiple destinations under certain constraints. They formulated the planning as a multi-objective optimization problem and used the Genetic Algorithm based route search engine to solve their schedule planning problem. Authors used the satisfaction degree and the total travel expense as an evolution function in their algorithm(T. Shiraishi et al., 2005).

Then, P-tour navigation personal system was extended for multiple days tour by Kinoshita et al.[2006]. Authors designed the Genetic Alogrithm to tackle the scheduling problem in practical time and made various sighseeing schedules across multiple days using the digital map of the Tohoku area, Japan. They input 108 sighseeing spots, measured calculation time and quality of output schedules. P-Tour navigation system is implemented for single day tour planning. The P-tour system was re-extended by Nagata et al. [2006] in order to plan a tour for group of users. The advantage of this version is every person in the group can state a interest value, a duration and arrival time for each POI. The goal is to find a schedule that joins the group along the way to visit POIs.

Figure1.2:P-Tour personal navigation system user interface and[1]

J.Lee et al. [2007] implement an intelligent tour planning system based on the personalized tour recommender. They used Lin-Kernighen heuristic [Lin, 1965] to experiment 2^n TSP test instances. Their system initiate selecting candidate POIs sorted by their rank and scores determined by specific criterions. Then it build subset

[1]http://www.slideshare.net/NaokiShibata/ptour-a-personal-navigation-system-for-tourist

from candidate POIs to create corresponding tour schedule considering calculated rank and given constraints (J.Lee, E.Kang, G.Park, 2007).

The ROSE (Routing System) is a mobile application which combines event recommendation and pedestrian navigation with public transportation support presented by Ludwig B. et al. [2009]. In this context, they proposed $h_{\epsilon}u$ optimal algorithm. They used OpenStreetMap as map data. The system reacts in real time delays in the public transportation and calculates alternatives routes.

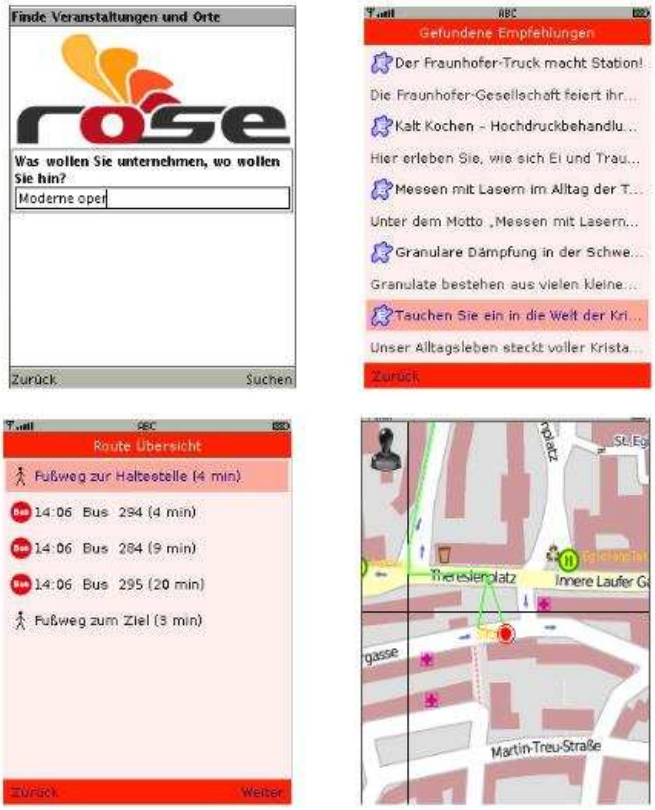


Figure 1.3: Screenshot of route recommendation system ROSE

The PECITAS built for the citizens and the travelers in Bolzano, Italy. This recommendation system calculates personalized route between two mandatory points in the city. Authors included public transportation in their system. PECITAS is formulated by knowledge based technology. Travel and user profile are introduced in order to rank different tours and to provide the highest ranked tour to user (Tumas.G.,Ricci.F, 2009).

Niaraki and Kim [2009] used Analytical Hierarchy Process method [Saaty,1980] to develop their personalizing route planning network impedances. The user state his preferences for attributes into the system .Then based on which weights in the road are calculated.

Yu and Chang [2009] presented a framework for the personalized recommendation system of hotels, restaurants and attractions. The author proposed nearest- neighbor constructive heuristic approach to tackle the problem with three functionalities.

R.A.Abbaspour and F.Samadzadegan used also the Genetic algorithm to solve the tour problem by scheduling an itinerary in multimodal urban transportation network. They analyzed transportation network of Tehran city as a case study. The main goal of their work is to get many score as possible considering mobile user interests, preferences and some restrictions of interested points. This system used geodatabase to access the transportation network of Tehran (R.A.Abbaspour and F.Samadzadegan , 2009). Based on their previous study R.A.Abbaspour and F.Samadzadegan proposed two adapted Genetic Algorithms to search for the solution f the shortest multimodal path finding. Their problem was time dependent tour planning in complex large urban areas for group of users. But the aim was still same as collecting the maximum total priority value from points of interest (R.A.Abbaspour and F.Samadzadegan, 2011).

Damianos gavalos et al. [2012] present DailyTRIP model which aims to maximize the overall score associated with visited POIs while not exceeding the daily time limit for sightseeing and introduce a novel heuristic that provides a near-optimal solution to solve the problem. Their approach takes into account user preferences, time, opening days, visiting time. They proved their proposed algorithm is suitable for online applications, whereas simulation results showed good performance (D.Gavalas, M.Kenteris, C.Konstantopoulos, G., 2012).

From the survey we can see that all of these recommendation systems and planners have a common structure by considering only the time budget constraint. However money budget and selection of transport mean functions are included in addition to total time limitation. On the other hand, every tour planning systems and architectures are modeled as a Tourist Tour Planning Problem and solved by exact algorithms, heuristics and meta-heuristics. The illustration of formulation is very important in problem modeling. In the following section 1.1.3 tourist tour planning problem (TTPP) is presented.

1.1.3 Tourist Tour Planning Problem

Once we decided to travel somewhere, we need to make a long list of things to do. Obviously, the most crucial things must be at the top of the list which is what to see, where to stay, where to eat and so on. When tourists visit a city or region, they cannot visit every available point of interest, as they are constrained in time and budget. It is not possible to visit every tourist attraction or interesting places during such a limited period, so the tourist has to make a selection of what is more important and valuable for him/her. Once the selection is made, the tourist keeps in mind the opening hours of the POIs, location, available time and entrance fee. They face many problems to solve. All of these requirements of the problem are considered as Tourist tour planning problem in the literature. The TTPP is illustrated by number of researchers and it has several versions and extensions. The main aim of the problem is to select attractions according to tourist's interest and preferences in order to maximize tourist satisfaction while considering several constraints. In the planning problem, visiting days, opening/closing hours, entrance fees, weather dependency, traveling distance between points of interest, visiting time required to visit each attraction, timetable of each attractions are considered as a main constraints.

The general TTPP assumes following data as an input (D.Gavalas, et al. 2012):

- The number of tours. Tourist decides how many days to stay at the place, based on that decision the number of the tours will be generated.

- The time windows for each tour. Tourist has to indicate the overall daily time limit to spend on everything on that day start by going out from the hotel until coming back to hotel. It includes visiting duration of attractions, traveling time between attractions (POIs), and having break.
- The candidate attractions (points of interest). Every point of interest has own attributes including its working timetable, location, type, rank, popularity, specialty and so forth.
- The time duration to visit each attraction. It can be forecasted from the average duration of visit and tourist's interest for that particular attraction.
- The traveling time between points of interest. Tourist can use every transportation means which are available at that destination. Thus, if necessary tourist can walk between attraction points.
- The satisfaction score of each point. This score indicates the weight of importance of each attraction based on tourist's preference.

Today, the number of researchers develops many theoretical approaches from the simplest version of tourist trip planning problem to the hardest version. The one of the most popular starting point to model TTPP is the Orienteering Problem. The OP consists of all necessity requirements of the TTPP. We introduce the OP in the Chapter 2.

1.2 Motivation

In this thesis, recent approaches with relevance to the TTPP are also examined, focusing on problem models that best capture a multitude of realistic user constraints, while also investigating several TTPP variants. We intend to study algorithmic techniques and methodologies concerning the problems related to tour planning problem. This dissertation deals with integrating the use of urban public transportation into the MCTOPTW problem. The problem takes into account the urban bus network, the travel time between locations will vary, and so far tourist has an option between walking and taking a bus. We present Iterated Local Search

algorithm to solve the Time Dependent Multi Constraint Team Orienteering Problem with Time Windows. Therefore, the proposed algorithm is applied to mobile based tour planning application so called, UB TOUR PLANNER.

1.2.1 Limitation of existing approaches

The reason why we integrate public transportation into the tour planning problem is transportation information has been identified as one of the most useful functionalities of tourist tour planning recommendation systems. In the literature review, authors successfully studied efficient algorithmic approaches to tackle the several variants of the Orienteering Problem which consists of some restrictions such as timetable of locations, entrance fees, travel distance and duration of the visits. However, they did not take into account the multi constraint problem with the use of public transportation. Inclusion of public transportation is much more complex as it has to consider traveling time, wide range of public transportation network including bus stops and bus lines, frequencies.

1.3 Thesis objectives

Based on the research gap of existing approaches we aim to integrate urban public bus to the problem and tackle the problem by Iterated Local Search meta-heuristic in this thesis.

- We investigated the theoretical foundations of several variants of tour planning problem.
- We studied the development of mathematical models and efficient algorithmic approaches to the problem of tourist tour planning.
- Therefore, we proposed the Iterated Local Search meta-heuristic to solve the time dependent multi constraint team orienteering problem with time windows.
- Finally, we applied and implemented the algorithm to tourist tour planning recommendation system.

1.4 Thesis outline

This thesis is organized as follows:

- Chapter 2 gives an overview of the state-of the-art in tourist tour planning problem. In the literature, the Orienteering Problem is confirmed as a very promising starting point to model the tour planning problem. This simplest case of tour planning problem represents all points as locations with assigned scores. According to the tourists need, this kind of tour planning problem has to be improved with other necessary constraints. Therefore, the several extensions of the OP and solution approaches are introduced in this chapter.
- Chapter 3 presents the problem statement of the Time Dependent Multi Constraint Team Orienteering Problem with Time Windows that addresses the tour planning requirements of tourist that visit city for a number of days. When including public transportation into the problem, it becomes time dependent problem. The time dependency is presented and integration of urban bus network is also illustrated. The main goal of the TDMCTOPTW is to maximize the tourist satisfaction (collect satisfaction score which are assigned to each point) while considering several constraints, parameters and respecting the timetable of (time window) available attraction point.
- In Chapter 4, our focus is on algorithmic approach to solve the hardest extension of the OP so called the TDMCTOPTW. The well-known local search meta-heuristic method is used to perform our algorithm.
- Chapter 5 describes the experimental validation of our approach that is tested with real life test set of Ulaanbaatar city. At the end of this chapter, we illustrate an application and implementation of our system. We implemented a mobile tourist recommendation system that enables to plan city tours for the Ulaanbaatar capital city of Mongolia.
- Finally chapter 6 concludes our research work.

Chapter 2

Background and State- of the -Art

Contents

Abstract. This chapter focused on overview of the research in the field Orienteering Problem with Time Windows, the Time Dependent TOPTW and the Multi Constraint TOPTW. In the Section 2.3, the Team Orienteering Problem and the used heuristic methods to solve the problem are introduced. The TOP is the extension with several day tours. In the following section, the TOP is extended with time windows. The problem is becoming more challenging in this section due to the

additional timetable constraint. At end of the chapter, time dependency constraint and multi constraint are presented.

2.1 The Orienteering Problem

Researchers identified the Orienteering Problem (OP) is one of most well-known starting point to model tourist tour planning problem. An OP consists of a set of locations that are determined by coordinates and a score. The pair wise travel times between the locations are known. The goal is to find a tour that maximizes the total score earned by visiting locations. The start and end of the tour do not need to coincide. The total travel time cannot exceed a predetermined value, which is called the time budget. Each location can be visited at most once (T.Tsiligirides, 1984).

Basically, the orienteering problem has originated from the group sport game so called "Orienteering" (I.Chao et al., 1996). The Orienteering is a family of sports that requires navigational skills using a map and compass to navigate from point to point in diverse and usually unfamiliar terrain, and normally moving at speed. Participants are given a topographical map, usually a specially prepared orienteering map, which they use to find control points[2]. In this game, players are given a map, points with coordinates and limited time. Then, players start at specified point, hurry to visit as many points as possible and return to the starting point. The main objective of the game is to collect the score which is associated to the each point. Obviously, players need to consider the total time constraint. The Orienteering Problem is based on the main concept of the Orienteering game.

In the literature, the Orienteering Problem is known as combination of two well-known optimization problems namely the Travelling Salesman Problem and the Knapsack Problem (P.Vansteenwegen et al.,2011).

[2] http://en.wikipedia.org/wiki/Orienteering

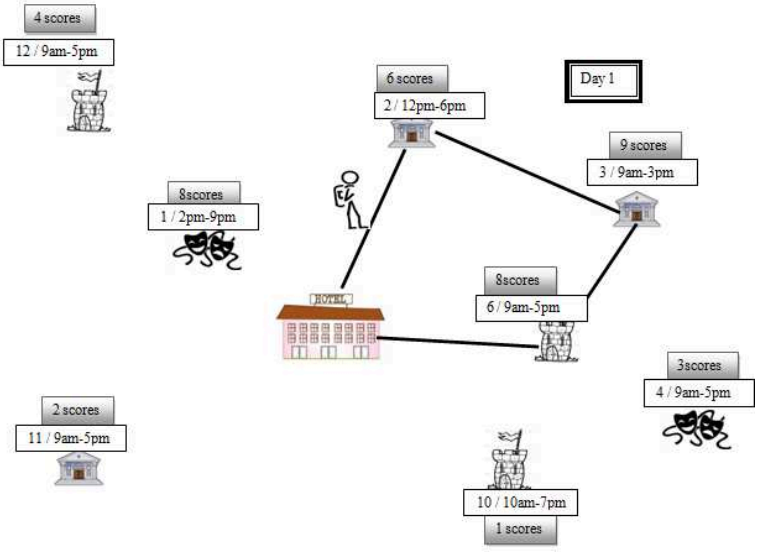


Figure 2.1: The Orienteering Problem

The Orienteering Problem showed some similarities to the popular Traveling Salesman Problem. This optimization problem is a NP-hard problem. TSP can be modeled as an undirected weighted graph, such that cities are the graph's vertices, paths are the graph's edges, and a path's distance is the edge's length. It is a minimization problem starting and finishing at a specified vertex after having visited each other vertex exactly once. Often, the model is a complete graph (i.e. each pair of vertices is connected by an edge). If no path exists between two cities, adding an arbitrarily long edge will complete the graph without affecting the optimal tour[3]. The objective of the TSP is to define the shortest path between the cities while try to visit as many cities as possible in the give time. The main difference of the TSP from the OP is the limited travelling distance, no matter of the collecting score and minimization of the travel distance.

TSP can be formulated as an integer linear program (Papadimitriou et al.,1998). Label the cities with the numbers 0, .., n and define:

[3]http://en.wikipedia.org/wiki/Travelling_salesman_problem

$$X_{ij} = \begin{cases} 1 \ the\ path\ goes\ from\ city\ i\ to\ city\ j \\ 0 \ otherwise \end{cases} \quad (2.1)$$

i = 1, ..., n

u_i- an artificial variable

C_{ij} - distance from city i to city j. Then TSP can be written as the following integer linear programming problem (Tucker, 1960):

$$min \sum_{i=0}^{n} \sum_{j \neq i, j=0}^{n} C_{ij} X_{ij} \quad (2.2)$$

$$0 \leq X_{ij} \leq 1\,, \quad \sum_{i=0, i \neq j}^{n} X_{ij} = 1, \quad i, j = 0, \dots, n \quad (2.3)$$

$$u_i - u_j + n x_{ij} \leq n - 1 \leq i \neq j \leq n \quad (2.4)$$

The first set of equalities (2.2) requires that each city be arrived at from exactly one other city, and the second set of equalities (2.3) requires that from each city there is a departure to exactly one other city. The last constraints (2.4) enforce that there is only a single tour covering all cities, and not two or more disjointed tours that only collectively cover all cities[4]. Meanwhile the OP shows some similarities with one of the famous problem in optimization problem so called knapsack problem or rucksack problem.

In the Knapsack Problem, given a set of items associated with mass and a value, aimed to define the number of each item to include in a collection so that the total weight is less than or equal to a given bound and the total value is as large as possible. It derives its name from the problem faced by someone who is constrained by a fixed-size knapsack and must fill it with the most valuable items. Let there be n items, z_1 to z_n where z_i has a value v_i and weight w_i. x_i is the number of copies of the item z_i, which, mentioned above, must be zero or one. The maximum weight that we can carry in the bag is W. It is common to assume that all values and weights are

[4] http://en.wikipedia.org/wiki/Travelling_salesman_problem

nonnegative. To simplify the representation, we also assume that the items are listed in increasing order of weight[5].

$$\text{Maximize} \quad \sum_{i=1}^{n} v_i x_i \qquad \text{subject to} \quad \sum_{i=1}^{n} w_i x_i \leqslant W, \qquad x_i \in \{0,1\} \tag{2.5}$$

In the OP, each location can be seen as an item from the Knapsack problem, its associated score is similar as the profit and the total time budget can be seen as the total capacity of the knapsack. But there is still a difference between these two optimization problem is that the total weight in the Knapsack problem is not dependent from the order of the selected items whereas the order of the visiting location is important in the Orienteering Problem because of the total travelling time depends on the order of the selected attraction point (W.Souffriau, 2010).

After the first application of the OP, there are several extensions that have been successfully used to model harder versions of the TTPP. The extensions of the OP:

1. Orienteering Problem with Time Windows (OPTW)
2. Team Orienteering Problem (TOP)
3. Team Orienteering Problem with Time Windows (TOPTW)
4. Time Dependent Team Orienteering Problem with Time Windows (TDTOPTW)
5. Multi Constraint Team Orienteering Problem with Time Windows (MCTOPTW)
6. Multi Constraint Team Orienteering Problem with Multiple Time Windows (MCTOPMTW)

The simple Orienteering Problem has extended by adding Time Windows. This problem considers visits to attraction points within a given time limit (opening and closing hours of locations). When the word “Team” is added as TOP, it extends the problem by taking into account multiple day tour or it can be multiple people (group of tourist).

[5] http://en.wikipedia.org/wiki/Knapsack_problem

The TOPTW denotes multiple day tour and pre-defined time window. And the TDTOPTW used to model multimodal transportation through attractions. This problem considers time dependency in the estimation of moving time from one point to another point. Therefore, the Multi constraint team orienteering problem (MCTOPTW) is used to model the hardest problems including multiple tasks. The MCTOPTW considers several constraints such as total time budget, total money budget, maximum n-type of point of interest and so on.

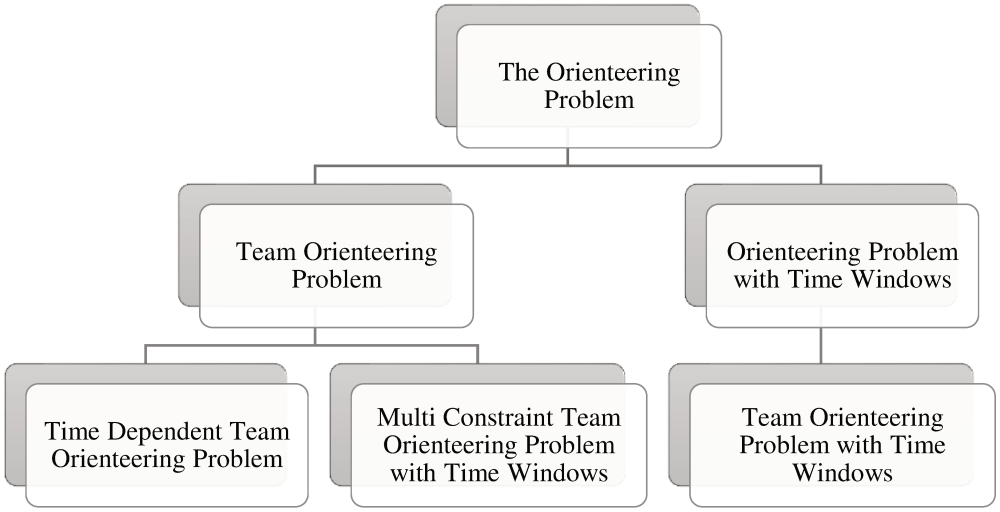


Figure 2.2 : Extensions of the TOP

2.1.1 Mathematical Formulation

The Orienteering Problem has a set of given points and each with associated score. The starting and ending points are fixed and traveling time between these points is known for all points. Total time budget is limited. For every tour from 1 to N point, X_{ij} is equal to 1 if visit i is followed by visit j, otherwise equal 0. The objective function is to determine the tour while maximizing the satisfaction score. In the OP, time is limited so cannot visit all of the points (P.Vansteenwegen et al., 2011).

Table2.1: Parameters and Decision variables

Notations	Descriptions
X_{ij}	= 1 if a visit point i is followed by a visit point j, 0 otherwise
Si	satisfaction score of point i
n	number of points
t_i	time duration to visit point i
T_{max}	time budget of each tour
t_{ij}	travel time between point i to point j
u_i	the position of the point i.

The Objective Function (2.6) is to maximize the total collected satisfaction score when visiting the points:

$$Max \sum_{i=1}^{n-1} \sum_{j=2}^{n} S_i X_{ij} \quad (2.6)$$

Constraint (2.7) shows fixed number of starting point 1and ending point n:

$$\sum_{j=2}^{n} X_{1j} = \sum_{i=1}^{n-1} X_{in} = 1 \quad (2.7)$$

Constraint (2.8) ensures connection of the points and every point is visited at most once.

$$\sum_{i=1}^{n-1} X_{ik} = \sum_{j=2}^{n} X_{kj} \leq 1 \quad (2.8)$$

$$\forall k = 2, \dots, n-1$$

Constraint (2.9) ensures total time limit (available time) of the tour.

$$\sum_{i=1}^{n-1}\sum_{j=2}^{n} t_{ij}X_{ij} \leq T_{max} \tag{2.9}$$

Constraint (2.10) and (2.11) show that single tour have to be constructed.

$$2 \leq u_i \leq n \forall i = 1, \dots, n \tag{2.10}$$

$$u_i - u_j + 1 \leq (n-1)\big(1 - X_{ij}\big); \qquad \forall i, j = 2, \dots, n; \tag{2.11}$$

$$X_{ij} \in \{0,1\}, \qquad \forall i, j = 1, \dots, n$$

Sub-tours are not allowed. These sub-tours elimination constraints use an extra variable u_i to order the points in the tour (Miller et al., 1960).

2.1.2 Exact solution methods for the Orienteering Problem

Number of exact algorithms (Zülal Sevkli et al. 2006) for the OP are introduced such as Variable Neighborhood Search, Branch and Bound method by G.Laporte and S.Martello (1990), Branch and cut approach by M.Fischetti et al. (1998) and integer programming by A.C.Leifer et al.(1994) and so on.

G.Laporte and S.Martello (1990) presented the Integer Linear Programming formulations to solve the selective travelling salesman problem. They also used a standard constraint relaxation algorithm. The branch and bound approach is used if violated conditions have occurred. M.Fischetti et al. (1998) described exact and heuristic separation algorithms and heuristic procedures to generate near-optimal solutions. Authors used five classes of additional inequalities for OP in their algorithm. Their branch and cut algorithm works in two stages. First stage branch cover cuts in order to avoid branching. Second stage works on sparse graph which is

resulted from branch cover cuts. They used classical branching strategy to close the integrality gap. Authors assumed that the second stage took advantage from a large number of relevant cuts.

0-1 integer programming model for the orienteering problem is introduced in (A.C. Leifer and M.Rosenwein, 1994). They focused on tightening the linear programming relaxation by adding constraints and valid inequalities. The proposed method aims to obtain upper bounds by tackling three linear programs. They begin to model 0-1 integer programming for the OP then relax the 0-1 conditions and put the other constraints. The cutting plane algorithm is used in order to get additional inequalities from constraint conditions. The OP with small sized data sets can be solved in reasonable amount of time by these exact methods above. Nevertheless, they are very time consuming if the amount of instances are increased.

2.1.3 Heuristic and meta-heuristic methods for the Orienteering Problem

The Orienteering Problem is referred as NP-hard problem (B.L.Golden et al., 1987), so there is no polynomial time algorithm that has been introduced to solve it. They assume that exact algorithms are very time consuming rather than heuristics which are more efficient. Furthermore, Gendreau et al. (1998) present a Tabu Search algorithm to tackle the Selective Travelling Salesman Problem. The OP is also called as Selective Travelling Salesman Problem (STSP). Authors assume that STSP is more difficult to model compared to the simple TSP because of its two separate attributes, the score of the each vertex and the travelling distance between the vertices. They are both independent so it become complicated to select the particular vertex. The tabu search heuristic takes into account cluster of vertices rather than one single vertex for insert and remove step.

Also, the OP is modeled as a multi-level optimization problem in (I.Chao etal. 1996). First, they choose subset of points to visit at the first level, then the TSP must be solved at the second level. Their heuristic approach includes two steps initialization

and improvement. They generate an ellipse over the entire set of points by using starting and ending points. The limited time budget is used as the length of the major axis. In order to generate a route, they only take into account the points that are in the ellipse because of the others outside the ellipse already violated the total time budget constraint. The greedy method is used to insert new points into the ellipse. Among all paths, the one with the highest score is chosen as the solution. This initial solution can be improved by two-point exchange. A point i is moved from the set of other paths and inserted onto the initial solution path. In same time, point j is moved from initial solution path and inserted onto the set of other paths in order to perform two-point exchange. They also used one-point movement at the time between paths. At the clean-up step, they applied two-opt improvement in order to make shorten the length of the path. They assume there is more chance to insert point from set of the paths onto the initial solution path by decreasing the length of the path. Authors test their heuristics using benchmark test instances with 67 problems, then they generate more 40 new test instances in order to apply their heuristic. The result has been shown computationally efficient and well on all test instances.

F.Tasgetiren (2001) propose genetic algorithm to solve the orienteering problem. Based on his previous study, author presents variable length permutation representation and injection crossover operator. Additionally, he adds a penalty function to the Genetic algorithm in order to penalize the infeasible solution since the orienteering problem is constraint problem. Proposed algorithm begins by constructing initial population based on distance between vertices. Two parents are iteratively generated by tournament selection procedure to produce an offspring. He applied local search method including several operators namely add, omit, replace and swap to individuals selected randomly from crossover operator. This procedure is iterated until the final criterion is reached. The algorithm is applied on 67 problems from the four test set and compared to previous methods. The computational results show the GA approach was able to outperform.

(A.Levi et al, 2006) introduce the first algorithm based on Variable Neighborhood Search (VNS) to solve the Orienteering problem. They consider the Euclidean OP

where the graph is a complete graph and the score of the point is the distance between points. They applied three versions of VNS namely VNS with VND (Variable neighborhood descent), VNS with RVNS (reduced VNS) and Pure RVNS. Authors explain these variations of VNS and the neighborhood structures used in these variations. In the VNS, VND is used as local search method. But RVNS differs by selecting random solutions from neighborhood without having local search method. The last variation of VNS used RVNS instead of local search phase of basic VNS. They test their three variations of VNS with 107 benchmark test problems. It over performs previous two approaches.

Table 2.2:Approximation algorithms for the Orienteering Problem

Reference	Directed OP / Undirected OP	Time
(C. Chekuri, N. Korula, and M. Pal., 2008)	Undirected	Polynomial
(C. Chekuri and A. Kumar., 2004)	Directed	Polynomial
(C. Chekuri and M. Pal., 2005)	Undirected	Quasi-polynomial
(V. Nagarajan and R. Ravi., 2011)	Undirected	Polynomial
A. Blum, S. Chawla, D. R. Karger, Lane T., A. Meyerson, and M. Minko, 2007)	Directed	Polynomial
(N. Bansal, A. Blum, S. Chawla, and A. Meyerson., 200)	Directed	Polynomial

Another effcient multi-level metaheuristic is proposed by (W.Souffriau et al, 2008). They intoduced an upper-level algorithm to determine the most appropriate set of parameters for lower-level metaheuristic, Ant Colony Optimization (ACO). An upper-level Genetic Algorithm is used to simulate the natural evolution in order to train ACO algorithm. Authors explain multi-level structure based on the general ant colony optimization. An ACO algorithm performs differently based on the actual setting of the parameters. The ACO is well known approach to tackle other optimization problems such as the Vehicle Routing Problem (VRP) (Bullnheimer, 1999), the travelling salesman problem (TSP) (Feillet, D., Dejax, P., Gendreau, M, 2005), the graph coloring problem (Costa, D., Hertz, A, 1997) and so forth.

They simulate natural evaluation using an upper–level genetic algorithm (GA), in order to train a general ACO algorithm. Ants build solutions to the OP by moving probabilistically from one point to another and the transition rule decides which location the ant should go to next. The considered locations are in the candidate list. Ants choose their move probabilistically according to a transition function, which is based on local observations, such as the length of the arc to traverse and the amount of pheromone on it. After adding a location to the solution, the available budget is reduced by the time needed for the movement and a certain amount of pheromone evaporates. The evaporation of pheromone is also known as local pheromone update (Bullnheimer, 1999).

Recently, A.Bock and L.Sanita (2014) introduce another variant of OP so called Capacitated Orienteering Problem. They consider the OP as subroutine and present a $(3 + \varepsilon)$-approximation algorithm by focusing on natural generalization considering node demand and capacity bound. Also, authors give a Polynomial time approximation scheme for Capacitated OP on tree metrics and on Euclidean metrics.

Table 2.2 summarizes the approximation algorithms for the Orienteering Problem in directed and undirected graphs (D.Gavalos, Ch.Konstantopoulos, K.Mastakas. G.Pantziou and Y.Tasoulas, 2012).

2.2 Team Orienteering Problem

The idea of the team orienteering problem is from the sport with several competitors. It is the extension of single competitor orienteering game. The team consists of more than 2 members, they start the game at the same starting point and each member of the team aim to visit as many points as possible until the ending point, under the given time limit. The team orienteering problem keeps these concepts from the game. It allows multiple tours; each tour must be under given time budget.

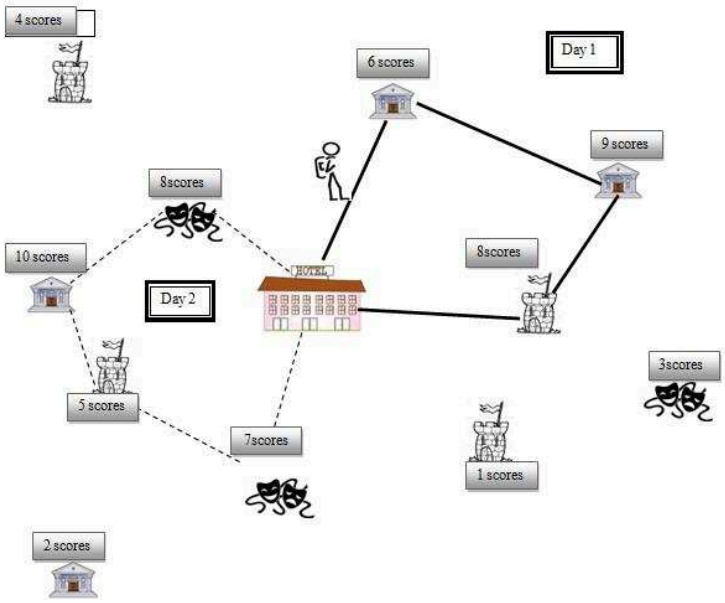


Figure2.3:The Team Orienteering Problem

Butt and Cavalier (1992) model high school athlete recruitment as a TOP under the name of multiple tour maximum collection problem. The recruiter has to visit schools nearby in order to pursuit member for the football team and return to his college. His goal is to maximize the recruiting potential but he has to come back to his college in same day and can meet only high school students during their class time. On the other hand, Tricoire et al., (2010) define the problem as scheduling customer visits of sales representatives. They develop an exact algorithm for the route feasibility check when having multiple time windows. Boussier et al., (2007) propose a method based on easily adaptable approach so-called branch and price algorithm in order to solve team orienteering problem. Their algorithm is based on dynamic programming.

2.2.1 Mathematical Formulation

Based on the existing mathematical formulations of the OP, the TOP can be formulated as shown below.

Table2.3:Parameters and Decision variables

Notations	Descriptions
X_{ijd}	1 if a visit point i is followed by a visit point j in a tour d, 0 otherwise
Y_{id}	=1 if point i is visited in tour d, 0 otherwise
S_i	satisfaction score of point i
M	number of tour
N	number of points
T_i	time duration to visit point i
T_{max}	time budget of each tour
T_{ij}	travel time between point i to point j
u_i	the position of the point i.

The Objective function (2.12) is maximizing satisfaction score in tour d:

$$Max \sum_{d=1}^{m} \sum_{i=2}^{n-1} S_i Y_{id} , \quad (2.12)$$

The (2.13) constraint shows fixed number of starting point 1and ending point n.

$$\sum_{i=1}^{N-1} X_{ikd} = \sum_{j=2}^{N} X_{kjd} = Y_{kd} , \quad \forall_k = 2, \dots, N-1 , \forall_d = 1, \dots, m \quad (2.13)$$

Next constraint (2.14) ensures connection of the points and every point is visited at most once.

$$\sum_{i=1}^{N-1} X_{ikd} = \sum_{j=2}^{N} X_{kjd} = Y_{kd} , \quad \forall_k = 2, \dots, N-1 , \forall_d = 1, \dots, m \quad (2.14)$$

Constraint (2.15) determines every point is visited at most once.

$$\sum_{d=1}^{m} Y_{kd} \leq 1 , \quad \forall_k = 2, \dots, N-1 \quad (2.15)$$

Constraint (2.16) shows limited total time budget for each tour.

$$\sum_{i=1}^{n-1}\sum_{j=2}^{n} t_{ij}X_{ijd} \leq T_{max}\,\forall_d = 1,\dots,m \tag{2.16}$$

Lastly, (2.17) and (2.18) prevents sub tours.

$$2 \leq u_{id} \leq n \forall i = 1,\dots,n,\ \forall_d = 1,\dots,m \tag{2.17}$$

$$u_{id} - u_{jd} + 1 \leq (n-1)\big(1 - X_{ijd}\big); \qquad \forall i,j = 2,\dots,n; \forall_d = 1,\dots,m \tag{2.18}$$

$$X_{ijd} \in \{0,1\}, \qquad \forall i,j = 1,\dots,n\ , \forall_d = 1,\dots,m$$

2.2.2 Exact solution methods for the Team Orienteering Problem

As far as we know authors (S.Boussier, D.Feillet and M.Gendreau, 2007)present the first exact algorithm for team orienteering problem. Authors evaluate this algorithm on two different problems namely the TOP and another one is selective Vehicle Routing Problem with Time Windows. Their aim in this work is to propose a generic Branch and Price scheme which can solve any kinds of orienteering problem. As we mention in several part of this thesis, the OP has many additional constraints depending on its application. But branch and price algorithm has ability that most of these constraints only affect the sub-problem used to generate new columns. In the subsequent section the algorithm is written based on dynamic programming. The main idea is to associate a label with each possible partial path until the best feasible path reached and to extend these labels checking the resource constraints. They solve 270out of the 387test instances. The remaining 117 instances could not be solved in 2 hours.

2.2.3 Heuristic and meta-heuristic methods for the Team Orienteering Problem

Authors in (H.Tang and E.Miller-Hooks, 2005) first introduce one of the well-known approach Tabu Search heuristic to solve the TOP. Their Tabu Search algorithm consists of three steps; initialization, solution improvement and evaluation. The initialization step is embedded in an Adaptive Memory Procedure. At the AMP current solution S must be determined and given, set tabu parameters to small neighborhood stage in which only a small number of neighborhood solutions will be explored. Then based on the current tabu parameters generate a number of feasible and infeasible solutions to the current solutions are generated by random procedures and greedy procedures. Basically the second step of the algorithm is focused on the generation of neighborhood solutions and improving these solutions. At the evaluation step, select the best non-tabu solution from the generated candidates at the improvement step. They tested and compared 320 test instances published from literature. The Tabu Search algorithm provides best average solutions in 14 out of 18 instance categories. The other instances got obtained close to the best average solution values by Tabu Search procedures.

Furthermore, C.Archetti et al. (2007) propose two well-known meta-heuristics for TOP which are calledthe Generalized Tabu Search Algorithm and the Variable Neighborhood Search Algorithm. The tabu search algorithm is developed with two possible different strategies. First strategy is to explore the set of feasible solutions; the second one is to visit admissible but not necessarily feasible solutions. They implement tabu search algorithm using two types of moves; 1-move and swap-move, respectively. In order to measure the quality of the solutions visited during the search, they used five functions including: the total profit of the routes in set of most profitable routes in s, total duration of the routes in set of most profitable routes in s, feasibility of the parameter s, number of the non-empty routes, total duration of the routes in all remaining routes. Authors test and compare three algorithms the generalized tabu feasible algorithm, the generalized tabu penalty algorithm and VNS

feasible algorithm. They test 320 benchmark test instances and with the proposed algorithm 128 of benchmark instances are shown improvement on the best known solution.

Again, the Ant colony optimization approach used to solve TOP in (L.Ke, C.Archetti, Z.Feng, 2008). They propose four methods to construct candidate solutions in the framework so called sequential, deterministic-concurrent, random concurrent and simultaneous method. The main procedure of ACO is starting by initialization of all parameters and then ants constructs feasable solutions at each iteration. Each ant construct a feasible solution according to the transition rule, that one or more solution can be improved during local search procedure. The iteration process repeats until the maximum number of cycles has been reached. In the ACO framework, as mentioned above there are four construction methods that are illustrated to constructing feasible solution. In the sequential method complete tours are generated one after another tour. In the random-concurrent method, in order to insert new point tour is selected randomly at every iteration. In the deterministic-concurrent method, the order of the tours is fixed. In the simultaneous method, a point is inserted to one of the tours until all tours reach their limited length at every iteration process. Then authors used local search procedure based on (I.Chao, B.Golden, E.Wasil, 1996) by using 2-opt move to shorten each tour and add as many feasible solution as possible.

Table2.4:Summary of heuristic algorithms for Team Orienteering Problem

Reference	Algorithm	Technique
(P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden., 2009)	GLS (guided local search)	Guided Local Search
(P.Vansteennwegen, W.Souffria, G.V.Berghe, D.V.Oudheusden, 2009)	SVNS (skewed variable neighborhood search)	Variable Neighborhood Search
(S.Butt and T.Cavalier, 1992)	BC	Greedy Insertions
(W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden., 2010)	FPR (fast variant of the algorithm) SPR(slow variant of the algorithm)	reedy randomized adaptive earch procedure with path relinking

(Chao, B. L. Golden, and E. A. Wasil., 1996)	CGW	Local search
C.Archetti, A.Hertz, M.Speranza, 2007)	SVN (slow variable neighborhood search) VN (fast variable neighborhood search) TS (tabu search)	Variable neighborhood search Tabu search
(L.Ke, C.Archetti, Z.Feng, 2008)	ASe (sequential) ADC (deterministic concurrent) ARC (random-concurrent) ASi (simultaneous)	Ant colony optimization

This procedure repeats until no improvement can be made. They made computational experiment with seven test sets including 387 benchmark instances from literature.

A hybrid method called Memetic Algorithm approach is presented in (H.Bouly, D.C.Dang, A.Moukrim, 2008). The Memetic algorithm is one of the recent techniques, combination of the genetic algorithm with local search method. A Genetic Algorithm considers solutions as a chromosome. Thus it needs encoding process to extract solutions from chromosomes. Authors use Optimal split procedure as the decoding process. In their algorithm, an evaluation procedure involves the splitting procedure corresponding to the chromosome decoding. Therefore, diversification process is obtained through mutation operation in order to avoid homogeneity in the population. During the mutation operation, authors use different neighborhood which is selected in random order. The well-known local search method works as mutation operator. They tested their algorithm on standard test instances of the TOP from previous literature. The Memetic Algorithm improves the best known solutions of 11 benchmark test instances from the literature.

In (Y.Kurata, 2009), researcher introduces another tour planning system CT-Planner. Author adopt cyclic model where the system shows a set of tour plans to the user, takes into account user's interest and preferences based on the user's response, then generate a new set of plans. The Collaborative Tour Planning system aims at the user-

driven planning rather than computer based tour planning. Selection of one preferable tour from the number of possible tours becomes hard to solve. However author makes paired comparison by using Analytical Hierarchy Process (AHP) in order to display only two tour plans each time. He applies well known multi-criteria decision analysis to select the best tour plan from several possibilities. He uses five-dimensional unit vector to model user's tour preference.

2.3 Team Orienteering Problem with Time Windows

The most common extension of the OP is the OPTW (Orienteering problem with Time Windows) and the TOPTW (Team orienteering problem with Time Windows). Extensions with time windows became hard to solve because of its constraint. In the OPTW, each point of interest has associated time window. The time window represents opening hour and closing hour of the each attraction. In this extension, if arrival happens before opening hour, waiting time is allowed until the opening hour of attraction point. In order to avoid violation of the constraint, ending time of the tour must be before or same time as closing time of the last visit. For the TOPTW, multiple days of tour planning is considered. In that case each point of interest (attraction) has an identical timetable (opening and closing time) for any day.

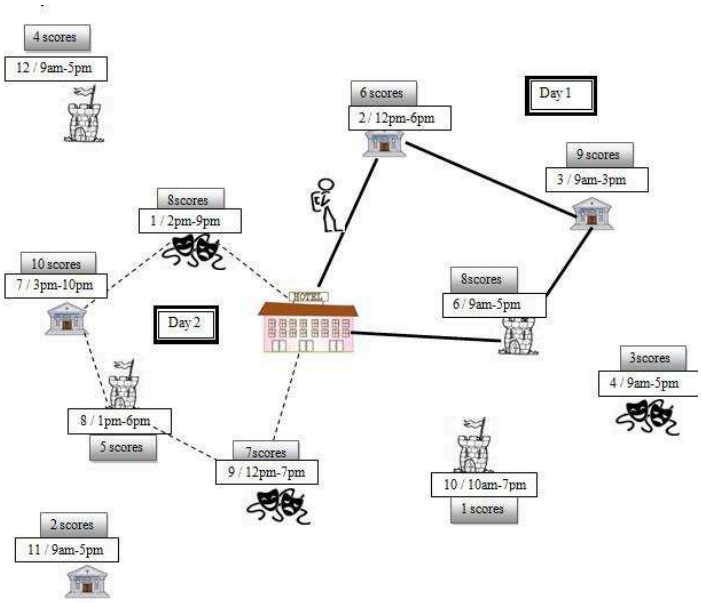


Figure2.4:The Team Orienteering Problem with Time Windows

2.3.1 Mathematical Formulation

The TOPTW can be formulated as an integer programming problem, in tour d, Xijd=1 if a visit i followed by visit j, 0 otherwise. Based on the notation presented on previous section, the TOPTW can be written as follows:

Table2.5:Parameters and Decision variables

Notations	Descriptions
X_{ijd}	= 1 if a visit point i is followed by a visit point j in tour d, 0 otherwise
Y_{id}	=1 if point i is visited in tour d, 0 otherwise
S_i	satisfaction score of point i
M	number of tour
N	number of points
t_i	time duration to visit point i
T_{max}	time budget of each tour
t_{ij}	travel time between point i to point j
[Oi, Ci]	Time window (opening/closing hour)
V_{id}	Start of the visit i in tour d
D	Artificial variable

The objective function (2.19) maximizes the total collected score:

$$Max \sum_{d=1}^{m} \sum_{i=2}^{n-1} S_i Y_{id} \ , \ Y_{id} \in \{0,1\} \ , \forall_{i,j} = 1, \dots, N \ , \ \forall_d = 1, \dots, m \tag{2.19}$$

Constraints (2.20) ensure each tour m start by point 1 and ends at point N.

$$\sum_{d=1}^{m} \sum_{j=2}^{N} X_{1jd} = \sum_{d=1}^{m} \sum_{i=1}^{N-1} X_{iNd} = m \ , \ X_{ijd} \in \{0,1\} \tag{2.20}$$

Constraints (2.21) guarantee that each visit to a point is followed by another visit to a next point.

$$\sum_{i=1}^{N-1} X_{ikd} = \sum_{j=2}^{N} X_{kjd} = Y_{kd} \ , \ \ \forall_k = 2, \dots, N-1 \ , \forall_d = 1, \dots, m \tag{2.21}$$

Following constraint (2.22) allows that every point is visited at most once in a tour while constraint (2.23) limits the time budget.

$$\sum_{d=1}^{m} Y_{kd} \leq 1 \ , \ \forall_k = 2, \dots, N-1 \tag{2.22}$$

$$\sum_{i=1}^{N-1} (t_i Y_{id} + \sum_{j=2}^{N} t_{ij} X_{ijd}) \leq T_{max} \ , \ \forall_d = 1, \dots, m \tag{2.23}$$

Constraint (2.24) restricts timeline of each tour.

$$v_{id} + t_i + t_{ij} - v_{jd} \leq D(1 - X_{ijd}) \ , i, j = 1, \dots, N; d = 1, \dots, M \tag{2.24}$$

The constraint (2.25) shows the start of the visit must be in its time window.

$$O_i \leq V_{id} \leq C_i, \ \forall_i \in 1, \ldots, N, \ \ \forall_d \in 1, \ldots, m \tag{2.25}$$

2.3.2 Heuristic and meta-heuristic methods for the (Team) Orienteering Problem with Time Windows

The Iterated Local Search meta-heuristic is presented in (P.Vansteenwegen, W.Souffriau, G,V.Berghe, D.V.Oudhesden, 2009). General local search algorithms explore neighborhood by iteratively generating the neighborhood of the current solution and moving from this current solution to an improving neighboring solution. This process is repeated until the current solution cannot be improved anymore until local optimum is reached. This algorithm iteratively generates a neighborhood of insert moves and selects the move with highest ratio. But this local search approach gets stuck in a local optimum. So there is a need to escape from these local optima. In the literature [Lourenco et al., 2003] solved this problem with Iterated Local Search (ILS) meta-heuristic. Researchers proved this approach can successfully tackle this kind of problem with time windows. The ILS keeps the general procedure of local search algorithm but it created a sequence of local search instead of random repeats of local search.

In the ILS they provide the insertion step to insert a new point into the tour one by one and the shake step as a diversification to escape from local optimium. The wait and maxshift parameters are calculated in order to avoid time consuming feasibility checking process. Based on the ratio calculation, selection of the point with the highest ratio for insertion is made. Then after each insertion process every point need to be updated. Furthermore, the shake step removes one or more points from the tour. After that step in order to avoid waiting, every point following the removed ones are relocated at the beginning of the tour. They tested the available benchmark instances. The average gap between ILS and the best known solution for all these instances is only 1.8%.

Therefore, authors in (P.Vansteennwegen, W.Souffria, G.V.Berghe, D.V.Oudheusden, 2009) present the application of Guided Local Search and Variable Neighborhood Search Method to tackle team orienteering problem. Also, they apply iterated local search method for team orienteering problem with time windows. Based on the utility function, guided local search penalizes unwanted solution features during the each local search iteration. During each iteration, the objective function is decreased by penalty. The penalty helps to escape from local optimum and to continue the search. Six steps are implemented in guided local search algorithm namely Insert, Replace, 2-opt, Swap, Disturb and Group. The GLS algorithm starts with Construct procedure and then the local search heuristics are implemented in 2 loops.

Furthermore, they applied SVNS (skewed variable neighborhood search) for TOP. The meta-heuristic VNS changes the neighborhood by escaping from the local optimum using shaking step. The SVNS is recent extension of VNS which is suitable for solving problems with near-optimal solutions. In SVNS algorithm, in order to generate the neighborhood, all the intensification steps are used. They test on large number of test instances from literature. The SVNS algorithm outperforms the GLS algorithm.

Authors also focus on application of iterated local search algorithm to solve TOPTW in their work. Regarding to the ILS method, they combine two steps in order to escape from local optimum. Firstly, authors apply insertion step which can add new points one by one to a tour. The shake step removes one or more points from the tour. It mainly focuses on to escape from local optimum. After the removal procedure, all the visits following the removed ones are transferred forward as much as possible. This procedure prevents unnecessary waiting time. The iterated local search algorithm tested on existing benchmark instance. The ILS algorithm performs very good on a large number of problems with up to 288 points. The average computation time is 1.6s and for 39 problems with range of 3 up to 20 tours, the average gap is 2.1% and the worst gap is 10%.

Table2.6: Summary of heuristics for the Team Orienteering Problem with Time Windows

Reference	Algorithm	Technique
(P.Vansteenwegen, W.Souffriau, ,V.Berghe, D.V.Oudhesden, 2009)	LS (Iterated Local Search)	erated Local Search meta-heuristic
. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden., 2009)	GLS (guided local search) VNS (variable neighborhood search)	Guided local search and Variable neighborhood search
(N.Labadi, J.Melechovsky, R.W.Calvo, 2010)	GRASP-ELS	reedy randomized adaptive search procedure (GRASP) and the Evolutionary local search
(D.Gavalas, M.Kenteris, Ch.Konstantopoulos, G.Pantziou, 2011)	DailyTRIP	euristic for deriving near-optimal solution
(J.K.Chilinska and P.Zabielska, 2013)	GA	Genetic algorithm Iterated local search

The OP and the TOPTW are solved by an effective hybrid metaheuristics in (N.Labadi, J.Melechovsky, R.W.Calvo, 2010). This approach is combination of two known method which are the Greedy Randomized Adaptive Search Procedure (GRASP) and the Evolutionary Local Search (ELS). The GRASP generates random solutions using randomized heuristic and improved by a local search procedure. The ELS is extension of the iterated local search and it generates multiple copies of solutions, and then applies ILS on each copy. Their approach behaves as a random heuristic if the perturbation is too strong otherwise the solutions are trapped in local optima very fast. That is why perturbation phase is very important in this approach. Authors test their approach with two benchmark data by Solomon's and Cordeau's. The GRASP-ELS approach improved 141 best known solutions out of 304 tests and found best known solutions in 118 cases.

(D.Gavalas et al. 2011) introduce, DailyTRIP, personalized recommendations for daily sightseeing itineraries. They propose a heuristic for deriving near-optimal personalized daily tourist itineraries aiming to maximize the overall profit. Their algorithm includes five execution phases and considers user preferences, required time to visit attractions, opening time. In the first phase, definition of the problem's model is involved. In order to decrease computational effort required to final valid solutions they reduce the problem space in the second phase. Phase three and phase four are for selection of set of nodes for first daily itinerary and constructing itinerary trees. Phase five is optional and goal is to improve previous solutions which are created on previous phase. The last phase is for transferring trees to multipoint lines through a post-order traversal of the corresponding trees.

Recently, (J.K.Chilinska and P.Zabielska, 2013) propose the genetic algorithm approach to tackle orienteering problem with time windows. They used main idea of well-known iterated local search method. The genetic algorithm consists of four steps including initialization, selection, crossover and mutation step. The initialization step starts by encoding a solution into a chromosome. Since the number of points in the tour is not set, the length of the chromosome is not fixed. This step is inspired from insertion step and shake step of ILS method. In the next step, they used tournament grouping selection. In the crossover step, two random individuals are selected for the crossover phase. Then they find the genes to replace without violating the time window constraint. The random tour is selected from the individuals in the mutation step. There are two types of mutation process: insertion mutation and delete mutation. Insertion mutation considers all chances to inclusion of every new gene. Delete mutation removes the selected genes except first and last ones in order to shorten tour length. Authors use same benchmark instance from literature and computational result show that mutation step can improve the solution. It outperforms the result of the iterated local search algorithm.

2.4 Summary

So far, we have been given the state-of the-art and background regarding to tourist tour planning problem. More specifically, the Orienteering Problem and its extended versions are introduced and mathematical formulations are provided. After every section of this chapter, an overview of existing efficient methods is presented. We summarized the exact algorithms, heuristic and meta-heuristic techniques since they will be needed to compare the result of the experimental validation. The promising results and newly generated data set of the iterated local search algorithm for the multi constraint team orienteering problem open new opportunities for further study on this subject.

Chapter 3

Problem statement

Content

Abstract. In this chapter we discuss the new model formulation by integrating time dependency constraint into the multi constraint team orienteering problem with time windows. In the section 3.1, existing time dependent tour planning model is illustrated. We integrate the use of public transportation to the problem by modeling the TDMCTOPTW in the section 3.2

3.1 Time Dependency for tour planning problem

Time dependency is mostly used to model traveling between points using multimodal transportation. Basically, this problem considers calculating the estimation of travelling time from one point to another point, therefore. The time dependent team orienteering problem with time windows is well known problem which comprises a number of points with associated data such as location of point, time windows (opening and closing hours) given score and so on. In the TDTOPTW, traveling between particular two points can be done by any of public transportation means or on foot. In this section, we introduce the extended TOPTW problem with use of public bus.

3.1.1 Time Dependent Team Orienteering Problem with Time Windows

As aforementioned, the TDTOPTW problem extends the TOPTW considering time dependent traveling time between attraction places by using public transportation. Traveling time between places depends on the leaving time of the point i and the transportation mode.

There are several things that make the TDTOPTW become tougher than others which are information and data related to public transportation network. It consists of a number of stops and different lines between these stops, each with a given frequency. In (R.Abbaspour and F.Samadzadegen, 2011) time dependent tour planning problem in urban area is presented. They model their problem as a TDOPTW and propose two adapted genetic algorithmic approach to solve. Authors assume that the previous literature reviews did not take into account any real dataset. They test their proposed framework and formulation using real dataset from Tehran city, Iran.

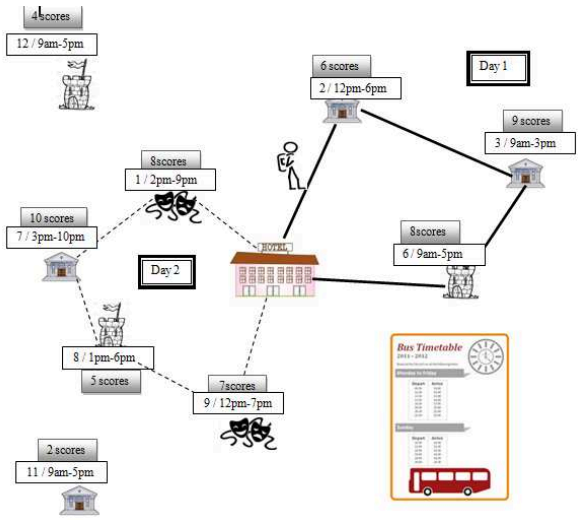


Figure3.1:The Time Dependent Team Orienteering Problem with Time Windows

3.1.2 Mathematical Formulation of the TDTOPTW

In the time dependent team orienteering problem with time windows, there is a give set of points, tourist has to visit the maximum number of points under limited time constraint aiming to maximize the satisfaction score. Herein, the traveling time between attraction places is fixed.

Table3.1:Parameters and Decision variables:

Notations	Descriptions
X_{ijdt}	= 1 if a visit point i is followed by a visit point j in tour d at the time period t, 0 otherwise
S_i	satisfaction score of point i
M	number of tour
n	number of points
t_i	time duration to visit point i
T_{max}	time budget of each tour
t_{ij}	travel time between point i to point j
v_{idt}	the start of the visit at point i in tour d, started at the time t
$[O_i, C_i]$	time window of point I Oi = opening time Ci = closing time

The objective function of the problem (3.1) is to maximize the collected satisfaction score when visiting points at certain time periods.

$$max \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \sum_{t=1}^{Tmax} S_i \times X_{ijdt} \tag{3.1}$$

Constraint (3.2) ensures that there is no sub tour (return tour) while constraint (3.3) describes starting point is 1.

$$\sum_{i>1}^{n} \sum_{t=1}^{Tmax} X_{i1t} = 0 \tag{3.2}$$

$$\sum_{j>1}^{n} \sum_{t=1}^{Tmax} X_{1jt} = 1 \tag{3.3}$$

Constraint (3.4) and constraint (3.5) ensures that the last visited point is point n.

$$\sum_{j=1}^{n-1} \sum_{t=1}^{Tmax} X_{njt} = 0 \tag{3.4}$$

$$\sum_{i=1}^{n-1} \sum_{t=1}^{Tmax} X_{int} = 1 \tag{3.5}$$

Next constraint guarantees (3.6) each point must be visited at most once.

$$\sum_{i=1}^{n-1} X_{ik} = \sum_{j=2}^{n} X_{kj} \leq 1 \tag{3.6}$$

The last constraint implies (3.7) every visit must be between its time windows. The time window represents an interval of daily opening and closing hour of that particular point.

$$O_{id} < V_{idt} < C_{id} \tag{3.7}$$

3.1.3 Heuristic and meta-heuristic methods for the Time Dependent Team Orienteering Problem with Time Windows

In (R.A.Abbaspour and F.Samadzadegen, 2011), authors propose the tour planner engine and the multimodal shortest pathfinder engine based on adapted genetic algorithms. First engine serves as main routine and second engine is modeled as subroutine that is called whenever it is necessary. Addition to the basic formulations of the OP, they formulized one key parameter which is the weight of paths connecting POIs. According to their formulations, the objective function comprises two general parts which are minimizing the weights of used arcs and minimizing the waiting time. The Genetic Algorithm is used as an engine to tackle tour planning problem. In order to create an itinerary, genetic algorithm engine uses a geospatial database and multimodal shortest path module. The first steps of the algorithm are coding the chromosome and initialization. Then the natural selection is made. At the third step authors use the roulette wheel pairing method for selection. Then mating and mutation steps are made. The process from step 2-5 is iterated until the termination criterion is satisfied and an acceptable solution is reached. They use data from Tehran city, capital of Iran. The computational experiment is made with 324 points which are categorized into 6 parts.

On the other hand, (A.Garcia et al., 2013) adapt the heuristic approach based on existing method for TOPTW, so called Iterated Local Search. They present personalized electronic tourist guide by modeling time dependent team orienteering problem with time windows. The ILS algorithm is designed based on the algorithm proposed by (P.Vansteenwegen, W.Souffriau, G,V.Berghe, D.V.Oudhesden, 2009). The general procedure is done step by step until the termination solution is reached. Since the TDTOPTW is more complicated problem than the general TOPTW, it needs more adapted method to tackle this problem. Authors propose two different approaches to tackle the public transportation problem. First approach is based on pre-calculation of the average traveling time for each pair of points in order to handle

the integration of the public transportation. Then, they made some concepts into the basic operators of the ILS. The first approach has two main parts. Firstly, the pre-calculation does not have the real-time requirement; secondly, in practice the average travel time is rather accurate due to the high frequency public transportation service. They design some concepts that allow fast local evaluation of each possible insertion in the second approach. They present a model considering the periodicity of the bus services which is limited to direct bus connections, having no transfer between lines. Finally, they extend this model to include transfers in public transportation, either pre-calculating some required values. They test the algorithm using real data set from San Sebastian in the Basque Country. They created 28 test instances to test their two approaches. The tours generated with two methods are very similar and there is average gap of 2.7% between them for 1 day tour, below 2% for two days tour.

Table3.2: Summary of heuristic algorithms for the Time dependent team orienteering problem

Reference	Algorithm	Technique
(R.A.Abbaspour and F.Samadzadegen, 2011)	AGA (adapted genetic algorithms)	adapted genetic algorithms
(A.Garcia, P.Vansteenwegen, Arbelaitz, W.Souffriau, M.T.Linaza, 2013)	ILS	Iterated Local Search
(D.Gavalas, Ch.Konstantopoulos, K.Mastakas, G.Pantziou and N.Vathis, 2013)	TDCSCroutes SlackCSCroutes	Cluster based Heuristics
(D.Gavalas, Ch.Konstantopoulos, K.Mastakas, G.Pantziou and N.Vathis, 2015)	TDCSCroutes SlackCSCroutes	Iterated Local Search Cluster based heuristics

3.2 Multi-Constraint Team Orienteering Problem with Time Windows

The MCTOPTW includes set of locations, each of them with a certain score, a time window and one or more associated attributes, such as an entrance fee, max-n types, mandatory POI types etc. Each attribute type has an associated constraint with a maximum allowed value for a route, such as a limited budget. Visiting a location within its time window allows collecting its score as a reward. The goal is to determine routes that maximize the collected score without violating any of the constraints. The starting point 1 and ending point N of every tour are fixed. Traveling time between POI i and j is known for all points. In the following, we rewrite problem definition and the mathematical formulation of the MCTOPTW problem presented by A.Garcia et al, 2010.

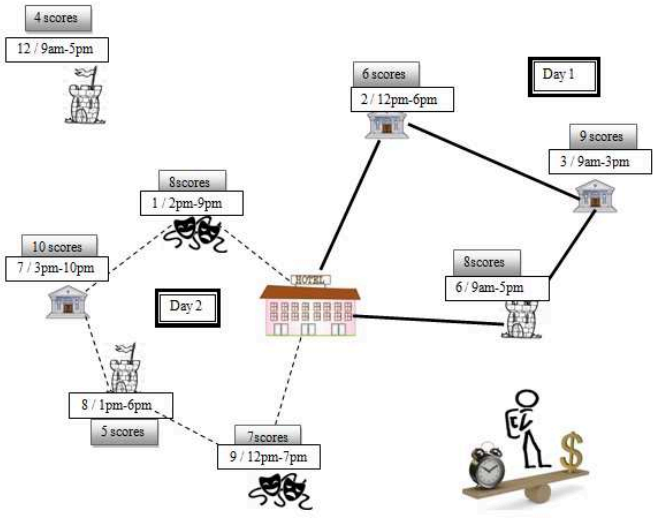


Figure 3.2 :The Multi Constraint Team Orienteering Problem with Time Windows

3.2.1 Mathematical Formulation

Table 3.3:Parameters and Decision variables

Notations	Descriptions
X_{ijd}	= 1 if a visit point i is followed by a visit point j in tour d, 0 otherwise
Y_{id}	=1 if point i is visited in tour d, 0 otherwise
S_i	satisfaction score of point i
M	number of tour
N	number of points
Ti	time duration to visit point i
Tmax	Total time budget of each tour d
Tij	travel time between point i to point j
Vid	the start of the visit at point i in tour d
[Oi, Ci]	time window of point i Oi = opening time Ci = closing time
f_{id}	spent money to visit point i in tour d (entrance fee)
F_{max}	Total money budget for each tour
e_{izd}	= is set to 1 if location i is category z in tour d, 0 otherwise
E_z	maximun number of visits of the particular category z

The Objective Function (3.8) is to maximize the total collected satisfaction score:

$$Max \sum_{d=1}^{m} \sum_{i=2}^{n-1} S_i \, Y_{id} \ ,$$
$$Y_{id} \in \{0,1\}, \forall_{i,j} = 1, \dots, N \ , \ \forall_d = 1, \dots, m \qquad (3.8)$$

Constraint (3.9) ensures each tour m start by point 1 and ends at point N.

$$\sum_{d=1}^{m} \sum_{j=2}^{N} X_{1jd} = \sum_{d=1}^{m} \sum_{i=1}^{N-1} X_{iNd} = m \ , \ X_{ijd} \in \{0,1\} \qquad (3.9)$$

Constraints (3.10) guarantee that each visit to a point is followed by another visit to a next point.

$$\sum_{i=1}^{N-1} X_{ikd} = \sum_{j=2}^{N} X_{kjd} = Y_{kd} \ , \quad \forall_k = 2, \dots, N-1 \ , \forall_d = 1, \dots, m \tag{3.10}$$

Following constraints (3.12) allow that every point is visited at most once in a tour

$$\sum_{d=1}^{m} Y_{kd} \leq 1 \ , \ \forall_k = 2, \dots, N-1 \tag{3.11}$$

Constraints (3.12) and (3.13) limit time budget of each tour, also timeline of each tour.

$$\sum_{i=1}^{N-1} (t_i Y_{id} + \sum_{j=2}^{N} t_{ij} X_{ijd}) \leq T_{max} \ , \ \forall_d = 1, \dots, m \tag{3.12}$$

$$V_{jd} = V_{id} + t_i + t_{ij} \ , \quad X_{ijd} = 1 \tag{3.13}$$

Constraint (3.14) restricts start of the visit in its time window.

$$O_i \leq V_{id} \leq C_i \ , \forall_i \in 1, \dots, N, \forall_d \in 1, \dots, m \tag{3.14}$$

Last two constraints (3.15) and (3.16) limit money budget for each tour and limit value of attribute constraint z of the point i.

$$\sum_{i=1}^{N} \sum_{d=1}^{m} f_{id} Y_{id} \leq F_{max} \ , \forall_i \in 1, \dots, N, Y_{id} \in \{0,1\} \ , \forall_d \in 1, \dots, m \tag{3.15}$$

$$\sum_{d=1}^{m} \sum_{i=1}^{N} e_{izd} Y_{id} \leq E_z, Y_{id} \in \{0,1\} \ , \forall_z = 1, \dots, Z \ , \tag{3.16}$$

3.2.2 Heuristic and meta-heuristic methods for the Multi Constraint Team Orienteering Problem with Time Windows

Tour planning problem is strongly NP-hard problem and there are certain efficient methodologies and techniques are introduced today to solve this kind of problem. The MCTOPTW is extended version of the TOPTW by adding multiple constraints. In the literature, (Montemanni and Gamberdella, 2009) solved the TOPTW by using Ant Colony System approach and recently (Lin and Yu, 2012) illustrated a Simulated Annealing approach to handle the TOPTW. According to studies none of previous TOPTW algorithms can solve this problem with multiple constraints. But A.Garcia et al [2010] and Sylejmani, K.,et al [2012] became the first to describe meta heuristics to tackle the MCTOPTW. A.Garcia et al [2010] used Iterated local search based algorithm which was already successfully used to solve TOPTW. Furthermore, they include new feasibility checks, a new ratio function to compare possible insertions and tabu list inside the perturbation phase. Authors compared their work with an existing method published for the SVRPTW (Selective Vehicle Routing Problem with Time Windows), their algorithm proves to solve the problem efficiently with an overall average gap of 4.4% and an overall average computation time of only 2.4 seconds.

Table 3.4: Heuristics methods and benchmark instances

Methods	**Authors**	**Origin of benchmark instances**	**umber of test nstance**	**Number f tours (m)**	**Number Of oints(N)**	**Multi const-Raint**
Ant Colony System	Montemanni and amberdella(2009)	Instances for TOPTW :				
		Solomon (c10,r10,rc10)	29	2,3,4	50	
		Solomon (c10,r10,rc10)	29	2,3,4	100	
		Cordeau (pr1-pr10)	10	2,3,4	48-288	
		Solomon (c20,r20,rc20)	27	2,3,4	100	
		Cordeau (pr11-pr20)	10	2,3,4	48-288	
Simulated Annealing heuristic	in and Yu (2012)	Instances for TOPTW :				
		Solomon (c10,r10,rc10)	29	2,3,4	50	
		Solomon (c10,r10,rc10)	29	2,3,4	100	
		Cordeau (pr1-pr10)	10	2,3,4	48-288	
		Solomon (c20,r20,rc20)	27	2,3,4	100	
		Cordeau (pr11-pr20)	10	2,3,4	48-288	

Iterated Local Search	A.Garcia et al (2009)	Instances for TOPTW : Solomon (c10,r10,rc10) Cordeau (pr11-pr20)	29 10	9-19 3-20	100 48-288	
Iterated Local Search	.Vansteenwegen et al., (2010)	Instances for MCTOPTW: Solomon (c10,r10,rc10) Cordeau (pr1-pr10) Solomon (c20,r20,rc20) Cordeau (pr1-pr10)	29 10 27 10	1 1 2 1	100 48-288 100 48-288	2 2 2 2
Tabu search approach	.Sulejmani et al., (2012)	ıstances for MCTOPTW: P.Vansteenwegen et al	148	1,2,3,4, 1-4	100 48-288	2

Additionally, since there were not any tests instances for MCTOPTW they develop new test set for MCTOPTW with one, two routes and 1, 2 attributes based on TOPTW test set. For 39 problems with one tour and 2 attribute constraints, the average gap with optimal results is only 3.9% in 1.1 seconds average computation time. As for problem with 2 tours, the average gap with the best known results is 0.9% and average computation time is 4 seconds. Due to its simplicity and the high quality results, the algorithm can easily be applied for problems with more attribute constraints.

Thereafter, (Sylejmani, K., et al 2012) propose A Tabu search approach for multi constrained team orienteering problem with time windows. They use same test instances as (A.Garcia et al. 2010). In this work, they apply three basic operators namely Insert, Replace, Swap and four additional operators so-called Delete, Perturbation, Restart and Penalize to escape from local optima. In 57.43% of test instances, the best solutions of (A.Garcia et al. 2010) were found or improved. So far, according to their results in 70 test instances, their algorithm shows up better solutions than results of (A.Garcia et al. 2010). We explain these two approaches in detail in next section 2.4.2.1 and section 2.4.2.2.

3.2.3 Iterated Local Search Method for the MCTOPTW

The ILS for TOPTW checks only the time windows feasibility. But since MCTOPTW is included extra attribute constraints, it needs to check more insertion feasibility of each constraint. Based on their well-known method ILS which successfully tackled the TOPTW, P.Vansteenwegen et al. adapt their heuristic to solve the MCTOPTW [P.Vansteenwegen et al., 2010]. They make few changes by inspecting each constraint's feasibility for each non included location and change the ratio function. Previous ratio function assumes only associated score of point and required time to visit that point. Nevertheless, there is a need to take into account attribute constraints to calculate the ratio. Authors in [P.Vansteenwegen et al., 2010] analyze certain different variants to define the best ratio function for the MCTOPTW. They keep nominator as a previous ratio function and suggested several different denominators by adding same weight for all constraints, a special weight for each attribute constraint k, and include the upper bound for each attribute constraint so forth. Finally, authors formulate the best ratio function by the combination of two functions. This option gives a special weight to each attribute constraint and includes the available quantity of each constraint on the route. Their empirical tests show the optimal weight for the attribute constraints was obtained by setting the weight of each constraint as the inverse of the number of constraints e.g. 0.5 for 2 attribute constraints [P.Vansteenwegen et al., 2010]. Then, authors make another change which actually improved the TOPTW algorithm. In the TOPTW, it was possible for points removed during one iteration to be inserted again immediately during the next iteration. So they try to avoid removing same points during consecutive iterations. A tabu list created in this shake step to improve the quality of the algorithm. At the end of the work, authors test their heuristic using existing test set for the Selective Vehicle Routing Problem with Time Windows SVRPTW. Therefore, they design new test set since there is no available test instances for the MCTOPTW.

3.2.4 Tabu Search approach for the MCTOPTW

After the ILS [P.Vansteenwegen et al., 2010] approach for MCTOPTW, [K.Sylejmani et al. 2013] introduce the Tabu Search method to tackle this problem. They use tabu memory that has same duty as "tabu list" from the literature to keep record of the moves that cannot be performed for a certain number of iterations. Authors apply 3 basic operators so-called Insert, Replace and Swap to explore the neighborhood and 4 different extra operators to make search diversification process. While insert operator aims to insert a new point into the tour from the other non-included points, replace operator exchanges a point from the tour with other point from out of the tour. The swap operator exchanges between any two points inside the tour. In the [P.Vansteenwegen et al., 2010] they allow waiting time to start a visit but they do not. Instead of recording Wait and Maxshift value to accelerate the time window feasibility check process, authors save two variables for each point inside the tour meaning how much the point in tour m could be shifted forward or backward. Whenever a new point is inserted, replaced or swapped the corresponding two values need to be updated.

Authors (K.Sylejmani et al., 2012) make experiments using same data instances from the literature and compare their results with the results of ILS heuristic. They conclude that their algorithm outperforms by ILS heuristic regarding the average performance in the set of instances.

3.3 Integration of public transportation constraint

As aforementioned, tourist faces several problems to decide what to see, where to stay and which activities to do, how much money to spend and so on. Therefore, their next step is to decide sequence of the attraction points and to decide how to move from one attraction point to another following one. This kind of problem is tackled by few researchers and there is very limited literature survey on this. As authors knowledge the integration of public transportation constraint with the Orienteering Problem is presented only in (A.Garcia, P.Vansteenwegen, o.Arbelaitz, W.Souffriau, M.T.Linaza, 2013) , (Ander Garcia, Olatz Arbelaitz, Pieter Vansteenwegen,Wouter

Souffriau, and Maria Teresa Linaza, 2010) and (D.Gavalas, Ch.Konstantopoulos, K.Mastakas, G.Pantziou and N.Vathis, 2015). So far, there is no paper takes into account the use of public transportation into the optimization problem with all constraints including multi constraints, time window, money budget and time dependency.

Integration of public transportation differentiates the problem from the other tour planning problem by its certain characterizations as follows:

- The public transportation networks consists of fixed number of bus stops
- The fixed number of bus lines
- Different lines between stops, each with given frequency

In the existing tour planning problems, the traveling time between the attraction points is always fixed. Thus it was easier to solve rather than the time dependent problem. In the TDTOPTW, tourist can choose walking or using public bus in order to get the place. Thus traveling time between attraction points depends on leaving time of the previous visiting place and decision of the transportation mode.

3.3.1 Integrating public transportation into the MCTOPTW

Integrating public transportation constraint into the optimization problem is one of the hardest issues to solve. Especially the problem like the multi constraint team orienteering problem with time windows which is definitely NP-hard problem itself and plus adding new constraint of public transportation is becoming very challenging task to handle.

In the literature, researchers use the time dependent team orienteering problem with time windows as a starting point to model in order to tackle tourist tour design problem with public transportation. In (R.A.Abbaspour and F.Samadzadegan, 2011), authors propose an architecture including tour planning block as the main routine and

the finding multimodal shortest path is modeled as a subroutine that is called whenever required. The authors' model their two engines based on evolutionary approach so called the genetic algorithm. These two engines interacts each other and include source/destination points, tour duration, starting time and transportation modes information. In order to search the multimodal shortest path route, they store the important information about transportation network, stations and service lines timetable. As the input of the algorithm, 500 points with different numbers of nodes and starting times are chosen to evaluate the multimodal shortest path algorithm. The authors discuss three different cases to illustrate the result.

1. First case is the longest one; all kind of public transportation means are used including taxi, bus, subway and even walking.
2. Second case is also combination of three transportation modes.
3. Third case is combination of walking and bus.

At the result, it indicates that number of iterations of the genetic algorithm increases as the tour duration increases. Also, sometimes the algorithm cannot find the tour during the experiment.

3.3.2 Modeling the new TDMCTOPTW problem

We propose the TDMCTOPTW model based on the mathematical formulation presented by (A.Garcia, P.Vansteenwegen, Wouter Souffriau, Olatz Arbelaitz , Maria Teresa Liaza, 2010). Later, this model is used in the implementation of the UB TOUR PLANNER (see chapter 5), this problem includes set of locations, each of them with a certain score, a time window, limited budget and time dependency of public transportation. Visiting a location within its time window allows collecting its score as a reward. Furthermore, the use of public bus is integrated. The bus network is included as well as bus stops. Tourist can move between two particular places by public bus or by foot.

The goal is to determine a tour that maximizes the collected score without violating any of the constraints. The starting point 1 and ending point N of every tour are fixed. Traveling time between POI i and j is known for all points. In the following, we rewrite problem definition and the mathematical formulation of the TDMCTOPTW problem based on equations presented by A.Garcia et al, 2010.

In addition to the presented mathematical formulation, we add an extra constraint for total money budget (F_{max}) which limits total money spent during a tour (d). This constraint equals entrance fee and tax to visit attraction point i in tour d. But tourist's accommodation, restaurant and personal shopping cost is not included.

Table3.5: Parameters and Decision variables

Notations	Descriptions
X_{ijdt}	= 1 if a visit point i is followed by a visit point j in tour d, in time t, 0 otherwise
Y_{idt}	=1 if point i is visited in tour d, in time t, 0 otherwise
S_i	satisfaction score of point i
M	number of tour
N	number of points
t_i	time duration to visit point i
T_{max}	Total time budget of each tour
t_{ij}	travel time between point i to point j
v_{idt}	the start of the visit at point i in tour d, in time t
[Oi, Ci]	time window of point I Oi = opening time Ci = closing time
f_{id}	Entrance fee to visit point i in tour d
Fmax	Total money budget for each tour

The objective function of the problem (3.17) is to maximize the collected satisfaction score when visiting points at certain time periods.

$$max \sum_{i=1}^{n} \sum_{j=1, j\neq i}^{n} \sum_{t=1}^{Tmax} S_i \times X_{ijdt} \quad (3.17)$$

$$Y_{id} \in \{0,1\}, \forall_{i,j} = 1, \dots, N, \qquad \forall_d = 1, \dots, m$$

Constraint (3.18) ensures that there is no sub tour (return tour) while constraint (3.19) describes starting point is 1.

$$\sum_{i>1}^{n} \sum_{t=1}^{Tmax} X_{i1t} = 0 \tag{3.18}$$

$$\sum_{j>1}^{n} \sum_{t=1}^{Tmax} X_{1jt} = 1 \tag{3.19}$$

Constraint (3.20) and constraint (3.21) ensure that the last visited point is point n.

$$\sum_{j=1}^{n-1} \sum_{t=1}^{Tmax} X_{njt} = 0 \tag{3.20}$$

$$\sum_{i=1}^{n-1} \sum_{t=1}^{Tmax} X_{int} = 1 \tag{3.21}$$

Next constraint guarantees (3.22) each point must be visited at most once.

$$\sum_{i=1}^{n-1} X_{ik} = \sum_{j=2}^{n} X_{kj} \leq 1 \tag{3.22}$$

Constraint (3.23) implies every visit must be between its time windows. The time window represents an interval of daily opening and closing hour of that particular point.

$$O_{id} < V_{idt} < C_{id} \tag{3.23}$$

Constraints (3.24) and (3.25) limit time budget of each tour, also timeline of each tour.

$$\sum_{i=1}^{N-1} (t_i Y_{id} + \sum_{j=2}^{N} t_{ij} X_{ijd}) \leq T_{max}, \forall_d = 1, \dots, m \tag{3.24}$$

$$V_{jd} = V_{id} + t_i + t_{ij}\ , \quad X_{ijd} = 1 \tag{3.25}$$

Constraints (3.26) guarantee that restrict that start of the visit in its time window.

$$O_i \leq V_{id} \leq C_i,\ \forall_i \in 1, \ldots, N,\ \ \forall_d \in 1, \ldots, m \tag{3.26}$$

Next constraints (3.27) limit money budget for each tour and limit value of attribute constraint z of the point i.

$$\sum_{i=1}^{N} \sum_{d=1}^{m} f_{id} Y_{id}\ \leq F_{max}\ \ , \ \forall_i \in 1, \ldots, N,$$

$$Y_{id} \in \{0,1\}\ , \forall_d \in 1, \ldots, m \tag{3.27}$$

3.4 Summary

This chapter extends the problems of the previous chapter to take time dependency into account. The time dependency is mostly used when the problem includes public transportation integration while multi constraint integrates the more attribute constraints such as money budget, mandatory categories and so on.

Firstly, we emphasize the existing time dependent problem and the approaches to deal with that problem. Then we presented integration of the public bus into the problem with multi constraint and explain the mathematical model formulation of the time dependent multi constraint problem with time windows.

Chapter 4

Technique and Methodology

Content

Abstract. In this chapter we explain the heuristic approach to tackle the TDMCTOPTW. The heuristic method is based on Iterated Local Search which has shown efficient result by solving the MCTOPTW(P.Vansteenwegen, W.Souffriau, D.V.Oudheusden, 2011). The ILS is a meta-heuristic approach based on iteratively building sequences of solutions generated by an embedded heuristic called local search. In the section 4.1, the basic local search heuristic method is explained.

Section 4.2 presented iterated local search meta-heuristic and proposed new algorithm to tackle the TDMCTOPTW.

4.1 Local Search Heuristic

Heuristic search refers to techniques with the aim of finding 'good' solutions for a very hard optimization and decision within a reasonable amount of computation time. Local Search Heuristic technique that works with complete solutions and seeks to find better solutions by making small local changes. All heuristic search techniques share similar concepts; e.g. the search space, feasible/infeasible solutions, neighborhoods, and the relation(s) between neighbors[6]. Local search is an iterative algorithm that moves from one solution S to another *S'* according to some neighborhood structure. Local search procedure usually consists of the following steps[7].

1. Initialization. Choose an initial schedule S to be the current solution and compute the value of the objective function F(S).
2. Neighbor Generation. Select a neighbor *S'* of the current solution S and compute F(*S'*).
3. Acceptance Test. Test whether to accept the move from S to *S'*. If the move is accepted, then *S'* replaces S as the current solution; otherwise S is retained as the current solution.
4. Termination Test. Test whether the algorithm should terminate. If it terminates, output the best solution generated; otherwise, return to the neighbor generation step.

[6]http://unow.nottingham.ac.uk/resources/resource.aspx?hid=5ade2b04-6d82-79cf-24b1-236084d32121

[7]http://community.stern.nyu.edu/om/faculty/pinedo/scheduling/shakhlevich/handout10.pdf

4.1.1 Insertion of neighborhood

As aforementioned, basic local search based algorithms explore a neighborhood by iteratively generating the neighborhood of the current solution and moving from this current solution to an improved neighboring solution. This process is repeated until the current solution cannot be improved anymore, and thus local optimum reached (W.Souffriau, 2010).

In this section the local search based on insertion of neighborhood is described which is well known to solve the optimization problems with time windows(P.Vansteenwegen, W.Souffriau, G,V.Berghe, D.V.Oudhesden, 2009). The insertion step aims to add new visit of attraction points into a tour one by one. But, due to the time window constraint the problem becomes very hard to insert new visits into the tour without violating their time window. Therefore, there is necessity to verify that all visits scheduled after the insertion place still satisfy their time window an extra visit can be inserted in a tour.

In the literature, there is a need to do a quick evaluation of each possible insertion thereto develop fast heuristic, thus the parameters called Wait and MaxShift are generated in order to avoid taking too much time for checking all other visits on their feasibility(A.Garcia, P.Vansteenwegen, Wouter Souffriau, Olatz Arbelaitz , Maria Teresa Liaza, 2010), (A.Garcia, P.Vansteenwegen, o.Arbelaitz, W.Souffriau, M.T.Linaza, 2013). In the following sections, useful parameters are discussed in detail.

4.1.2 Wait and Maxshift

The definition of the $Wait_{id}$ parameter is waiting time in case the arrival time at point A_{id} takes place before the time window in tour d. The service of attraction points can only start when the time window opens, see equation 4.1. Obviously, $Wait_{id}$ equals to zero if the arrival at attraction point takes place during the time window.

$$Wait_{id} = max[0, O_i - A_{id}] \tag{4.1}$$

The Maxshift is explained as the maximum delayed time of the service completion of the given visit without making any visit infeasible. Maxshift of the point i is equal to the sum of Wait and Maxshift of the next point i+1, unless Maxshift is limited by its own time window (closing time of point i), see equation 4. 2.

$$Maxshift_{id} = min\left[C_i - V_{id}, Wait_{i+1,d} + Maxshift_{i+1,d}\right] \tag{4.2}$$

4.1.3 Shift and Ratio

In equation (4.3) the total time consumption to insert an extra visit j between visits i and k in tour d is determined as $Shift_{ijkd.}$

$$Shift_{ijkd} = t_{ij} + T_j + Wait_{jd} + t_{jk} - t_{ik} \tag{4.3}$$

After the calculation of wait, maxshift and shift, the ratio should be calculated for each visit in order to determine the visit that will be selected for insertion.

$$ratio_{ijkd} = \frac{S_i^2}{Shift_{ijkd}} \tag{4.4}$$

Therefore, visits after the insertion require an update of the arrival time, starting time, Wait and Maxshift. The local search method iteratively generates a neighborhood of insertion step and applies the visit with highest ratio score. But, this local search procedure has a disadvantage of stacking in local optimum solution. So in order to escape the local optimum there is a need of diversification procedure.

4.2 Iterated Local Search Meta-heuristic

The importance of high performance algorithms to tackle difficult optimization problems cannot be understated, and in many cases the most effective methods are meta-heuristics(H.Lourenco, O.Martin, T.Stutzle, 2003).In the ILS, a sequence of local search solutions is made instead of random repeats of the local search procedure. The ILS meta-heuristic perturbs the solution found by the local search to generate new solution. After that it takes the best solution as the new starting solution for the local search. The procedure is iterated until a termination condition is reached. According to basic local search heuristic the procedure terminated by selecting the visit with highest ratio score for insertion. However, it gets stuck to the local optimum and could not give the best feasible solution. Thus, the diversification step is added in the iterated local search algorithm. In (P.Vansteenwegen, W.Souffriau, G,V.Berghe, D.V.Oudhesden, 2009), the Shake step is executed in order to avoid the local optima. This phase aims to remove one or more consecutive visits (POI) from the tour. The shake step is presented in section 4.2.1.

4.2.1 Shake phase

In the shake phase, two integers are used as an input.

- First integers defined as how many consecutive visits to remove in a tour

- Other one indicates the position in the tour to start the removing process.

If during the removal process the end point is reached, it continues after the start point. Because of the varied tour length, during the performance of the algorithm the value of the second integer will become different for different tours.

After the removal process in order to avoid unnecessary waiting time, all visits following the removed visits are shifted towards the beginning of the tour. Due to its time window if a visit cannot be shifted that visit and other visits sequenced after it will not be changed. At the end, the shifted visits should be updated as the process mentioned in previous section. Until the termination condition is reached, shake phase and the local search heuristic are performed. Finally, the heuristic returns the incumbent solution as the result. The ILS meta-heuristic can be summarized as follows:

```
PositionStartRemove=1;
NumberToRemove=1;
NumberOfTimesNoImprovement=0;
maxNumberToRemove=NumberOfPOIs/(3*NumberOfDays);
maxIter=factorNoImprovement * SizeOfFirstTour;
While NumberOfTimesNoImprovement < MaxIter do
While not local optimum do
        For each non included visit;
            Determine the best possible insert position and Shift;
 Calculate Ratio;
          Insert visit with highest ratio;
If Solution better than BestFound then
       BestFound=Solution;
       NumberToRemove=1;
       NumberOfTimesNoImprovement=0;
else
```

```
    NumberOfTimesNoImprovement+1;
Shake Solution (NumberToRemove,PositionStartRemove);
PositionStartRemove=PositionStartRemove+NumberToRemove;
NumberToRemove+1;
PositionStartRemove >= Size of smallest Tour;
If NumberToRemove=maxNumberToRemove then
NumberToRemove=1;
Return BestFound;
```

Listing 4.1 Pseudo code of Iterated Local Search meta-heuristic

As aforementioned the local search heuristic adds new visits to a tour one by one. The least insert time ($Shift_j$) is calculated for each visit i. Then ratio should be determined for each of these visits by dividing the score of the point to the time required to visit that point. Heuristic selects the point with highest ratio for insertion based on the ratio calculation. This process is iterated until no more point can be inserted. The Shake step is applied in order to remove one or consecutive points from a tour. At the end of heuristic, the heuristic returns the incumbent solution as the result.

4.3 The TDMCTOPTW Algorithm

In order to solve the Time Dependent Multi Constraint Team Orienteering Problem with Time Windows, we need to adapt the Iterated Local Search method. Before inserting the most expected attraction point, many evaluations of possible insertions are taken into account every iteration of the algorithm. Therefore, adding use of public transportation into the problem makes the evaluation of possible insertion very difficult to tackle.

In real time, tourist can face many simple problems such as spending little more time to visit at the current attraction point and miss the bus to travel to next attraction point. Thereupon, tourist needs to wait for the next bus or he can walk to the

attraction point instead of waiting. For the TDMCTOPTW, the calculation time for evaluation is very insufficient to verify because of the inclusion of public transportation network. Thus we make local evaluation of possible insertion, only including the attraction point that is inserted and two attraction points between which new attraction point is inserted. This way is illustrated based on the method proposed in (P.Vansteenwegen, W.Souffriau, G,V.Berghe, D.V.Oudhesden, 2009) and (A.Garcia, P.Vansteenwegen, o.Arbelaitz, W.Souffriau, M.T.Linaza, 2013). The pseudo code of TDMCTOPTW algorithm is shown as follows:

```
PositionStartRemove=1;
NumberToRemove=1;
NumberOfTimesNoImprovement=0;
maxNumberToRemove=NumberOfPOIs/(3*NumberOfDays);
maxIter=factorNoImprovement * SizeOfFirstTour;
While NumberOfTimesNoImprovement < MaxIter do
While not local optimum do
        For each non included visit;
            Determine the best possible insert position and Shift;
 Calculate Ratio;
          Insert visit with highest ratio;
If Solution better than BestFound then
       BestFound=Solution;
       NumberToRemove=1;
       NumberOfTimesNoImprovement=0;
else
       NumberOfTimesNoImprovement+1;
Shake Solution (NumberToRemove,PositionStartRemove);
PositionStartRemove=PositionStartRemove+NumberToRemove;
NumberToRemove+1;
PositionStartRemove >= Size of smallest Tour;
If NumberToRemove=maxNumberToRemove then
```

```
NumberToRemove=1;
Return BestFound;
For dt =0 to numberOfTravelingDays
        For each included Visits[d] in BestFound
                if WalkDistanceToBusStop < WalkDistanceToNextVisitPlace then
                        find the nearest bus stop
                        find the nbs near to the next visiting place;
                        if next visiting place shares the bus stop then
                                walk to the next visiting place;
                        else
                                get list of busses that stops  at nbs
                                select theshortest bus line to the nbs
                                take off the bus and walk to the next visiting place
                else
                        walk to the next visiting place;
nbs – next bus stop
```

Listing 4.2: ILS algorithm for the TDMCTOPTW

The detail programming code of the ILS algorithm for TDMCTOPTW is listed on the Appendix C.

4.3.1 TDMCTOPTW- main concept

As we discussed before we need very fast and efficient local evaluation of each possible insertion. In order to adopt the ILS method to TDMCTOPTW algorithm there is a need to reconsider the Ratio function.

We cannot use the same ratio function to the TDMCTOPTW due to the insufficiency of the previous ratio function which was used to calculate comparison of each point

that can be inserted to the tour. The previous ratio function only took into account the score of the attraction point and the visiting time required to visit that particular point. But we have to make some changes into the ratio function in the case of including several attributed constraints such as money budget and public transportation. Thus, we decided to add a special weight to each attribute constraint and include the available quantity of each constraint on the tour.

$$ratio = \frac{S_i^2}{\frac{shift_i}{Time_{available}} + \sum_{k=1}^{l} \frac{1}{K} \frac{f_{ik}}{available_k}} \tag{4.5}$$

From the formulation (4.5) we can see that the weight of other attribute constraints (like money budget) are also important as well as time budget constraint and the more additional constraints will not raise the total weight of the denominator.

4.3.2 Example scenario

Based on the abovementioned concepts we can apply them into example scenario. We use “bus” as a public transportation option and “walk” as a second option.

Walking or Taking bus

This scenario depends on the exact leaving time of the attraction point i what the fastest option will be walking or taking the bus. The option one “walk” will be chosen when there is long waiting time for the bus. Thus second option “taking bus” will be chosen when the waiting time is short enough. Based on this, the traveling time between two attraction points equals to the walking time at most or equals to the traveling time of bus at least.

If the tourists wait for the bus, they could leave later and still arrive at the same time. If they leave too late they will miss the bus and they will walk.

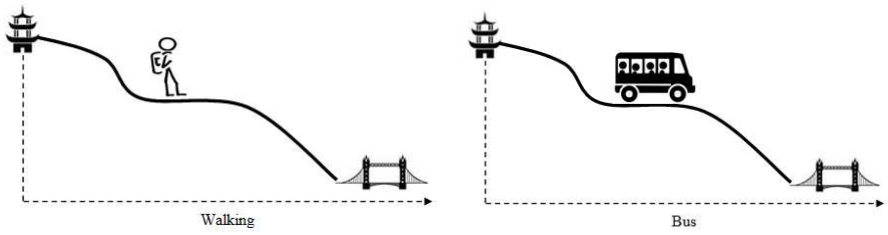


Figure 4.1: Walking or Taking bus scenario

In order to explain the algorithm using the example scenario, we make the following example.

Table 4.1 Example Scenario

Tourist Information	Name: Johnny Cooper
	Nationality: USA
	Age:31
Personal Interest	Love to visit Buddhist monasteries.
	Also like to visit national history museums.
	Interested in statues and monuments.
	Like to walk by enjoying street life.
Travel duration in UB city	Only one day
Travel estimated budget	200$ excluding accommodation cost
Accommodation	Chinggis Khaan Hotel (starting point and ending point)

We import tourist's data into to machine and the TDMCTOPTW algorithm calculate very fast. Our algorithm solves the problem in 3.6 seconds which is very good result compared to the existing methods for similar problems. The algorithm can be modified easily to solve these small instances efficiently.

The result of the TDMCTOPTW algorithm for above mentioned example is shown as below:

1 visit: 8:: Wedding Palace

2 visit: 20:: Statue of Natsagdorj

3 visit: 24:: Statue of Political persecution victims

4 visit: 26:: Statue of Marshal Choibalsan

5 visit: 7:: Chinggis Khaan Square

6 visit: 27:: Statue of National Seal

7 visit: 29:: Peace Bridge

8 visit: 2:: Museum of Choijin Lama

****

5 dugaariin zogsooloos: Chinggis Square

6 dugaariin zogsool ruu: Yonsei Hospital

Available bus lines :

Found it!: M1, 5, 6

Found it!: M2, 5, 6

Found it!: M3, 5, 6

Found it!: M7, 5, 17, 0, 0, 0

Found it!: M29, 5, 6

Found it!: M34, 5, 17, 18, 0, 0, 0

Found it!: M37, 5, 6

Found it!: M50, 5, 6

Found it!: M55, 6, 18, 16, 5

Found it!: T2, 5, 6

Found it!: T4, 5, 6

Found it!: T5, 5, 6

Take busline : T5

6 dugaariin zogsooloos: Yonsei Hospital

18 dugaariin zogsool ruu: Ard Cinema

Available bus lines :

Found it!: M8, 18, 16, 15, 19, 20, 21, 22, 0, 0

Found it!: M29, 18, 5, 6

Found it!: M34, 18, 0, 0, 0

Found it!: M48, 6, 18

Found it!: M53, 6, 17, 18

Found it!: M55, 6, 18

Take busline: M55

18 dugaariin zogsooloos: Ard Cinema

17 dugaariin zogsool ruu: NUM

Available buslines:

Found it!: M29, 17, 18

Found it!: M30, 17, 18

Found it!: M34, 17, 18

Found it!: M48, 17, 6, 18

Found it!: M53, 17, 18

Take busline: M53

17 dugaariin zogsooloos: NUM

5 dugaariin zogsool ruu: Chinggis Square

Available buslines:

Found it!: M7, 5, 17

Found it!: M29, 17, 18, 5

Found it!: M30, 5, 6, 17

Found it!: M34, 5, 17

Take busline: M34

5 dugaariin zogsooloos: Chinggis Square

17 dugaariin zogsool ruu: NUM

Available buslines:

Found it!: M7, 5, 17

Found it!: M29, 17, 18, 5

Found it!: M30, 5, 6, 17

Found it!: M34, 5, 17

Take busline: M34

17 dugaariin zogsooloos: NUM

16 dugaariin zogsool ruu: National Library

Available bus lines:

Found it!: M7, 16, 5, 17

Found it!: M34, 16, 5, 17

Found it!: M53, 16, 6, 17

Take busline: M53

16 dugaariin zogsooloos: National Library

5 dugaariin zogsool ruu: Chinggis Square

Available buslines:

Found it!: M7, 16, 5

Found it!: M34, 16, 5

Found it!: M50, 16, 17, 5

Found it!: M55, 16, 5

Take busline: M55

Figure 2 Screenshot of map "Example scenario"

4.4 Summary

In this chapter we present the meta-heuristic approach to deal with the time dependent multi constraint team orienteering problem with time windows. Several efficient techniques to tackle the previous extensions of this optimization problem are mentioned in Chapter 2 and Chapter 3. However we select the iterated local search algorithm to solve the problem since it has been shown the best result and run on a mobile device with limited computational resources.The local search heuristic is introduced at the beginning of the chapter. In doing so, we adopt the iterated local search meta-heuristic to deal with the TDMCTOPTW problem. Furthermore, we adopt the classic ILS method to our algorithm by making some changes in calculation of Ratio function.

Chapter 5

Experimental Validation and Prototype Implementation

Content

Abstract. In this experimental validation chapter, we applied the ILS algorithm to the real life data set of Ulaanbaatar city (UB) of Mongolia including 35 attraction points. The computational validation results are shown in section 5.2. Therefore, we implemented the new mobile tourist tour planning recommendation system so called UBTour Planner

(UBTP). The UBTP aims to help tourists to plan their travel in UB taking into account the tourist interest, preferences, budget and time to spend. The one of the main features of UBTP is integration of public transportation into the system. The section 5.3 introduces implementation of the tour planning recommendation system.

5.1 Experiment Setup

5.1.1 Satisfaction score estimation

In order to model and implement the TDMCTOPTW, the satisfaction score of every attraction points have to be defined. We made a survey "Tourist Interest and Satisfaction" and we are able to evaluate the satisfaction score based on the result of questionnaires. This survey is developed and distributed during the peak season of July and August of 2014 to determine the satisfaction levels of international tourists in Mongolia. The self-administered survey includes 6 attraction and public transportation related questions. All listed attraction points are situated in Ulaanbaatar city were measured on a five-point scale ranging from 1 to 5. Obviously, tourists' rate 1 for very dissatisfied, 2 for dissatisfied, 3 for Average, 4 for Satisfied and 5 for highly satisfied. The survey was written in English

5.1.2 Data collection procedure

Since Mongolia has only one international airport we collected the data at the Chinggis Khaan International Airport of Ulaanbaatar city. We assume that all airline passengers use this airport. The survey was conducted in the departure lounge from the beginning of July to end of August in 2014. Totally, 800 questionnaires are collected and 120 were incomplete. Questionnaire consisted of two sections: first section includes 6 questions related to general information about trip while second

section requested to indicate the satisfaction on main attraction points of UB city. See Appendix B for sample questionnaire.

5.1.3 Survey result

As the result of the survey, it was interesting to note that 66% of the international tourists had information about Mongolia from Internet. Therefore, 85% of the participated tourists stayed in UB city for 2 days and 15% of them stayed 3 days. Most of them were interested in nature and nomad life of Mongolia. Also, we can see from the survey that tourists preferred to use public transportation in the capital city.
Tourists asked to evaluate the attraction points inside the UB city. We choose 35 main tourist attraction points and tourists gave the score on every point on a 5-point scale ranging from 1(least satisfied) to 5 (highly satisfied). In the table 4.1, we summarize the average satisfaction score of 35 attraction points in UB based on the survey. The average satisfaction score is defined by dividing total given satisfaction scores by the number of total participants. In the table 4.1, satisfaction score of 35 attraction points are listed which are resulted from the survey.

Table 5.1: Satisfaction score of attraction points

	Category	**Name**	**Average Satisfaction score**
1	Palace& Center	Wedding Palace	3
2		National wrestling palace	3
3		Palace of Sport	2
4		Center of Culture	4
5		Parliament Palace	5
6		Palace of Children	4
8	Monastery	Geser Monastery	4
9		Gandan Monastery	5
10		Zaisan Hill	5
11	Statue	Statue of Political persecution victims	5
12		Statue of Sukhbaatar	5
13		Statue of Natsagdorj	3

14		Statue of Soviets soldiers	2
15		Statue of Marshal Choibalsan	3
16		Statue of National Seal	4
17	Bridge	Lion Bridge	3
18		Bridge of Tuul river	3
19		Peace Bridge	5
20	Park & Station& Square & Store	Buddha's park	5
21		Ulaanbaatar station	4
22		Chinggis Khaan's Square	5
23		State department store	5
24	Theatre& Circus	National Circus	5
25		National Theatre	4
26		National Theatre of Opera	5
27		Tsagaan darium art museum gallery	5
28		Xanadu Art Gallery	4
29	Museum	Museum of Choijin Lama	5
30		International Intellectual Museum	4
31		Winter palace of Bogd Khan	5
32		Fine art museum of Zanabazar	5
33		National museum of Mongolia	5
34		Mongolian National Costume Museum	4
35		Museum of National History	5

5.2 Test Set

The proposed algorithm and formulation are evaluated by conducting some experiments using data from the Ulaanbaatar, capital city of Mongolia. The Ulaanbaatar city is the biggest city in Mongolia, including 9 big districts and employed three kinds of public transportation modes: taxi, bus and trolleybus. Also, Ulaanbaatar city is famous for its interesting tourist attractions. The number of main tourist attraction points is more than fifty. In order to make our experiment we create a dataset including 35main tourist point of interest. The dataset is classified into main 10 categories namely Palace, University, Square, Statue, Museum, Theatre, Petrography, Mountain, Bridge and Monastery. All data prepared as needed and

tables are made including associated information such as coordination location, time needed to visit, entrance fee, related satisfaction score and timetable of attractions. Additionally, the associated satisfaction score is determined from the result of "tourist satisfaction and interest" survey which includes around 800 international tourists who have just finished their travel in Mongolia.

Table 5.2: Real life data set of Ulaanbaatar city

	Category	Name	Location		Visiting duration /di, min/	Entrance fee /$/	Opening hour /Oi/	Closing hour /Ci/	Satis-faction core /Si/
			Xi	Yi					
1	Palace & Center	Wedding Palace	47.914358	06.920015	120	0	180	660	5
2		National wrestling palace	47.91787	106.93414	30	10	120	840	3
3		Palace of Sport	47.92024	106.92396	120	0	60	750	2
4		Center of Culture	47.91954	106.91932	120	10	120	850	4
5		Parliament Palace	47.92065	106.91795	120	0	120	620	5
6		Palace of Children	47.91306	106.9157	120	0	180	600	5
7	Monastery & Temple	Geser Monastery	47.92679	106.89453	100	5	180	600	4
8		Gandan Monastery	47.921	106.905	90	10	0	960	5
9	Mountain & Hill	Mountain Bogd Khan	47.80389	106.98639	120	15	0	960	5
10		Zaisan Hill	47.884	106.915	120	5	0	960	5
11		Statue of Political persecution victims	47.92043	106.91533	30	0	0	960	5
12		Statue of Sukhbaatar	47.91822	106.91708	30	0	0	960	2
13		Statue of Natsagdorj	47.91784	106.92305	30	0	0	960	4
14		Statue of Soviets soldiers	47.91797	106.95093	40	0	0	960	5
15		Statue of Marshal Choibalsan	47.92172	106.91864	100	0	0	840	3
16		Statue of National Seal	47.92154	106.91722	30	0	0	900	5
17	Bridge	Lion Bridge	47.91844	106.93028	20	0	0	960	2
18		Bridge of Tuul river	47.89007	106.90986	90	0	0	960	3
19		Peace Bridge	47.90826	106.91312	30	0	0	960	4
20	Park & Station& Square & Store	Buddha's park	47.88619	106.91225	90	0	120	840	5
21		Chinggis Khaan's Square	47.91822	106.91708	100	0	0	960	5
22		State department store	47.91542	106.90614	150	0	180	840	5
23		National Circus	47.91202	106.9072	120	15	480	780	5

24	Theatre& Circus	National Theatre	47.91457	106.91457	90	12	480	850	5
25		National Theatre of Opera	47.91851	106.91928	90	10	660	900	5
26		sagaan darium art museum gallery	47.88868	106.91135	90	20	240	700	4
27		Xanadu Art Gallery	47.91917	106.90983	90	10	180	720	4
28	Museum	Museum of Choijin Lama	47.915	106.91897	110	15	120	620	5
29		International Intellectual Museum	47.91743	106.9417	120	10	120	720	5
30		Winter palace of Bogd Khan	47.89742	106.90703	150	15	240	600	5
31		Fine art museum of Zanabazar	47.92039	106.90956	120	10	120	660	5
32		National museum of Mongolia	47.92261	106.91464	80	5	120	540	5
33		Mongolian National Costume Museum	47.91637	106.92004	120	20	180	600	5
34		Museum of National History	47.92099	106.91519	90	15	180	540	5
35		Ulaanbaatar station	47.90832	106.88502	120	0	0	960	3

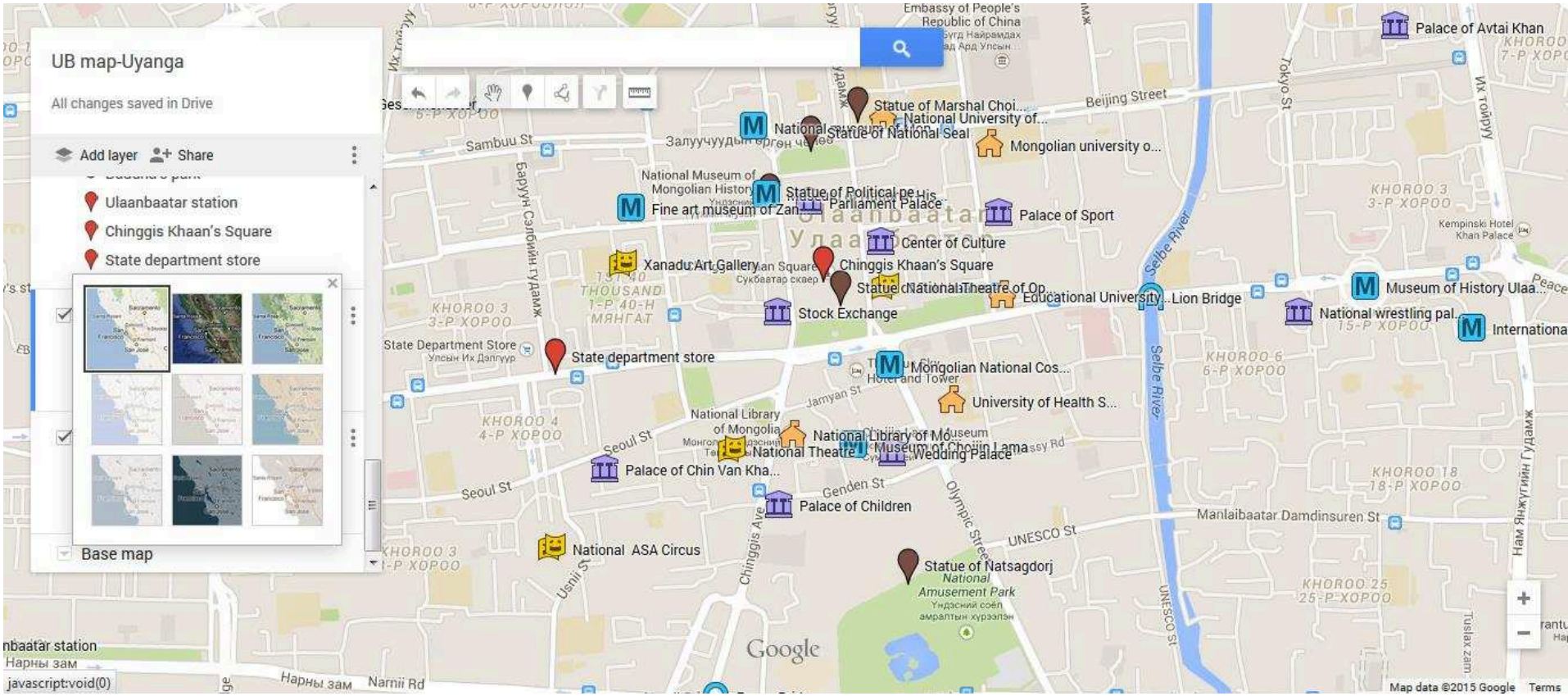


Figure 5.1. Location of Point of Interests in Ulaanbaatar city (Google map)

5.2.1 Computational result

In order to analyze the performance of our algorithm, we made our own data set based on the real data of Ulaanbaatar city which is presented on previous section. As we mentioned before, the ILS heuristic was shown very good results with Solomon's data set and Cordeau's data set. Since the TDMCTOPTW problem was more challenging to solve by its additional attribute constraints we adopt the ILS algorithm by changing some function.

In the table 5.3, computational result of the experiment is presented. We run the test with 1 or 2 days tour and different starting point.

Table 5.3: Computational experiment results of the TDMCTOPTW

M	**CPU time (sec)**	**Number of attraction Point**	**Starting Point /hotel/**	**Location**	**M**	**CPU time (sec)**
1	3.5	35	Coco	47.918442, 106.862774	2	4
1	5.5	35	Amure	47.918363, 106.883385	2	5.5
1	4	35	Grand Hill	47.918269, 106.884189	2	5.5
1	4	35	White House	47.918485, 106.889994	2	3
1	3.5	35	Narantuul	47.915709, 106.895873	2	2.5
1	2.8	30	Khongor Guest House	47.916227, 106.903619	2	3.5
1	2	30	Mongolian Steppe	47.918183, 106.906301	2	6
1	2.2	30	Bishrelt	47.923561, 106.907149	2	3.3
1	1	30	Sun Path	47.924582, 106.912825	2	4.4
1	1.5	30	Blue Sky	47.916184, 106.918741	2	3.5
1	3.7	35	Lotus Guest House	47.92944, 106.912256	2	5.5
1	3.5	35	UB City	47.901089, 106.902198	2	5.6
1	2.9	35	Khilchin	47.932648, 106.923392	2	1.5
1	3.2	35	Peace Bridge	47.905185, 106.910266	2	4.3
1	2.8	35	Ramada	47.915954, 106.893164	2	3.5
1	1.5	30	Best Western	47.910445, 106.875097	2	4.3

1	2.9	30	New West	47.913955, 106.862179	2	3.8
1	3.1	35	Chinggis Khaan	47.921965, 106.933998	2	3
1	2	35	Kempinski	47.919189, 106.944383	2	4
1	3.1	35	Continental	47.912617, 106.924857	2	4.5
1	2.8	35	Corporate	47.913013, 106.914241	2	4
1	3	35	Flower	47.924984, 106.938139	2	5
1	1	30	Zaluuchuud	47.924510, 106.922008	2	3.5
1	1	30	Bayangol	47.912315, 106.913940	2	2.7
1	2	35	Khunnu Palace	47.930053, 106.931643	2	4.5
1	3.5	35	Narlag Hotel	47.930556, 106.918371	2	3
1	1.5	35	Ulaanbaatar	47.948736, 106.922856	2	3
1	2	35	Topaz	47.932612, 106.923189	2	1.8
1	5.5	35	Plantinium	47.916565, 106.926793	2	6.8
1	2	35	New World	47.931850, 106.924594	2	5.5
1	3	30	Kaiser	47.916436, 106.927158	2	5.5
1	4.1	30	Springs	47.915522, 106.921472	2	5.8
1	2.3	30	Edelweiss	47.915163, 106.928467	2	3.6
1	3	30	Selenge	47.923302, 106.948718	2	6.3
1	1.5	30	Park	47.921433, 106.952924	2	2.7

The results for all new data instances are summarized in table 5.3. Totally, 35 different hotels are tested as a starting point and experiment made by one day tour and two days tour with 35 attraction points. In some case, we tested only 30 attraction points in order to check the performance of the algorithm.

All computations were made on a laptop computer HP 12nr with AMD A6-3420M APU processor and 4 GB Ram. Our algorithm solves the problem with an average computation time of only 3 seconds with one day tour and an average computation time of 5 seconds with 2 days tour. The performance of the ILS shows that algorithm can easily be applied and adopted for problems with more extra constraints due to its simplicity.

5.3 Implementation of the UB TOUR PLANNER

This chapter presents the UB TOUR PLANNER, a mobile tourist tour recommendation system. This recommendation system aims to give a suggestion of city tours based on the user's personal interests, preferences and constraints. The UB TOUR PLANNER (UBTP) considers one or multiple day's tour in Ulaanbaatar. It integrates the selection of attraction points and the ordering, routing, scheduling these points.

Ulaanbaatar is the capital and the largest city of Mongolia. It might be the first stop of international tourists who want to travel in Mongolia. An independent municipality, the city is not part of any province, and its population as of 2014 is over 1.3 million.[8]The city is located in north central Mongolia, the city lies at an elevation of about 1,310 meters (4,300 ft) in a valley on the Tuul river. It is the cultural, industrial, and financial heart of the country. It is the center of Mongolia's road network, and is connected by rail to both the Trans-Siberian Railway in Russia and the Chinese railway system. Tourism is becoming increasingly important for the Mongolian economy as the demand for tourism in Mongolia is increasing every year. Also, recent depreciation of MNT (tugriks) makes tourism products cheaper and more attractive. Mongolia's travel and tourism sector accounts for 9 % of Mongolia's GDP. However, the number of the tourists visited Mongolia was only 460,000 in 2011 and it has been increasing since after.

[8] http://en.wikipedia.org/wiki/Ulan_Bator

Figure 5.2:Map of Mongolia

The UB city is divided into six major districts, but there's a multitude of sub-districts and micro-districts. Mongolians rarely use the Western system of street names and numbers, so tracking down an address place can be difficult. Thus, there is huge need to have a recommendation system which can provide useful information including tour plan for the tourist. The UB Tour Planner can help to user to select the attractions and planning their city tour. In the following sections the components of the UBTP is organized.

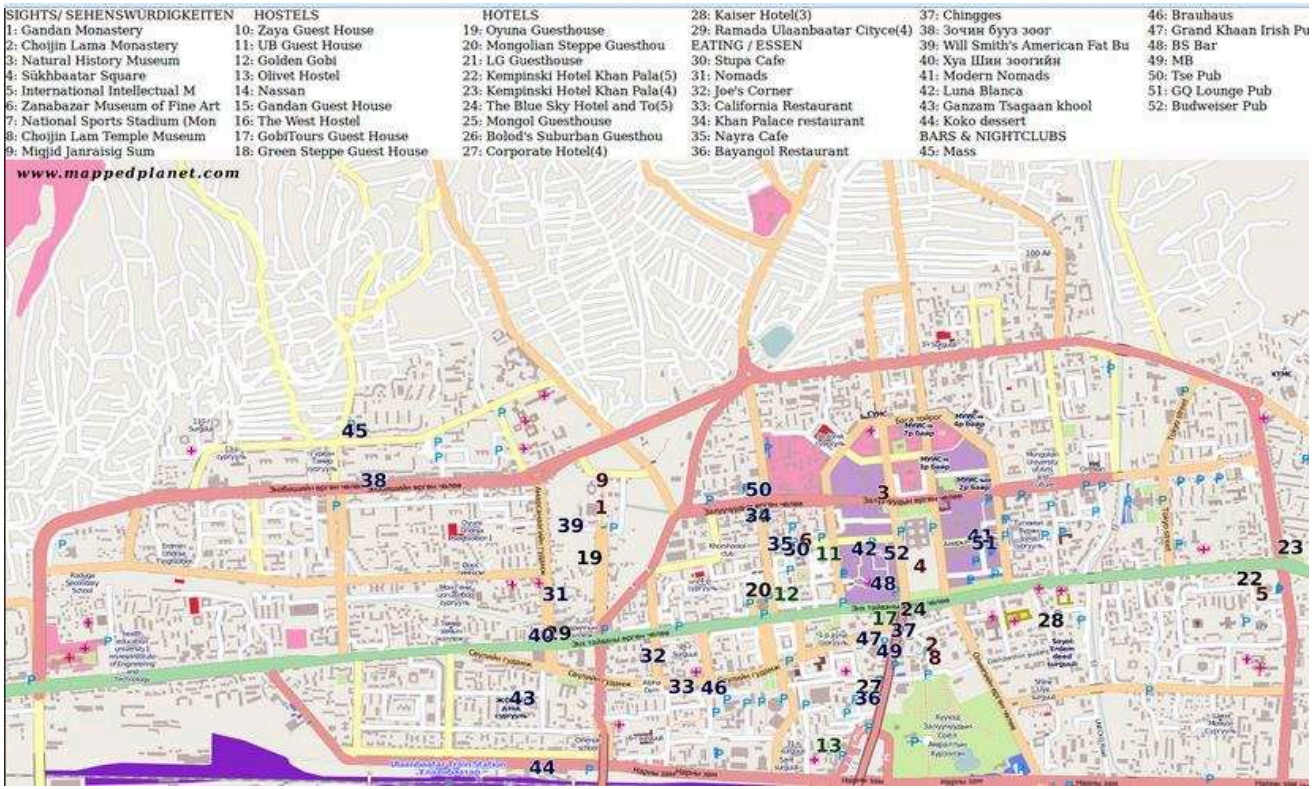


Figure 5.3:Tourist map of Ulaanbaatar city

5.3.1 System Architecture

The UB TOUR PLANNER includes 35 main attraction points of the Ulaanbaatar city and its related information such as location of the point, timetable, entrance fee and popularity of the point. In order to plan tours, the one of the hardest extension of the Orienteering problem is used to model the planning problem namely Time Dependent Multi Constraint Team Orienteering Problem with Time Windows.

The TDMCTOPTW takes into account certain hard constraints besides time windows. Each point is extended with extra attribute constraint like money budget or max-n types of attraction and also its opening and closing timetable. In the figure below, the system architecture of the UB TOUR PLANNER is shown. Main function of this system is mobile tourist provide his own preference of trip into the server to calculate optimal tour planning. Mobile tourist need to introduce his total trip budget, total time, preference for attractions (POIs), other specific requests to his device in order to receive back as convenient as possible tour itinerary plan. According to these requirements we can give following framework for optimal tourist tour planning system (Figure 5.4).

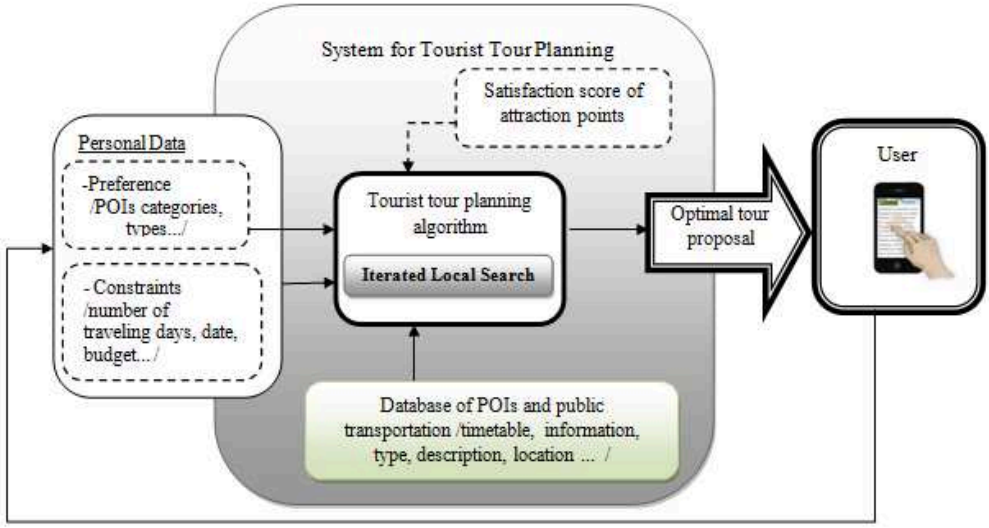


Figure 5.4:System architecture of the UB TOUR PLANNER

Before calculating tourist's preferred city tour, the UB TOUR PLANNER estimates tourist's interest and preference based on tourist satisfaction survey. At this stage tourist can express his interest's level on the each types of attractions. Then information about each attraction points (opening, closing hours, location – GPS coordination, entrance fee), the tourist's personal preference scores and other related information transferred to the tour planning algorithm in order to calculate specific tour suggestion. Obviously, right after the calculation of tour planning algorithm, the system provide the tour tailored to the tourist's preferences, location, available time and destination.

5.3.2 Database Input and User Input

Database input includes road map on Google map, urban public transportation lines, bus stops and data of destinations. Road map is given as a directional graph. Each edge is assigned a length. The shortest distance between particular two points is fixed. Bus lines are fixed as well as all bus stops locations.

In UB city, public transportation service is well developed with fixed timetable. The public bus network consists overall 66bus lines thereof 40 lines serve in the city center where the most of the attraction points are located. Thus tourist can use direct bus to the attraction points without making transfers. Furthermore, there are 610 bus stops in UB city thereof approximately 50are situated in the tourist main street “Peace Avenue”.

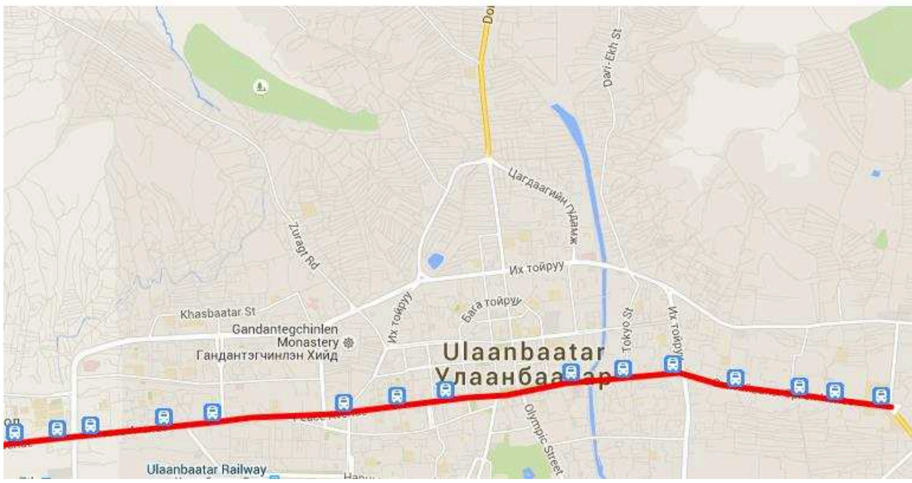


Figure 5.5:Location of main bus stops at Peace Avenue

The length of the Peace Avenue is 8.9km. Approximately 300 buses travel through the Peace Avenue per day. According to the survey, the frequency of buses is around 5-8 minutes. So far, 5 buses stop per a minute at the one bus stop(Kh.Bulga, 2015).
See Appendix B.2 for location of the bus stops and overlapping public bus network.

As user input, tourist introduces personal data into the system including:

- Date of arrival at UB city
- Number of days to stay at UB
- Rate of tourist's travel interest
- Starting and ending location of the each day tour
- Starting and ending time of the each day tour

Based on this information, system can calculate near optimal tours including the set of attraction points.

The UB TOUR PLANNER recommendation system contains attraction point's information of 35 main points of interests in Ulaanbaatar city. Each point of interest is included its database containing:

- Location – GPS coordination
- Timetable – Opening and Closing hours
- An average duration to visit particular point of interest
- Type of the POI.
- Satisfaction scores
- Entrance fee

5.3.3 User Interface

In the UBTP, the TDMCTOPTW algorithm is applied in order to tackle the planning problem. This problem includes set of locations, each of them with a certain satisfaction score, a time window and one or more associated attributes, such as an entrance fee. Each attribute type has an associated constraint with a maximum allowed value for a route, such as a limited budget. Visiting a location within its time window allows collecting its score as a reward. The problem is time dependent because of the public transportation lines and bus stop. The timetable of bus lines makes this problem very challenging. The goal is to determine routes that maximize

the collected score without violating any of the constraints. The starting point 1 and ending point N of every tour are fixed. Traveling time between poi i and j is known for all points. See chapter 3 section 3.2.2.

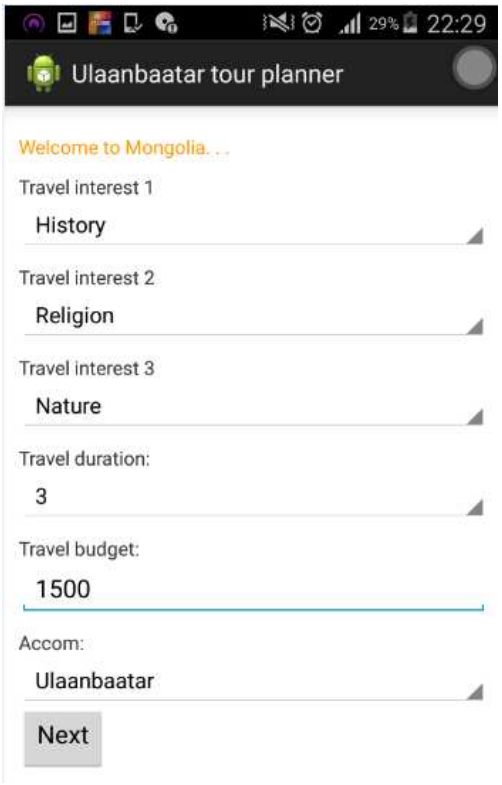


Figure 5.6: Screenshot of Ulaanbaatar Tour Planner (UBTP) User Interface (on Android system)

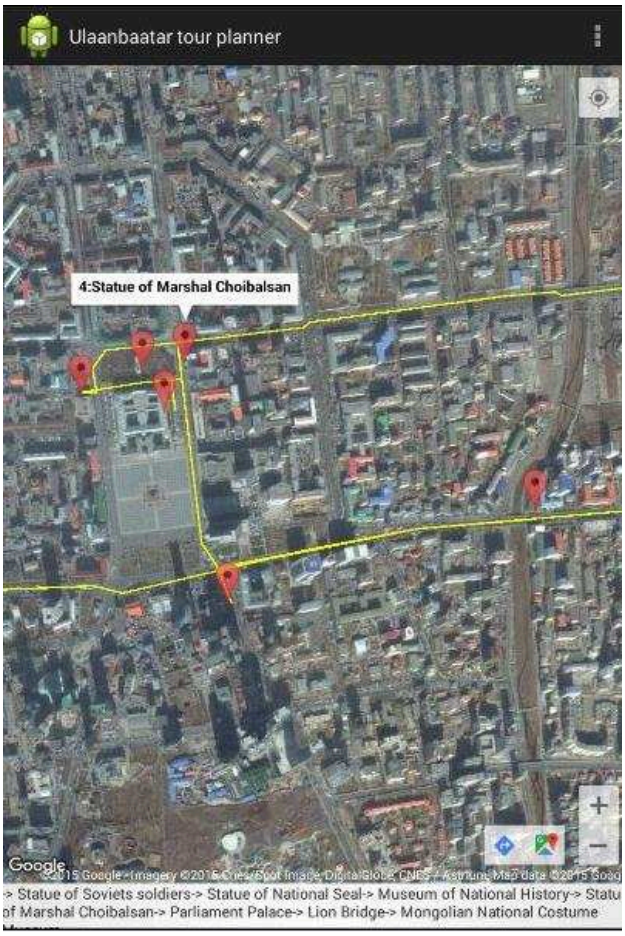
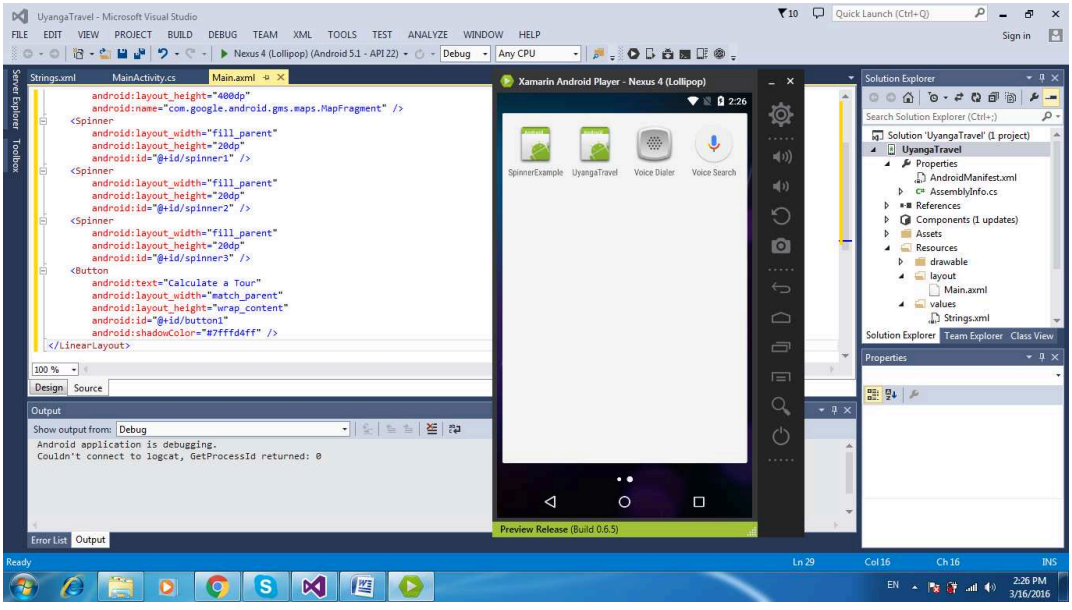


Figure 5.7: Screenshot of map result of UBTP



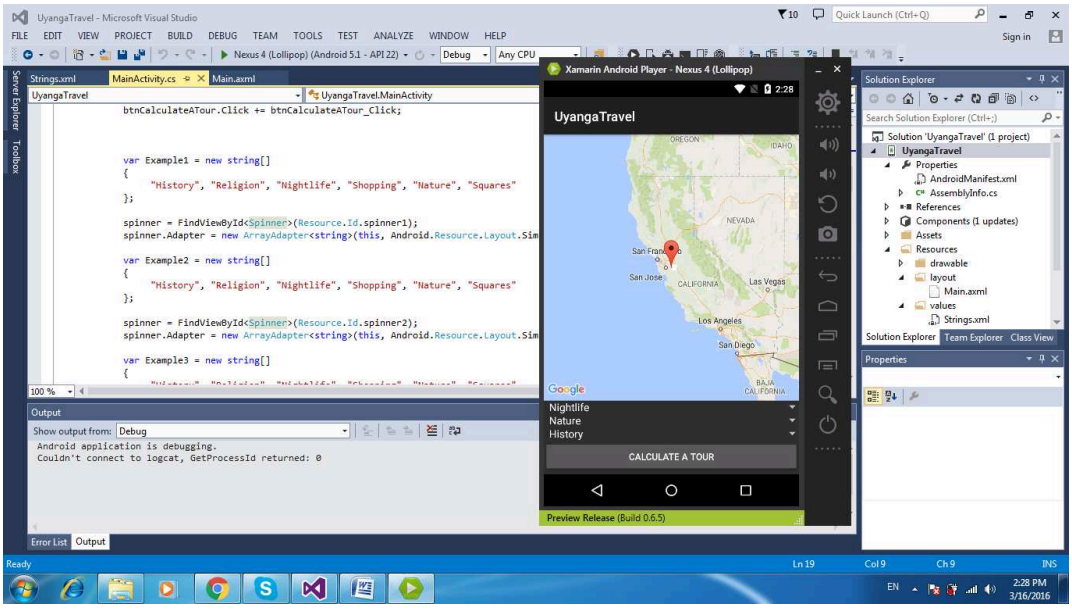


Figure 5.8 Screenshot of UBTP tour planner

5.4 Summary

In this chapter we generated a new test set in order to make the experimental validation of the problem and we applied the meta-heuristic to the TDMCTOPTW. Therefore, we presented the estimation of the satisfaction score and test the data set that we collected from the survey. Our algorithm solves the problem with an average computation time of only 3 seconds with one day tour and an average computation time of 5 seconds with 2 days tour. The performance of the ILS shows that algorithm can easily be applied and adopted for problems with more extra constraints due to its simplicity.

Then, the mobile tourist tour planning recommendation system, named UB Tour Planner, is introduced to demonstrate the applicability of the proposed model and ILS algorithm. The mobile application allows planning one or two day plans in UB city considering multiple constraints and time dependent public bus integration.

Chapter 6

Conclusion

In this thesis, we investigated the well-known combinatorial optimization problem of Operations Research and the efficient heuristic techniques to deal with the problem. We aim to implement tourist tour planning problem including multiple tasks and functionalities. In order to develop the tourist tour planning recommendation system, we start by emphasizing the basic Orienteering Problem and its extended versions. The model of the Orienteering Problem has the all requirements of the tourist tour planning functionalities. We summarized all versions of the problem which are extended with multiple day tours, time windows, multiple constraints and time dependent constraint. Based on the literature review, we focus on the multi constraint problem which has not integration of the time dependency constraint.

Also, number of implemented heuristic approaches is introduced in order to achieve near-optimal solution quality within limited calculation times. Several efficient methods like Tabu Search, Genetic Algorithm and Ant Colony Optimization techniques have been shown good solutions. But the Iterated Local Search meta-heuristic has proven to be best approach to deal with the multi constraint problem. Thus we adopt the iterated local search method by adding several changes since the time dependent problem with multi constraint has extra hard constraints.

Time dependency is mostly used when the problem includes the use of public transportation besides other constraint like entrance fee, timetable of attraction points and so on. This is the one of the hardest extension of the Orienteering Problem, where the time needed to travel between attraction point 1 to attraction point 2 depends on the leaving time from attraction point 1. However, in our knowledge none of the previous papers summarized on state-of the-art are shown an algorithm which can tackle the problem includes time windows, multiple day tours, multiple constraints and use of public bus in same time. One of the key contributions of this thesis is combining all these constraints into the one problem and proposes the efficient technique to handle.

The model we propose is the time dependent multi constraint team orienteering problem which combines the previous models and makes it possible to add public transportation networks to travel between attraction points. As aforementioned, we tried to adopt the iterated local search meta-heuristic to deal with our model which has already shown the good solution with TOPTW problem. In the proposed ILS algorithm, we made some changes in calculation ratio function. In the previous method, researchers calculate the ratio function depends on satisfaction score and required time to visit attraction point. But our problem is more complicated to solve because of its additional attribute constraints. Thus we add more weight related to the extra constraints in the formulation. In order to check the algorithm, we created an example scenario to select one of the transportation modes between walking or taking public bus. We have tested the algorithm with new generated test set of Ulaanbaatar including 35 main tourist attraction points and large public transportation network. Our algorithm solves the problem with an average computation time of only 3 seconds with one day tour and an average computation time of 5 seconds with 2 days tour. The performance of the ILS shows that algorithm can easily be applied and adopted for problems with more extra constraints due to its simplicity.

In order to see the applicability of the model and algorithm we develop a mobile application for tourist tour planning which able to create tourist tour plans by considering mobile users interest and preferences. This tourist tour planning recommendation system targeted the international tourists in Ulaanbaatar, Mongolia

Appendix A

A.1 Benchmark test set for the Multi Constraint Team Orienteering Problem with Time Windows

Table .A.1New test set of MCTOPTW by(A.Garcia, P.Vansteenwegen, Wouter Souffriau, Olatz Arbelaitz , Maria Teresa Liaza, 2010)

umber of points	Coordination		Number of tours m	Total time Ti	Score of VisitSi	Timetable of point		Number of Attribute constraint	
						Oi	Ci	e_{i1}	e_{i2}
0	40	50	1	-	-	0	1236		
1	45	68		90	10	912	967	1	5
2	45	70		90	30	825	870	2	5
3	42	66		90	10	65	146	3	5
4	42	68		90	10	727	782	4	5
5	42	65		90	10	15	67	5	5
6	40	69		90	20	621	702	6	10
7	40	66		90	20	170	225	7	10
8	38	68		90	20	255	324	8	10
9	38	70		90	10	534	605	9	10
10	35	66	1	90	10	357	410	10	10
11	35	69		90	10	448	505	11	15
12	25	85		90	20	652	721	12	15
13	22	75		90	30	30	92	13	15
14	22	85		90	10	567	620	14	15
15	20	80		90	40	384	429	15	15
16	20	85		90	40	475	528	16	5
17	18	75		90	20	99	148	17	5
18	15	75		90	20	179	254	18	5
19	15	80		90	10	278	345	19	5

Appendix B

B.1 Sample of Questionnaire

Tourist Interest and Satisfaction Survey

Dear Sir / Madam!

First of all, thank you very much for your stay in Mongolia. We hope that your trip was nice as you expected. We kindly ask you to participate in a survey which will help us to improve the service in tourism and particularly the implementation work of Ph.D dissertation of Ms.Uyanga SUKHBAATAR who is a Ph.D candidate of University of Grenoble, France. The research work aims to develop tourist tour planning recommendation system by indicating the satisfaction score of the attraction points in Ulaanbaatar and proposing new algorithm to implement the UB Tour Planner, a new mobile application to help tourists.

Thank you for your time. Hope we will see you again in Mongolia.

Please circle your answers.

Table B.1 Information table

	Questions	nswers
	Nationality	
	Have you visited Mongolia before?	Yes No
	How many days did you stay in Ulaanbaatar city?	1 day 3. 3 days 2days 4. 4 days
	What is your information source about Mongolia?	TV Internet Guidebooks/magazines Tour Brochures
	What is your travel interest and goal?	Nature/Adventure Religion/History Nomad life Business Others
	What kind of transportation mode did you use in UB city?	Public bus Taxi Walk

Please rate your satisfaction score on your visited attraction points in UB city.

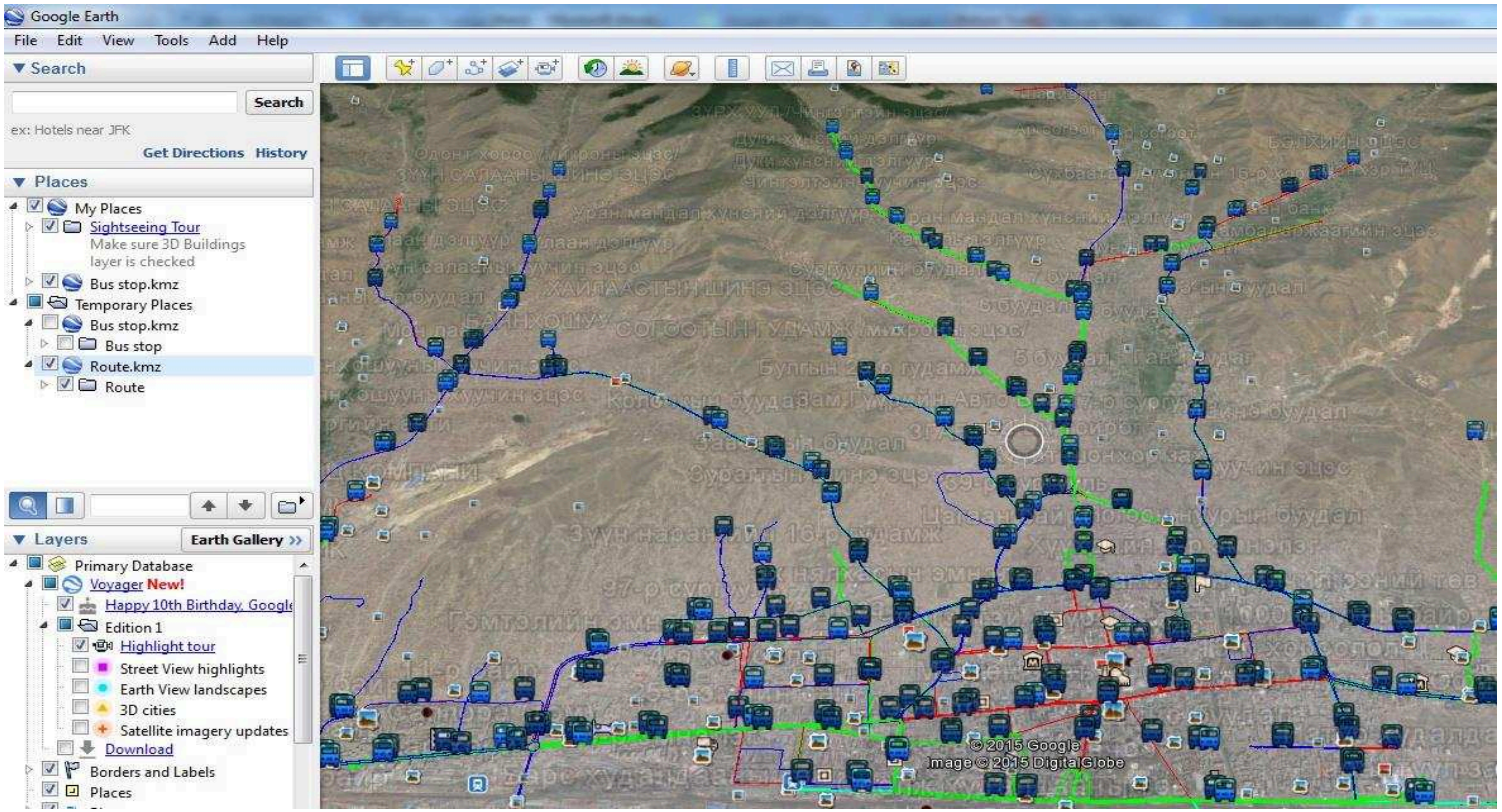
Table B.2 Survey table

	Category	**Name**	**Ratings** 1 dissatisfied to 5 highly satisfied/				
1	Palace& Center	Wedding Palace	1	2	3	4	5
2		National wrestling palace	1	2	3	4	5
3		Palace of Sport	1	2	3	4	5
4		Center of Culture	1	2	3	4	5
5		Parliament Palace	1	2	3	4	5
6		Palace of Children	1	2	3	4	5
8	Monastery	Geser Monastery	1	2	3	4	5
9		Gandan Monastery	1	2	3	4	5
10		Zaisan Hill	1	2	3	4	5
11	Statue	Statue of Political persecution victims	1	2	3	4	5
12		Statue of Sukhbaatar	1	2	3	4	5
13		Statue of Natsagdorj	1	2	3	4	5
14		Statue of Soviets soldiers	1	2	3	4	5
15		Statue of Marshal Choibalsan	1	2	3	4	5
16		Statue of National Seal	1	2	3	4	5
17	Bridge	Lion Bridge	1	2	3	4	5
18		Bridge of Tuul river	1	2	3	4	5
19		Peace Bridge	1	2	3	4	5
20	Park & Station& Square & Store	Buddha's park	1	2	3	4	5
21		Ulaanbaatar station	1	2	3	4	5
22		Chinggis Khaan's Square	1	2	3	4	5
23		State department store	1	2	3	4	5
24	Theatre& Circus	National Circus	1	2	3	4	5
25		National Theatre	1	2	3	4	5
26		National Theatre of Opera	1	2	3	4	5
27		Tsagaan darium art museum gallery	1	2	3	4	5
28		Xanadu Art Gallery	1	2	3	4	5

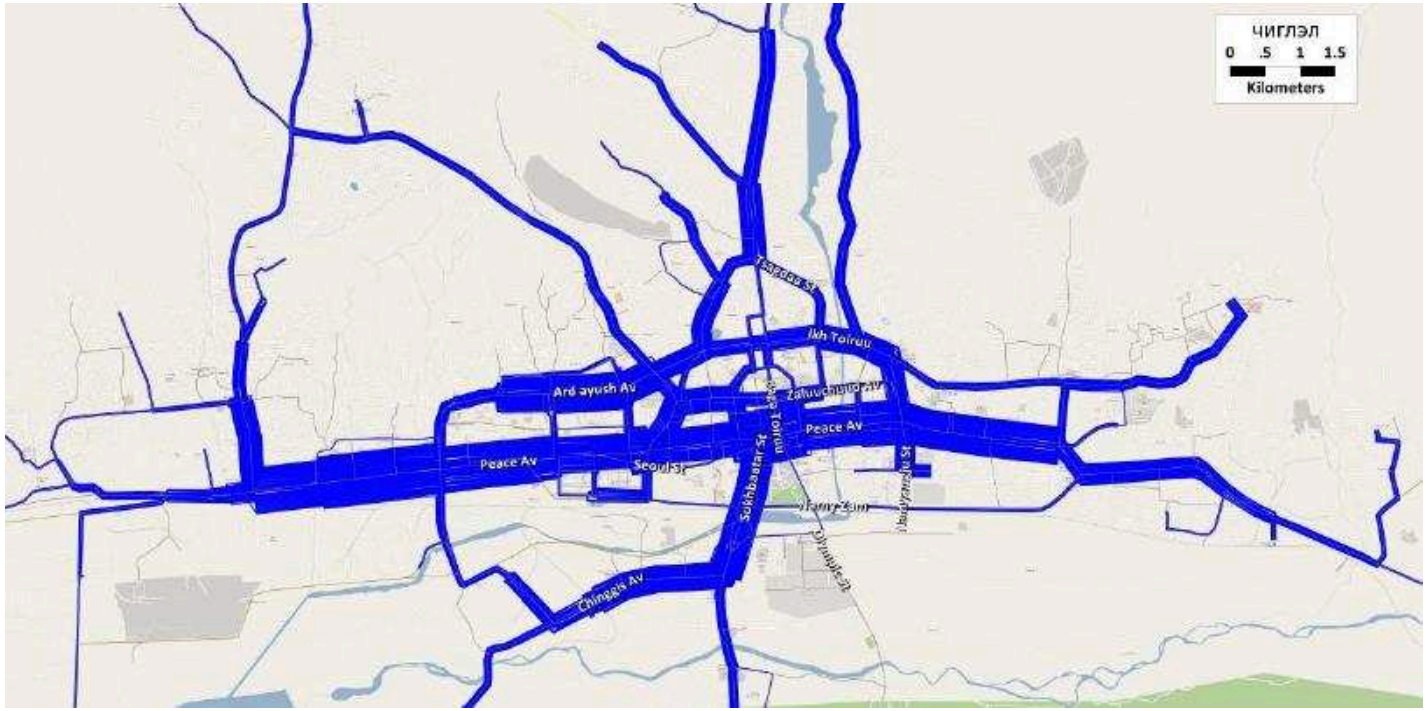
29	Museum	Museum of Choijin Lama	1	2	3	4	5
30		International Intellectual Museum	1	2	3	4	5
31		Winter palace of Bogd Khan	1	2	3	4	5
32		Fine art museum of Zanabazar	1	2	3	4	5
33		National museum of Mongolia	1	2	3	4	5
34		Mongolian National Costume Museum	1	2	3	4	5
35		Museum of National History	1	2	3	4	5

Thank you for your time and cooperation!!! Have a nice flight back home!!!

B.2 Location of the bus stops in UB



B.3 The map of overlapping bus lines in UB



Appendix C. Listing of the TDMCTOPTW Algorithm on C++

```
#include<windows.h>
#include<iostream>
#include<fstream>
#include<vector>
#include"libxl.h"
#include<string>

usingnamespace std;
usingnamespace libxl;

#defineNUM 35

structDataType
{
	double Dugaar;
	constwchar_t* name;
	float X;
	float Y;
	int VisitDuration;
	int EnteranceFee;
	int OpeningHour;
	int ClosingHour;
	int SatisfactionScore;
};

//Declarations
void ReadData();
int minimum(inta, intb)
{
	returna<b ? a :b;
}
int maximum(inta, intb)
{
```

```
	returna>b ? a :b;
}

void htmlRndr(ofstream&htmlFile, stringstr)
{
	htmlFile<<str.c_str() << endl;
}

void main()
{
	//Data load
	int c[NUM][NUM]; // nem
	for (int i = 0; i < 35; i += 2)
	{
Book* book2 = xlCreateBook();
if (book2->load(L"d:\\data\\data.xls"))
           {
              Sheet* sheet1 = book2->getSheet(1);
              if (sheet1)
              {
              for (int row = sheet1->firstRow() + i; row < sheet1 >firstRow() + 2 + i; ++row)
                {
                     for (int col = sheet1->firstCol(); col < sheet1->lastCol(); ++col)
                     {
                     c[row - 1][col] = sheet1->readNum(row, col);
                     }
                 }
             }
         }
         book2->release();
  }
	vector<string> BusName = { "M1", "M2", "M3", "M4", "M6", "M7", "M8", "M9", "M10", "M17B", "M17A", "M18A", "M18B",
"M21B", "M24", "M25", "M27", "M29", "M30", "M32", "M34", "M36", "M37", "M39", "M40", "M42", "M45", "M47", "M48", "M49",
"M50", "M51", "M52", "M53", "M55", "M56", "T2", "T3", "T4", "T5" };
```

```
	vector<string> StopName = { "West 4 Zam", "UB Store", "State Dep.Store", "Mungun Zaviya", "Chinggis Square", "Yonsei Hospital", "Wrestling Palace", "East 4 zam", "Gandan", "Bayanburd", "UB Station", "Bars", "Narantuul Market", "Ajilchdiin Soyol tov", "120", "National Library", "NUM", "Ard Cinema", "MCS", "NUA", "Zaisan", "Statue of Soviet", "Tengis Cinema", "Bombogor Market", "National Park" };
	vector<vector<double>> StopLoc = { { 47.915292, 106.897627 }, { 47.915609, 106.90164 }, { 47.916184, 106.908265 }, { 47.916428, 106.908678 },{ 47.916587, 106.918049 }, { 47.918039, 106.926337 }, { 47.918312, 106.933735 }, { 47.918758, 106.941122 },{ 47.91934, 106.891034 }, { 47.928216, 106.908514 }, { 47.909633, 106.884726 }, { 47.908741, 106.891678 },{ 47.908484, 106.948197 }, { 47.899132, 106.899682 }, { 47.901326, 106.909896 }, { 47.914379, 106.915614 },{ 47.922511, 106.922062 }, { 47.91778, 106.911655 }, { 47.895492, 106.908544 }, { 47.886561, 106.910982 },{ 47.886561, 106.910982 }, { 47.881636, 106.912141 }, { 47.922123, 106.904595 }, { 47.919952, 106.899521 }, { 47.908302, 106.922867 } };
	vector<vector<int>> BusStop;
	vector<vector<int>> BusStopSq;
	for (int i = 0; i < 40; i += 2)
	{
	Book* book = xlCreateBook();
	if (book->load(L"d:\\data\\data2.xls"))
	{
		Sheet* sheet = book->getSheet(0);
		if (sheet)
		{
		for (int row = sheet->firstRow() + i; row < sheet->firstRow() + 2 + i; ++row)
		{
			vector<int> temp;
			//cout << row << ": ";
			for (int col = sheet->firstCol(); col < sheet >lastCol(); ++col)
			{
			//cout << sheet->readNum(row, col) << ", ";
			temp.push_back(sheet->readNum(row, col));
			}
			//cout << endl;
			BusStop.push_back(temp);
		}
		}
	}
	book->release();
```

```
}
for (int i = 0; i < 40; i += 2)
{
Book* book = xlCreateBook();
if (book->load(L"d:\\data\\data2.xls"))
{
	Sheet* sheet = book->getSheet(1);
	if (sheet)
	{
	for (int row = sheet->firstRow() + i; row < sheet->firstRow() + 2 + i; ++row)
	{
		vector<int> temp;
		for (int col = sheet->firstCol(); col < sheet->lastCol(); ++col)
		{
		temp.push_back(sheet->readNum(row, col));
		}
		BusStopSq.push_back(temp);
	}
	}
}
book->release();
}
DataTypedt[NUM];
for (int i = 0; i < 35; i += 2)
{
Book* book1 = xlCreateBook();
if (book1->load(L"d:\\data\\DATA.xls"))
{
	Sheet* sheet = book1->getSheet(0);
	if (sheet)
	{
	for (int row = sheet->firstRow() + i; row < sheet->firstRow() + 2 + i; ++row)
	{
	dt[row - 1].Dugaar = sheet->readNum(row, 0);
	dt[row - 1].name = sheet->readStr(row, 1);
```

```
        dt[row - 1].X = sheet->readNum(row, 2);
        dt[row - 1].Y = sheet->readNum(row, 3);
        dt[row - 1].VisitDuration = sheet->readNum(row, 4);
        dt[row - 1].EnteranceFee = sheet->readNum(row, 5);
        dt[row - 1].OpeningHour = sheet->readNum(row, 6);
        dt[row - 1].ClosingHour = sheet->readNum(row, 7);
        dt[row - 1].SatisfactionScore = sheet->readNum(row, 8);
        }
        }
        }
}
//end Data load
//OD search
vector<int> Wait(NUM, 0);
vector<int> MaxShift(NUM, 0);
vector<int> Shift(NUM, 0);
vector<int> a(NUM, 0);
vector<int> s(NUM, 0);

float Lat = 47.912617, Lon = 106.924755; //47.921940, Lon = 106.934090; // UBHotel

// Search closest POI
float min1 = 99999999.9;
float min2 = 99999999.9;
float mini1 = 0, mini2 = 0;
for (int i = 0; i <NUM; i++)
{
        float R = 6371.0;
        float dLat = (dt[i].X - Lat) * 3.14159 / 180;
        float dLon = (dt[i].Y - Lon) * 3.14159 / 180;
        float Lat1 = Lat * 3.14159 / 180;
        float Lat2 = dt[i].X * 3.14159 / 180;
        float a = sin(dLat / 2) * sin(dLat / 2) + sin(dLon / 2) * sin(dLon / 2) * cos(Lat1) * cos(Lat2);
        float c = 2 * atan2(sqrtf(a), sqrtf(1 - a));
        float d = R * c;
```

```
        if (min1 > d)
        {
                min2 = min1;
                mini2 = mini1;
                min1 = d;
                mini1 = i;
        }

        int bb;
}

int startPos = mini1;
int endPos = mini2;

// end of Search closest POT
vector<int> Visit;

// 1st POI
Visit.push_back(startPos);
a[startPos] = 120;
Wait[0] = maximum(0, dt[startPos].OpeningHour - a[startPos]);
s[startPos] = a[startPos] + Wait[startPos];
MaxShift[0] = dt[startPos].ClosingHour - s[startPos];
Shift[0] = 0;

// 2nd POI
Visit.push_back(endPos);
a[1] = a[0] + dt[0].VisitDuration + c[0][1];
Wait[1] = maximum(0, dt[1].OpeningHour - a[1]);
s[1] = a[1] + Wait[1];
MaxShift[1] = dt[1].ClosingHour - s[1];
Shift[1] = 0;

vector<int> Bo(NUM, 1);
```

```
	Bo[Visit[0]] = 0;
	Bo[Visit[1]] = 0;
	int n = 2;

	// Insertion method % Bo.insert(Bo.begin() + 2, 5);
	for (int l = 0; l < 6; l++)
	{
	int Ratio = 0;
	int Inc = 33;
	int Shiftj;
int Pos;
	for (int j = 0; j <NUM; j++)
	{
	if (Bo[j])
	{
	for (int i = 0; i < Visit.size() - 1/* length */; i++)
		{
		int aShift = s[i] + dt[Visit[i]].VisitDuration + c[Visit[i]][j];
		int Shift2 = maximum(0, dt[j].OpeningHour - aShift);
		//cout << Shift2 << "+" <<c[Visit[i]][j] << "+" << c[j][Visit[i + 1]] << "-" << c[Visit[i]][Visit[i + 1]] <<
endl;
		Shift2 = Shift2 + c[Visit[i]][j] + c[j][Visit[i + 1]] - c[Visit[i]][Visit[i + 1]];
		//cout << j << ", " << i << ", " << Shift2 << ", " <<Wait[i + 1] + MaxShift[i + 1] << endl;
		if (Shift2 <= Wait[i + 1] + MaxShift[i + 1])
		{
		if (Shift2 == 0) Shift2 = 1;
		float iRatio = dt[j].SatisfactionScore * dt[j].SatisfactionScore / Shift2;
		if (iRatio >= Ratio)
		{
			Ratio = iRatio;
			Pos = i;
			Inc = j;
			Shiftj = Shift2;
		}
		}
```

```
                }
        }
}
if (Inc <= NUM)
{
        Bo[Inc] = 0;
        n = Visit.size();
        Visit.insert(Visit.begin() + Pos + 1, Inc);
        a.insert(a.begin() + Pos, s[Pos] + dt[Visit[Pos]].VisitDuration + c[Visit[Pos]][Inc]);
s.insert(s.begin() + Pos, maximum(a[Pos + 1], dt[Inc].OpeningHour));
        Wait.insert(Wait.begin() + Pos, maximum(0, dt[Inc].OpeningHour - a[Pos + 1]));
        MaxShift.insert(MaxShift.begin() + Pos, 0);
        Shift.insert(Shift.begin() + Pos, 0);

        for (int k = Pos + 2; k < n + 1; k++)
        {
        Wait[k] = maximum(0, Wait[k] - Shift[k - 1]);
        a[k] = a[k] + Shift[k - 1];
        Shift[k] = maximum(0, Shift[k - 1] - Wait[k]);
        s[k] = s[k] = Shift[k];
        MaxShift[k] = MaxShift[k] - Shift[k];
        }
        MaxShift[Pos + 1] = minimum(dt[Inc].ClosingHour - s[Pos + 1], MaxShift[Pos + 2] + Wait[Pos + 2]);
        for (int k = Pos; k < 1; k++)
        {
        MaxShift[k] = minimum(dt[Visit[k]].ClosingHour - s[k], MaxShift[k + 1] + Wait[k + 1]);
        }
        n++;
        }
}
        for (int i = 0; i < Visit.size(); i++)
        {
                cout<< i + 1 <<" visit: "<< Visit[i] + 1 <<":: ";
                wcout<< dt[Visit[i]].name << endl;
        }
```

```
cout<<"************"<< endl;

// Hotel to nearest Bus stop
for (int xaix = 0; xaix < Visit.size() - 1; xaix++)
{
	float cl = 9999.9;
	int st = 0;

	for (int stopName = 0; stopName < 25; stopName++)
	{
		float R = 6371.0;
		float dLat = (StopLoc[stopName][0] - dt[Visit[xaix]].X) * 3.14159 / 180;
		float dLon = (StopLoc[stopName][1] - dt[Visit[xaix]].Y) * 3.14159 / 180;
		float Lat1 = dt[Visit[0]].X * 3.14159 / 180;
		float Lat2 = StopLoc[stopName][0] * 3.14159 / 180;
		float a = sin(dLat / 2) * sin(dLat / 2) + sin(dLon / 2) * sin(dLon / 2) * cos(Lat1) * cos(Lat2);
		float c = 2 * atan2(sqrtf(a), sqrtf(1 - a));
		float d = R * c;
		if (cl > d)
		{
			cl = d;
			st = stopName;
		}
	}
	cout<< st + 1 <<" dugaariin zogsooloos: "<< StopName[st] << endl;

	// ochix gazriin xamgiin oirxon buudal
	cl = 9999.9;
	int st1 = 0;
	for (int stopName = 0; stopName < 25; stopName++)
	{
		float R = 6371.0;
		float dLat = (StopLoc[stopName][0] - dt[Visit[xaix + 1]].X) * 3.14159 / 180;
		float dLon = (StopLoc[stopName][1] - dt[Visit[xaix + 1]].Y) * 3.14159 / 180;
		float Lat1 = dt[Visit[1]].X * 3.14159 / 180;
```

```
        float Lat2 = StopLoc[stopName][0] * 3.14159 / 180;
        float a = sin(dLat / 2) * sin(dLat / 2) + sin(dLon / 2) * sin(dLon / 2) * cos(Lat1) * cos(Lat2);
        float c = 2 * atan2(sqrtf(a), sqrtf(1 - a));
        float d = R * c;
        //cout <<BusStopName[stopName] << ": " << d << endl;
        if (cl > d)
        {
                cl = d;
                st1 = stopName;
        }
}
int num = 0;
cout<< st1 + 1 <<"  dugaariin zogsool ruu: "<< StopName[st1] << endl;
// 2 buudal deer zogsdog bus xaix BusStop vector
bool alx = false;
for (int i = 0; i < 40; i++)
{
        if (BusStop[i][st] == 1 && BusStop[i][st1] == 1)
        {
                cout<<"Found it!: "<< BusName[i];
                alx = true;
                bool started = false;
                for (int k = 0; k < BusStopSq[i].size(); k++)
                {
                        if (st + 1 == BusStopSq[i][k] || st1 + 1 == BusStopSq[i][k])
                        {
                                cout<<", "<< BusStopSq[i][k];
                                started = !started;
                        }
                        elseif (started)
                        {
                                cout<<", "<< BusStopSq[i][k];
                                continue;
```

```
#include<windows.h>
#include<iostream>
#include<fstream>
#include<vector>
#include"libxl.h"
#include<string>

usingnamespace std;
usingnamespace libxl;

#defineNUM 35

structDataType
{
	double Dugaar;
	constwchar_t* name;
	float X;
	float Y;
	int VisitDuration;
	int EnteranceFee;
	int OpeningHour;
	int ClosingHour;
	int SatisfactionScore;
};

//Declarations
void ReadData();
int minimum(int a, int b)
{
	return a < b ? a : b;
}
	int maximum(int a, int b)
{
	return a > b ? a : b;
```

```
}

	void htmlRndr(ofstream &htmlFile, string str)
{
	htmlFile<< str.c_str() << endl;
}

	void main()
{
	//Data load
		int c[NUM][NUM]; // nem
		for (int i = 0; i < 35; i += 2)
		{
		Book* book2 = xlCreateBook();
		if (book2->load(L"d:\\data\\data.xls"))
		{
			Sheet* sheet1 = book2->getSheet(1);
			if (sheet1)
			{
			for (int row = sheet1->firstRow() + i; row < sheet1->firstRow() + 2 + i; ++row)
			{
			for (int col = sheet1->firstCol(); col < sheet1->lastCol(); ++col)
			{
					c[row - 1][col] = sheet1->readNum(row, col);
				}
			}
			}
		}
		book2->release();
		}
		vector<string> BusName = { "M1", "M2", "M3", "M4", "M6", "M7", "M8", "M9", "M10", "M17B",
"M17A", "M18A", "M18B", "M21B", "M24", "M25", "M27", "M29", "M30", "M32", "M34", "M36", "M37", "M39", "M40", "M42", "M45",
"M47", "M48", "M49", "M50", "M51", "M52", "M53", "M55", "M56", "T2", "T3", "T4", "T5" };
		vector<string> StopName = { "West 4 Zam", "UB Store", "State Dep.Store", "Mungun Zaviya",
"Chinggis Square", "Yonsei Hospital", "Wrestling Palace", "East 4 zam", "Gandan", "Bayanburd", "UB Station", "Bars",
```

```
"Narantuul Market", "Ajilchdiin Soyol tov", "120", "National Library", "NUM", "Ard Cinema", "MCS", "NUA", "Zaisan", "Statue
of Soviet", "Tengis Cinema", "Bombogor Market", "National Park" };
                                vector<vector<double>> StopLoc = { { 47.915292, 106.897627 }, { 47.915609, 106.90164 }, {
47.916184, 106.908265 }, { 47.916428, 106.908678 }, { 47.916587, 106.918049 }, { 47.918039, 106.926337 }, { 47.918312,
106.933735 }, { 47.918758, 106.941122 }, { 47.91934, 106.891034 }, { 47.928216, 106.908514 }, { 47.909633, 106.884726 }, {
47.908741, 106.891678 }, { 47.908484, 106.948197 }, { 47.899132, 106.899682 }, { 47.901326, 106.909896 }, { 47.914379,
106.915614 }, { 47.922511, 106.922062 }, { 47.91778, 106.911655 }, { 47.895492, 106.908544 }, { 47.886561, 106.910982 }, {
47.886561, 106.910982 }, { 47.881636, 106.912141 }, { 47.922123, 106.904595 }, { 47.919952, 106.899521 }, { 47.908302,
106.922867 } };
                                vector<vector<int>> BusStop;
                                vector<vector<int>> BusStopSq;
                                for (int i = 0; i < 40; i += 2)
                                {
                                Book* book = xlCreateBook();
                                if (book->load(L"d:\\data\\data2.xls"))
                                {
                                        Sheet* sheet = book->getSheet(0);
                                        if (sheet)
                                        {
                                        for (int row = sheet->firstRow() + i; row < sheet->firstRow() + 2 + i; ++row)
                                        {
                                                vector<int> temp;
                                                //cout << row << ": ";
                                                for (int col = sheet->firstCol(); col < sheet->lastCol(); ++col)
                                                {
                                                        //cout << sheet->readNum(row, col) << ", ";
                                                        temp.push_back(sheet->readNum(row, col));
                                                }
                                                //cout << endl;
                                                BusStop.push_back(temp);
                                        }
                                        }
                                }
                                book->release();
                                }
```

```
                                for (int i = 0; i < 40; i += 2)
                                {
                                Book* book = xlCreateBook();
                                if (book->load(L"d:\\data\\data2.xls"))
                                {
                                       Sheet* sheet = book->getSheet(1);
                                       if (sheet)
                                       {
                                              for (int row = sheet->firstRow() + i; row < sheet->firstRow() + 2 + i;
++row)
                                              {
                                              vector<int> temp;
                                              for (int col = sheet->firstCol(); col < sheet->lastCol(); ++col)
                                              {
                                                     temp.push_back(sheet->readNum(row, col));
                                              }
                                              BusStopSq.push_back(temp);
                                              }
                                       }
                                }
                                book->release();
                                }
                                DataTypedt[NUM];

        //*****************************************************************************************

        //*****************************************************************************************

               int choice1 = 0; //0~7 Love it
               int choice2 = 1; //0~7 Like it
               int choice3 = 2; //0~7 Okey



        //*****************************************************************************************
```

```
//*****************************************************************************************

	for (int i = 0; i < 35; i += 2)
	{
		Book* book1 = xlCreateBook();
		if (book1->load(L"d:\\data\\DATA.xls"))
		{
			Sheet* sheet = book1->getSheet(0);
			if (sheet)
			{
				for (int row = sheet->firstRow() + i; row < sheet->firstRow() + 2 + i; ++row)
				{
					dt[row - 1].Dugaar = sheet->readNum(row, 0);
					dt[row - 1].name = sheet->readStr(row, 1);
					dt[row - 1].X = sheet->readNum(row, 2);
					dt[row - 1].Y = sheet->readNum(row, 3);
					dt[row - 1].VisitDuration = sheet->readNum(row, 4);
					dt[row - 1].EnteranceFee = sheet->readNum(row, 5);
					dt[row - 1].OpeningHour = sheet->readNum(row, 6);
					dt[row - 1].ClosingHour = sheet->readNum(row, 7);
					int add1 = sheet->readNum(row, 8 + choice1);
					int add2 = sheet->readNum(row, 8 + choice2);
				int add3 = sheet->readNum(row, 8 + choice3);
					dt[row - 1].SatisfactionScore = add1 * 3 + add2 * 2 + add3;
				}
			}
		}
	}
//end Data load
//OD search
vector<int> Wait(NUM, 0);
vector<int> MaxShift(NUM, 0);
vector<int> Shift(NUM, 0);
vector<int> a(NUM, 0);
```

```
	vector<int> s(NUM, 0);

	float Lat = 47.908892, Lon = 106.899553; //47.921940, Lon = 106.934090; // UBHotel

	// Search closest POI
	float min1 = 99999999.9;
	float min2 = 99999999.9;
	float mini1 = 0, mini2 = 0;
	for (int i = 0; i <NUM; i++)
	{
		float R = 6371.0;
		float dLat = (dt[i].X - Lat) * 3.14159 / 180;
		float dLon = (dt[i].Y - Lon) * 3.14159 / 180;
		float Lat1 = Lat * 3.14159 / 180;
		float Lat2 = dt[i].X * 3.14159 / 180;
		float a = sin(dLat / 2) * sin(dLat / 2) + sin(dLon / 2) * sin(dLon / 2) * cos(Lat1) * cos(Lat2);
		float c = 2 * atan2(sqrtf(a), sqrtf(1 - a));
		float d = R * c;

		if (min1 > d)
		{
			min2 = min1;
			mini2 = mini1;
			min1 = d;
			mini1 = i;
		}

		int bb;
	}

	int bestSat = 0;
	int GoodSat = 0;
int idx1 = 0, idx2 = 0;
	bool isChosen = false;
	for (int i = 0; i <NUM; i++)
```

```
{
	cout<< dt[i].SatisfactionScore << endl;
	if (dt[i].SatisfactionScore >= bestSat)
{
		if (isChosen)
		{
			GoodSat = bestSat;
			idx2 = idx1;
		}
			bestSat = dt[i].SatisfactionScore;
			idx1 = i;
			isChosen = true;
	}
}
cout<< bestSat <<" at "<< idx1 <<", "<< GoodSat <<" at "<< idx2 << endl;
int startPos = idx1;
int endPos = idx2;

// end of Search closest POT
vector<int> Visit;

// 1st POI
Visit.push_back(startPos);
a[startPos] = 120;
Wait[0] = maximum(0, dt[startPos].OpeningHour - a[startPos]);
s[startPos] = a[startPos] + Wait[startPos];
MaxShift[0] = dt[startPos].ClosingHour - s[startPos];
Shift[0] = 0;

// 2nd POI
Visit.push_back(endPos);
a[1] = a[0] + dt[0].VisitDuration + c[0][1];
Wait[1] = maximum(0, dt[1].OpeningHour - a[1]);
s[1] = a[1] + Wait[1];
MaxShift[1] = dt[1].ClosingHour - s[1];
```

```
	Shift[1] = 0;

	vector<int> Bo(NUM, 1);
	Bo[Visit[0]] = 0;
	Bo[Visit[1]] = 0;
	int n = 2;

	// Insertion method % Bo.insert(Bo.begin() + 2, 5);
	for (int l = 0; l < 6; l++)
	{
		int Ratio = 0;
		int Inc = 33;
		int Shiftj;
		int Pos;
		for (int j = 0; j <NUM; j++)
		{
			if (Bo[j])
			{
			for (int i = 0; i < Visit.size() - 1/* length */; i++)
			{
				int aShift = s[i] + dt[Visit[i]].VisitDuration + c[Visit[i]][j];
				int Shift2 = maximum(0, dt[j].OpeningHour - aShift);
				//cout << Shift2 << "+" <<c[Visit[i]][j] << "+" << c[j][Visit[i + 1]] << "-" <<
c[Visit[i]][Visit[i + 1]] << endl;
				Shift2 = Shift2 + c[Visit[i]][j] + c[j][Visit[i + 1]] - c[Visit[i]][Visit[i + 1]];
				//cout << j << ", " << i << ", " << Shift2 << ", " <<Wait[i + 1] + MaxShift[i + 1] << endl;
				if (Shift2 <= Wait[i + 1] + MaxShift[i + 1])
				{
					if (Shift2 == 0) Shift2 = 1;
					float iRatio = dt[j].SatisfactionScore * dt[j].SatisfactionScore / Shift2;
					if (iRatio >= Ratio)
					{
						Ratio = iRatio;
						Pos = i;
						Inc = j;
```

```
                                        Shiftj = Shift2;
                                    }
                                }
                            }
                        }
                }
        if (Inc <= NUM)
        {
        Bo[Inc] = 0;
                n = Visit.size();
                Visit.insert(Visit.begin() + Pos + 1, Inc);
                a.insert(a.begin() + Pos, s[Pos] + dt[Visit[Pos]].VisitDuration + c[Visit[Pos]][Inc]);
                s.insert(s.begin() + Pos, maximum(a[Pos + 1], dt[Inc].OpeningHour));
                Wait.insert(Wait.begin() + Pos, maximum(0, dt[Inc].OpeningHour - a[Pos + 1]));
                MaxShift.insert(MaxShift.begin() + Pos, 0);
                Shift.insert(Shift.begin() + Pos, 0);

                for (int k = Pos + 2; k < n + 1; k++)
                {
                        Wait[k] = maximum(0, Wait[k] - Shift[k - 1]);
                        a[k] = a[k] + Shift[k - 1];
                        Shift[k] = maximum(0, Shift[k - 1] - Wait[k]);
                        s[k] = s[k] = Shift[k];
                        MaxShift[k] = MaxShift[k] - Shift[k];
                }
                MaxShift[Pos + 1] = minimum(dt[Inc].ClosingHour - s[Pos + 1], MaxShift[Pos + 2] + Wait[Pos + 2]);
                for (int k = Pos; k < 1; k++)
                {
                        MaxShift[k] = minimum(dt[Visit[k]].ClosingHour - s[k], MaxShift[k + 1] + Wait[k + 1]);
                }
                n++;
        }
}
for (int i = 0; i < Visit.size(); i++)
{
```

```
        cout<< i + 1 <<" visit: "<< Visit[i] + 1 <<":: ";
        wcout<< dt[Visit[i]].name << endl;
}
        cout<<"************"<< endl;

// For each included Visits in BestFound
for (int xaix = 0; xaix < Visit.size() - 1; xaix++)
{
        float cl = 9999.9;
        int st = 0;
        //find the nearest bus stop
        for (int stopName = 0; stopName < 25; stopName++)
{
                float R = 6371.0;
                float dLat = (StopLoc[stopName][0] - dt[Visit[xaix]].X) * 3.14159 / 180;
                float dLon = (StopLoc[stopName][1] - dt[Visit[xaix]].Y) * 3.14159 / 180;
                float Lat1 = dt[Visit[0]].X * 3.14159 / 180;
                float Lat2 = StopLoc[stopName][0] * 3.14159 / 180;
                float a = sin(dLat / 2) * sin(dLat / 2) + sin(dLon / 2) * sin(dLon / 2) * cos(Lat1) * cos(Lat2);
                float c = 2 * atan2(sqrtf(a), sqrtf(1 - a));
                float d = R * c;
                if (cl > d)
                {
                        cl = d;
                        st = stopName;
                }
        }
        // find the nbs near to the next visiting place;
        cl = 9999.9;
        int st1 = 0;
        for (int stopName = 0; stopName < 25; stopName++)
        {
                float R = 6371.0;
                float dLat = (StopLoc[stopName][0] - dt[Visit[xaix + 1]].X) * 3.14159 / 180;
                float dLon = (StopLoc[stopName][1] - dt[Visit[xaix + 1]].Y) * 3.14159 / 180;
```

```
                float Lat1 = dt[Visit[1]].X * 3.14159 / 180;
                float Lat2 = StopLoc[stopName][0] * 3.14159 / 180;
                float a = sin(dLat / 2) * sin(dLat / 2) + sin(dLon / 2) * sin(dLon / 2) * cos(Lat1) * cos(Lat2);
                float c = 2 * atan2(sqrtf(a), sqrtf(1 - a));
                float d = R * c;
                if (cl > d)
                {
                        cl = d;
                        st1 = stopName;
                }
        }
        int num = 0;
        bool alx = false;
        //walk to the next visiting place
        if (st == st1)
        {
                cout<<"Alxsan n deerdee xajuud chin!\n"<< endl;
                alx = true;
        }
        else
{
                cout<< st + 1 <<" dugaariin zogsooloos: "<< StopName[st] << endl;
                //get list of busses that stops  at nbs
                for (int i = 0; i < 40; i++)
                {
                        if (BusStop[i][st] == 1 && BusStop[i][st1] == 1)
                        {
                                cout<<"Found it!: "<< BusName[i];
                                alx = true;
                                bool started = false;
                                for (int k = 0; k < BusStopSq[i].size(); k++)
                                {
                                        //select the shortest bus line to the nbs
                                if (st + 1 == BusStopSq[i][k] || st1 + 1 == BusStopSq[i][k])
                                        {
```

```
                                        cout<<", "<< BusStopSq[i][k];
                                        started = !started;
                                }
                                elseif (started)
                                {
                                        cout<<", "<< BusStopSq[i][k];
                                        continue;
                                }
                        }
                        cout<< endl;
                        num = i;
                }
        }
}
if (!alx) cout <<"Alxsan n deerdee!\n"<< endl;
else {
        if (st != st1)
        {
                cout<< num + 1 <<": "<< BusName[num] << endl;
                cout<< st1 + 1 <<"  dugaariin zogsool ruu: "<< StopName[st1] << endl;
        }
}
//take off the bus and walk to the next visiting place
                                                wcout<< dt[Visit[xaix + 1]].name << endl;
                                                cout<< endl <<"************************"<< endl << endl;
                                        }
                                        int aa = 0;
                                        system("PAUSE");
                                }
                        }
                        cout<< endl;
                        num = i;
                }
        }
        if (!alx) cout <<"Alxsan n deerdee!\n"<< endl;
```

```
                else {
                        cout<< num + 1 <<": "<< BusName[num] << endl;
                        cout<< endl;
                }
        }
        system("PAUSE");
}
```

Bibliography

(n.d.). Retrieved from http://www.doc.ic.ac.uk/: http://www.doc.ic.ac.uk/~nd/surprise_96/journal/vol1/tcw2/article1.html#ECON

A Bock and L.Sanita. (2014). A Capacited orienteering problem. Discrete Applied mathematics.

A.Garcia, P.Vansteenwegen, o.Arbelaitz, W.Souffriau, M.T.Linaza. (2013). Integrating public transportation in personalized electronic tourist guides. Computers and Operations research 40, 758-774.

A.Garcia, P.Vansteenwegen, Wouter Souffriau, Olatz Arbelaitz , Maria Teresa Liaza. (2010). Solving Multi constrained team orienteering problems to generate tourist routes. . Computers and Industrial engineering, 1-19.

A.Leifer,M.Rosenwein,. (1994). Strong linear programming relaxations for the Orienteering problem. European Journal Operations Research, (pp. 517-523).

Ander Garcia, Olatz Arbelaitz, Pieter Vansteenwegen,Wouter Souffriau, and Maria Teresa Linaza. (2010). Hybrid Approach for the Public Transportation Time Dependent Orienteering Problem with Time Windows. Springer-Verlag Berlin Heidelberg , 151-158.

B.L.Golden, L.Levy and R.Vohra. (1987). The Orienteering Problem. Naval Research Logistics, 307-318.

C.Archetti, A.Hertz, M.Speranza. (2007). Metaheuristics for the team orienteering problem . Journal of Heurictics, 49-76.

Chao, B. L. Golden, and E. A. Wasil. (1996). The team orienteering problem. European Journal of Operational Research, 88, 464-474.

D.Gavalas, Ch.Konstantopoulos, K.Mastakas, G.Pantziou and N.Vathis. (2013). Efficient heuristics for the time dependent team orienteering problem with time windows. eCompass.

D.Gavalas, Ch.Konstantopoulos, K.Mastakas, G.Pantziou and N.Vathis. (2015). Heuristics for the time dependent team orienteering problem: Application to tourist route planning. Computers and Operations research, 36-50.

D.Gavalas, Ch.Konstantopoulos, K.Mastakas, G.Pantziou, Y.Tasoulas. (2012). A Survey on Algorithmic Approaches for Solving Tourist Trip Design Problems. eCOMPASS.

D.Gavalas, M.Kenteris, Ch.Konstantopoulos, G.Pantziou. (2011). Personalized routes for mobile tourism. International coneference on wireless and mobile computing, networking and comunications (pp. 295-300). IEEE.

E.Miller, A.W.Tucker, R.A.Zemlin. (1960). Integer Programming formulations and traveling salesman problems. Journal of the ACM, 326-329.

F.Tricoire, M.Romauch, M.Doerner and R.Hartl. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. Computers and Operations Rsearch, 351-367.

G.Laporte and S.Martello. (1990). The selective travelling salesman problem. Discrete a pplied mathematics 26 , 193-207.

H.Bouly, D.C.Dang, A.Moukrim. (2008). A Memetic algorithm for the team orienteering problem . Evo Workshop 2008, LNCS 4974 (pp. 649-658). Berlin: Springer.

H.Lourenco, O.Martin, T.Stutzle. (2003). Iterated local search. In Handbook of Metaheuristics (pp. 321-353). Springer.

H.Tang and E.Miller-Hooks. (2005). A tabu search heuristic for the team orienteering problem. Computers and Operations research 32, 1379-1407.

Helena R. Louren¸co, Olivier C. Martin, and Thomas Stutzle. (n.d.). Iterated Local Search: Framework and Applications. 1-38.

I.Chao, B.Golden, E.Wasil. (1996). A fast and effective heuristic for the orienteering problem. European Journal of Operational Research 88, 475-489.

J.K.Chilinska and P.Zabielska. (2013). Genetic Algorithm solving the Orienteering Problem with time windows. Proceedings of the International Conference on Systems Science (pp. 609-619). Springer International Publishing.

Joanna Karbowska-Chilinska and Pawel Zabielski. (2013). Genetic Algorithm with Path Relinking for the. 245-258.

Kh.Bulga. (2015). Improving the public transportation system in Ulaanbaatar city. PhD thesis.

L.Ke, C.Archetti, Z.Feng. (2008). Ants can solve the team orienteering problem. Computers and Industrial Engineering, 648-665.

M.Fischetti, J.Salazar, P.Toth. (1998). Solving the orienteering problem through branch and cut. INFORMS Journal on Computing 10, 133-148.

M.Gendreau, G.Laporte, F.Semet. (1998). A Tabu search heuristic for the undirective selective traveling salesman problem. European Journal of Operational research 106, 539-545.

N.Labadi, J.Melechovsky, R.W.Calvo. (2010). An effective hybrid evolutionary local search for orienteering and team orienteering problem with time windows. PPSN 11, Part 2, LNCS 6239 (pp. 219-228). Berlin: Springer.

P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. (2009). A guided local search metaheuristic for the team orienteering problem. European Journal of Operation research, 118-127.

P.Vansteennwegen, W.Souffria, G.V.Berghe, D.V.Oudheusden. (2009). Metaheuristics for Tourist trip planning. Metaheuristics in the Service Industry, 15-31.

P.Vansteenwegen, W.Souffriau, D.V.Oudheusden. (2011). The orienteering problem: A Survey. European Journal of Operational Research, 1-10.

P.Vansteenwegen, W.Souffriau, G,V.Berghe, D.V.Oudhesden. (2009). Iterated Local Search for the team orienteering problem with time windows. Computers and operations research 36, 3281-3290.

Papadimitriou, C.H.; Steiglitz, K. (1998). Combinatorial optimization: algorithms and complexity. 308-309.

Pieter Vansteenwegen, Wouter Souffriau, GreetVanden Berghe, and DirkVan Oudheusden. (2009). Metaheuristics for tourist trip planning. Metaheuristics in the Service Industry, 15-31.

R.A.Abbaspour and F.Samadzadegen. (2011). Time-dependent personal tour planning and scheduling in metropolises. Expert systems with Applications, 12439-12452.

R.A.Abbaspour and F.Samadzadegen. (2011). Time-dependent personal tour planning and scheduling in metropolises. Expert Systems with Applications, 12439–12452.

R.Chelouah and P.Siarry. (2000). A Continuous genetic algorithm designed for the Global optimization of Multimodal functions. Journal of Heuristics, 191-213.

R.Montemmani, L. Gamberdella, . (2009). Ant Colony System for Team Orienteering Problems with Time Windows. Found.Comput.Decision Sci.34(4) , 287-306.

Rodrigues, A. M. and Soeiro Ferreira, J. (2015). Waste Collection Routing—Limited Multiple Landfills and Hetergenous Fleet. Wiley Online library, Networks. doi: 10.1002/net.21597.

S.Boussier, D.Feillet and M.Gendreau. (2007). An exact algorithm for team orienteering problems. Operational Research, 211-230.

S.Butt and T.Cavalier. (1992). A heuristic for the multiple tour maximum collection problem. Computers and Operations Research, 101-111.

S.N.Sivanandam and S.N.Deepa. (2008). Introduction to Genetic Algorithms. Berlin Heidelberg: Springer.

Sh.X.Lin, F.V.Yu. (2012). A simulated annealing heuristic for the team orienteering problem with time windows. European Journal of Operational Research 217 , 94-107.

Sylejmani, K., Dorn, J.& Muslua,N. (2012). A Tabu search approach for multi constrained team orienteering problem and its application in touristic trip planning. Proceedings of 2th International Conference of Hybrid Intelligent Systems.

T.Kinoshita, M.nagata, Yo.Murata, N.Shibata, K.Yasumoto, M.Ito. (2006). A Personal navigation system for sightseeing across multiple days. ISMU, 254-259.

T.Shiraishi, M.Nagata, N.Shibata, Y.Murata, K.Yasumoto, M.Ito. (2005a). A Personal navigation system with a Schedule planning facility based on multi-objective criteria. International conference on mobile computing and ubiquitous networking, (pp. 104-109).

T.Tsiligirides. (1984). Heuristic methods applied to orienteering. The Journal of the Operational Research Society, 797-809.

Tasgetiren, M. (2001). A Genetic Algorithm with an adaptive penalty function for the orienteering problem. Journal of economic and social research 4 (2), 1-26.

Tucker, A. W. (1960). On Directed Graphs and Integer Programs. IBM Mathematical research Project (Princeton University).

W. Souffriau, P. Vansteenwegen, G. Vanden Berghe, and D. Van Oudheusden. (2010). A path relinking approach for the team orienteering problem. Computers & Operations Research (37), 1853 -1859.

W.Souffriau. (2010). Automated Tourist Decision Support. Leuven: Katholoieke Universiteit Leuven.

W.Souffriau, P.Vansteenwegen, J.Vertommen, G.B.Vanden.V.Oudhesden. (2008). A personalized tourist trip design algorithm for mobile tourist guides . Applied Artificial Intelligence 22 (10), 964-985.

Y.Kurata. (2009). Interactive Assistance for Tour Planning. AGILE 09 (pp. 1-14). GISA-japan research conference.

Z.Michalewicz and David B.Fogel. (2004). How to solve it:Modern heuristics.

Zulal Sevkli and Erdogen Sevilgen. (2006). Variable Neighborhood Search for the Orienteering Problem. ISCIS 2006, LNCS 4263 (pp. 134-143). Berlin: Springer.

Zülal Sevkli and F. Erdoğan Sevilgen. (2006). Variable Neighborhood Search for the Orienteering Problem. ISCIS, 134-143.