

**The Roaming Salesman Problem and its
Application to Election Logistics**

by

Masoud Shahmanzari

A Dissertation Submitted to the
Graduate School of Business
in Partial Fulfillment of the Requirements for
the Degree of

Doctor of Philosophy

in

Operation Management and Information Systems



KOÇ ÜNİVERSİTESİ

February 22, 2019

Koç University
Graduate School of Business

This is to certify that I have examined this copy of a doctoral's thesis by

Masoud Shahmanzari

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

Assoc. Prof. Deniz Aksen

Prof. Zeynep Akşin Karaesmen

Assist. Prof. Hande Küçükaydın

Prof. Ahmet Faruk Aysan

Assist. Prof. Barış Yıldız

Date: February 22, 2019



To my beloved family...

A B S T R A C T

Campaign planning is one of the important decisions to make while dealing with determining routes, schedule of the activities, and accommodation planning. The campaign planner is required to plan the schedule of the visits to the customers, to satisfy time constraints, and to organize activities at its best with a proper decision on the scheduling and routing. In this study, we investigate a new problem the goal of which is to determine daily tours for a traveling salesman (referred to as the campaigner) who collects rewards from various cities during a campaign period. We call this new problem the *roaming salesman problem* (RSP) motivated by various real-world applications including election logistics, touristic trip planning, and marketing campaigns. RSP can be characterized as a combination of the traditional *periodic TSP* and the *prize-collecting TSP* with static edge costs and time-dependent vertex rewards. RSP seeks a closed or open tour for each day of a campaign period with the objective of maximizing the net benefit which is defined as the sum of all collected rewards minus the traveling costs. The campaigner is not required to visit all cities, and the daily tours do not have to start and end at the same city. Moreover, he/she can stay overnight in any city to start the tour of the next day. In particular, each city is associated with a base reward and a fixed activity duration. In addition, he/she cannot stay outside the campaign center for more than a given number of consecutive days and the total length of the activities and travel times between cities on the same day cannot exceed a certain maximum tour duration.

We develop a MILP formulation for this problem in which we adopt existing routing constraints and introduce an entirely new class of constraints and binary variables. As an application of RSP in election logistics, we introduce the *multi-period traveling politician problem* (MPTPP). The problem

is tackled efficiently by adapting analytical model and an extensive scenario analysis. Commercial solvers are capable of solving small-size instances of the RSP to near optimality in a reasonable time. To tackle large-size instances we propose a two-phase matheuristic where the first phase deals with the city selection while the second phase focuses on the route generation. The latter capitalizes on an integer program to construct an optimal route among selected cities. The proposed matheuristic decomposes the RSP basically into as many subproblems as the number of campaign days.

We also introduce a new hybrid metaheuristic algorithm for the RSP, called *granular skewed variable neighborhood tabu search* (GSVNTS). It consists of a Granular Tabu Search which is embedded in a Skewed Variable Neighborhood Search algorithm. The suggested method is experimentally tested on the real-life instances including 81 cities and 12 towns in Turkey with actual travel distances. The computational results show that GSVNTS can generate optimal or near-optimal solutions in a small amount of CPU time. Using effective analytical models, the two-phase matheuristic, and GSVNTS, we show that promising results can be obtained to hopefully assist campaign planners in their strategic decision making.

Ö Z E T Ç E

Kampanya planlaması, rota belirleme, etkinlik takvimi programlama ve konaklama planlama ile ilgili olarak alınacak önemli kararlardan biridir. Kampanya planlayıcısının, zamanlama ve yönlendirme konusunda uygun bir kararla en iyi şekilde müşteri ziyaretlerini planlaması, zaman kısıtlarını belirlemesi ve faaliyetleri düzenlemesi gerekmektedir. Bu çalışmada, kampanya süresince çeşitli şehirlerden ödüller toplayarak seyahat eden bir gezgin satıcı için günlük turlar belirlemek amaçlı yeni bir problemi araştırıyoruz. Bu yeni probleme, seçim lojistiği, turistik gezi planlaması ve pazarlama kampanyaları dahil olmak üzere çeşitli gerçek hayat uygulamalarının neden olduğu dolaşım satıcı problemi (*Roaming Salesman Problem, RSP*) diyoruz. RSP, geleneksel periyodik Gezgin Satıcı Problemi (*Traveling Salesman Problem, TSP*) ile Ödül Toplayan TSP'nin statik ayırıt maliyetleri ve zamana bağlı düğüm ödülleri içeren bir kombinasyonu olarak tanımlanabilir. RSP, kazanılan tüm ödüllerin toplamından seyahat masraflarını çıkararak elde edilen net faydayı enbüyüklemek için, kampanya döneminin her bir günü açık veya kapalı bir tur arar. Satıcının tüm şehirleri ziyaret etmesi veya günlük turlarının aynı şehirde başlaması ve bitmesi gerekmez. Ayrıca satıcı bir sonraki günün turuna başlamak için şehirde bir gece konaklayabilir. Her şehir bir temel ödül ve sabit bir faaliyet süresi ile ilişkilidir. Ek olarak, satıcı belirli sayıda ardışık günden daha uzun bir süre kampanya merkezinin dışında kalamaz. İlaveten aynı gün içindeki etkinliklerin ve şehirler arası seyahatlerin toplam süresi belirli bir maksimum tur süresini aşamaz.

Bu problem için mevcut rotalama kısıtlarını kabul ederek, tamamen yeni bir kısıt sınıfı ve ikili deęişkenleri içeren bir Karışık Tamsayılı .doğrusal Programlama (*Mixed-Integer Linear Programming, MILP*) formülasyonu geliştirilmiştir. RSP'nin seçim lojistięi bir uygulaması olarak, çok dönemli seyahat eden politikacı problem tanıtılmaktadır. Problem analitik model ve kapsamlı bir senaryo analizine adapte edilerek verimli bir şekilde çözülmeye çalışılmıştır. Ticari çözücüler, RSP'nin küçük boyutlu örneklerini makul bir zamanda neredeyse optimal çözüme yeteneęine sahiptir. Büyük ebattaki örneklerin üstesinden gelmek için, birinci aşamada şehir seçimiyle ilgilenirken, ikinci aşamada rota üretimine odaklanan iki aşamalı bir metasezgisel önerilmiştir. İkincisi, seçilen şehirler arasında en uygun rotayı inşa etmek için bir tamsayılı programlama modeli kullanılmıştır. Önerilen metasezgisel, RSP'yi temel olarak kampanya günlerinin sayısı kadar alt probleme ayrıştırır.

Ayrıca bu çalışmada, RSP için granül ve eğritilmiş deęişken komşuluk tabu araması (*Granular Skewed Variable Neighborhood Tabu Search, GSVNTS*) olarak adlandırılan yeni bir hibrit metasezgisel algoritma sunuyoruz. Bu metasezgisel Eğritilmiş Deęişken Komşuluk Arama algoritmasına gömülü bir Granül Tabu Araması'ndan oluşur. Önerilen yöntem, Türkiye'nin 81 ili ve 12 ilçesi dahil olmak üzere gerçek seyahat mesafeleri içeren örnek problemler üzerinde deneysel olarak test edilmiştir. Elde edilen deneysel sonuçlar, GSVNTS'nin az miktarda CPU zamanında optimal veya optimale yakın çözümlerin üretilebileceğini göstermektedir. Sunduğumuz matematiksel model, bu modelin çözümü için geliştirdiğimiz iki farklı metasezgisel ve bu

metasezgiseli bařlangıç çözümlü olarak kullanan çözümlü yöntemi GSVNTS sayesinde kampanya planlamacılarına stratejik karar vermelerinde yardımcı olunabilir.



ACKNOWLEDGMENTS

First and foremost, I would like to express my sincere gratitude and appreciation to my supervisor Assoc. Prof. Dr. Deniz Aksen for his advice, contribution, patience, and excellent support. His passion and positive attitude, made me feel profoundly privileged. He was more than just an advisor; He was a big brother for me.

I would like to thank Prof. Dr. Zeynep Akşin Karaesmen, Assist. Prof. Dr. Hande Küçükaydın, Prof. Dr. Ahmet Faruk Aysan, and Assist. Prof. Dr. Barış Yıldız for taking part in my thesis committee and for their attention and insightful feedback on this dissertation. In addition, my warm appreciation goes to Prof. Dr. Zeynep Gürhan Canlı, dean of the College of Administrative Sciences and Economics for her continuous support. I would also like to thank Didem Gürses for her attention and administrative support.

I would like to thank my friends Morteza, Sanam, Arezou, Siamak, and Başak for their valuable and warm friendships during my Ph.D. studies at Koç University.

I gratefully acknowledge Mr. Esmail Akbarpour Paydar for his generosity and support.

I would like to thank Prof. Dr. Ahmet Faruk Aysan to give me the opportunity to present my work in Istanbul City University.

My deepest gratitude is expressed to my parents Alireza and Tayyebeh, and brother Ali for always believing in my success and trusting me all my life.

Finally, I would like to thank my wife Maryam Ekhtiari for loving and taking care of me. Her continuous patience and superb support during my graduate studies have been invaluable.





Table of Contents

List of Tables	xvii
List of Figures	xix
Nomenclature	xxii
Chapter 1: Introduction	1
Chapter 2: Literature review	9
2.1 Traveling salesman problem variants	9
2.2 Mathheuristics	16
2.3 Variable Neighborhood Search	17
Chapter 3: Notation and formulation	20
3.1 Notation	20
3.2 Mixed integer linear programming formulation	22
3.3 More details on the newly-developed constraints	29
3.4 An alternative formulation for satisfying maximum tour duration constraint	33
3.5 Added valid inequalities	35
3.6 An arc-based formulation for subtour elimination.....	35
Chapter 4: An application to election logistics	38

4.1	A real case example.....	40
4.2	Data collection and analysis.....	41
4.3	Preprocessing of the symmetric time/cost matrix.....	42
4.4	Time-dependent rewards	44
4.4.1	Factor 1: Population	45
4.4.2	Factor 2: Ratio of votes and Criticality Factor	45
4.4.3	Factor 3: Number of remaining days until the election day.....	49
4.4.4	Factor 4: Number of days passed since the previous meeting	50
4.5	Supplementary assumptions in MPTPP	50
Chapter 5:	Scenario Analysis.....	52
5.1	Scenario analysis level 1: Extreme scenarios	52
5.1.1	Scenario 1: Model Full-MILP (Base Scenario).....	53
5.1.2	Scenario 2: Model Full-1Meet.....	55
5.1.3	Scenario 3: Model Rew-Only	56
5.1.4	Scenario 4: Model Alt-1Depot	56
5.2	Scenario analysis level 2: Alternative reward function	58
Chapter 6:	The proposed two-phase matheuristic	59
6.1	Mathematical formulation of FDORM1 and FDORM2	65
6.2	City selection approaches	69

6.2.1	Deterministic City Selection.....	70
6.2.2	Greedy City Selection	70
6.2.3	Pseudo-random city selection	71
Chapter 7: A Granular Skewed Variable Neighborhood tabu search .		75
7.1	Variable Neighborhood Search	76
7.2	Solution Representation.....	80
7.3	Initial solution.....	81
7.3.1	Exhaustive search of the candidate cities	81
7.4	Neighborhood structures.....	84
7.5	Granular neighborhoods	92
7.6	Shaking.....	96
7.7	Local Search	97
7.8	Penalty value for strategic oscillation	101
7.9	Skewed moves.....	104
7.10	Termination criteria	105
7.11	Granular Skewed Variable Neighborhood Tabu Search	106
Chapter 8: Computational Results		110
8.1	Data sets	110
8.2	Computational Platform and Solver specifications.....	111

8.2.1	Cuncurrent MIP configuration	113
8.3	Comparison of original formulation with alternative formulation.....	114
8.4	Speeding up GUROBI using the results of FDOR	116
8.5	Linear relaxation of the binary decision variables	118
8.6	Effect of the added valid inequalities (VI) on solution quality	119
8.7	Changing values of α	121
8.8	Computational results of scenario analysis level 1	122
8.9	Computational results of scenario analysis level 2	124
8.10	Tightening scheme.....	126
8.11	Omitting binary decision variables and parameters.....	127
8.12	Performance of FDOR–DCS.....	129
8.13	Performance of FDOR–GCS.....	131
8.14	Performance of FDOR–PCS	133
8.15	Comparison of performances of FDOR–DCS, FDOR–GCS, and FDOR–PCS.....	136
8.16	Comparison of GUROBI and FDOR–DCS for all instances.....	140
8.17	Results for GSVNTS	143
8.18	Performance of GSVNTS	146
8.19	Comparison of GSVNTS with Party’s actual meeting plan.....	149
9.	Conclusions and future work	152

Bibliography 155

Appendix A 167

Appendix B 173



LIST OF TABLES

Table 4.1 Statistics of rewards in criticality categories	48
Table 6.1 Comparison of the results of original MPTPP and Semi-Solved Scenario.....	60
Table 6.2 Reduction of original MILP formulation by using FDORM1 and FDORM2	68
Table 7.1 Termination criteria of GSVNTS	106
Table 8.1 List of GUROBI specific options applied to all runs.	112
Table 8.2 Comparison of Cplex with different CONCURRENTMIP configurations of Gurobi.....	114
Table 8.3 Comparison of two MTD formulation	115
Table 8.4 The results of setting initial values of MPTPP to optimal values of FDOR	117
Table 8.5 Comparison of different solutions of the same instance with 39 cities and 15 days.....	119
Table 8.6 Comparison of the models with and without valid inequalities ...	120
Table 8.7 Results of the instances for different values of α	121

Table 8.8 Results of the four scenarios.....	124
Table 8.9 Comparison of new and original reward function.....	125
Table 8.10 Comparison of the original reward function with the reward function without binary variable FM and Z	128
Table 8.11 Computational Results of FDOR–DCS ($\lambda = n$)	130
Table 8.12 Computational Results of FDOR–GCS	134
Table 8.13 Computational Results of FDOR–PCS	135
Table 8.14. Comparison of FDOR–DCS, FDOR–GCS, and FDOR–PCS..	137
Table 8.15 Comparison of routes of FDOR–DCS and optimal solution.....	139
Table 8.16 Comparison of GUROBI with FDOR–DCS for all instances.....	141
Table 8.17 Comparison of GUROBI with GSVNTS for all instance sizes...	144
Table 8.18 Comparison of GSVNTS with Party’s actual meeting plan.....	151

LIST OF FIGURES

Figure 1.1	Type 1 tour.....	4
Figure 1.2	Type 2 tour.....	5
Figure 1.3	Type 3 tour.....	5
Figure 1.4	An instance with both closed and open tours	6
Figure 3.1	Type 1 tours	29
Figure 3.2	Type 3 tours	31
Figure 4.1	A simple example presented to illustrate the problem structure.	40
Figure 4.2	Population of 81 provinces in Turkey	46
Figure 4.3	Base rewards (π_i values) of all 81 provinces.	49
Figure 5.1	Different tours in the first three scenarios.	57
Figure 7.1	Steps of the Basic VNS (Hansen and Mladenović, 2001)	78
Figure 7.2	Basic VNS scheme (Hansen et al. 2010).....	79
Figure 7.3	The 1-Add operator	85

Figure 7.4 The 1-Drop operator	86
Figure 7.5 The Drop Add operator	87
Figure 7.6 The 1-1 Exchange Non-Visited operator	87
Figure 7.7 The 1-1 Exchange Intra Route operator	88
Figure 7.8 The 1-0 Relocate operator	89
Figure 7.9 The 2-0 Relocate operator	89
Figure 7.10. The 1-1 Swap operator	90
Figure 7.11 The 2-2 Swap operator	90
Figure 7.12 The Triple Rotation operator	91
Figure 7.13 The Quadruple Rotation operator	92
Figure 7.14 An example of granular neighborhoods	96
Figure 8.1 Procedure for tightening the optimal value of main problem	126
Figure 8.2 Comparison of FDOR–DCS, FDOR–GCS, and FDOR–PCS ...	138
Figure 8.3 The comparison of the net benefit of FDOR and MPTPP	140
Figure 8.4 Performance of GSVNTS (PE.I)	146

Figure 8.5 Average runtime of GSVNTS (PE.I)	147
Figure 8.6 Performance of GSVNTS (PE.II).....	147
Figure 8.7 Average runtime of GSVNTS (PE.II)	148
Figure 8.8 Performance of GSVNTS (LE)	148
Figure 8.9 Average runtime of GSVNTS (LE).....	149

NOMENCLATURE

ACO	Ant Colony Optimization
ATSP	Asymmetric Traveling Salesman Problem
BFS	Best Feasible Solution
BRP	Bank Robber Problem
CDF	Cumulative Density Function
CF	Criticality Factor
CPTP	Capacitated Profitable Tour Problem
CPU	Central Processing Unit
DCS	Deterministic City Selection
ESCC	Exhaustive Search of the Candidate Cities for Each Day
FDOR	Finding Daily Optimal Routes
FDORM	Finding Daily Optimal Routes Model
GCS	Greedy City Selection
GLS	Guided Local Search
GSVNTS	Granular Skewed Variable Neighborhood Tabu Search
HRC	Highest Reward Cities
KP	Knapsack Problem
L-MTZ	Lifted Miller-Tucker-Zemlin

LE	Local Election
LR	Linear Relaxation
LRP	Location Routing Problem
MCP	Maximum Collection Problem
MILP	Mixed-Integer Linear Programming
ML-MTZ	Modified Lifted Miller-Tucker-Zemlin
MPTPP	Multi-Period Traveling Politician Problem
MTD	Maximum Tour Duration
MTMCP	Multiple Tour Maximum Collection Problem
MTZ	Miller-Tucker-Zemlin
MuPOPTW	Multi-Period Orienteering Problem with Multiple Time Windows
OP	Orienteering Problem
OVRP	Open Vehicle Routing Problem
PCS	Pseudo-Random City Selection
PCTSP	Prize Collecting Traveling Salesman Problem
PE	Presidential Elections
PLR	Partial Linear Relaxation
PP	Politician's Party
PTP	Profitable Tour Problem

PTSP	Periodic Traveling Salesman Problem
QTSP	Quota Traveling Salesman Problem
RAM	Random Access Memory
RSP	Roaming Salesman Problem
RVNS	Reduced Variable Neighborhood Search
SCP	Simple Cycle Problem
SFLR	Semi-Full Linear Relaxation
SPCTSP	Selective Prize Collecting Travelling Salesman Problem
STSP	Selective Traveling Salesman Problem
SVNS	Skewed Variable Neighborhood Search
SVRPTW	Selective Vehicle Routing Problem with Time Windows
TOP	Team Orienteering Problem
TS	Tabu Search
TSP	Traveling Salesman Problem
TSPP	Traveling Salesman Problem with Profits
TuOP	Tour Orienteering Problem
VI	Valid Inequalities
VND	Variable Neighborhood Descent
VNDS	Variable Neighborhood Decomposition Search
VNS	Variable Neighborhood Search

VRP	Vehicle Routing Problems
VRPB	Vehicle Routing Problem with backhauls
VRPTW	Vehicle Routing Problem with Time Windows



Chapter 1

INTRODUCTION

In this study, we study a logistical problem arising in promotion and marketing campaigns where the campaigner and his/her team needs to plan an efficient schedule through the campaign period to maximize the total reward by visiting appropriate cities. This logistical problem has a wider range of applications including election logistics, touristic trip planning, marketing campaign planning, promotion of a new product launch, and planning of client visits by company representatives, among others. We refer to this new problem as the *roaming salesman problem* (RSP).

RSP has some similarities with a multi-period extension of the prize collecting traveling salesman problem (PCTSP) with time-dependent rewards and multiple visits. It is however very important to stress that the RSP does not involve a fixed central node designated as ‘the depot’. This implies that a daily tour may or may not terminate at the same node where it has started. In other words, the tour does not have to be a round-trip plus a node may be visited either in transit or for the purpose of an activity. In the latter case, the campaigner (i.e. the roaming salesman) spends a node-specific activity time and consequently collects a reward associated with that node. There are two types of rewards, namely a base reward and a depreciated reward. The base reward has been defined a priori for each node according to its characteristics.

However, the actual reward that can be claimed is subject to twofold depreciation:

- (a) It decreases linearly with the activity date. The later the activity, the smaller the collectible reward.
- (b) It decreases linearly with the recency of the past activity in the same node. The lesser the number of days that passed since the previous activity, the smaller the reward.

Especially nodes of exorbitant size or significance may observe multiple visits. The second type of reward depreciation circumvents successive activities in such nodes within short time intervals.

The goal in the RSP is to find an optimal or the ‘best’ schedule of daily visits for a campaigner who seeks to maximize his/her net benefit throughout a given number of periods (days). The net benefit is defined as the sum of all collected rewards minus the traveling costs incurred by the salesman. The RSP can be therefore classified as a rich *traveling salesman problem* (TSP) with the following six properties which together make this problem rather unique. For an overview of rich routing problems, see Lahyani et al. (2015).

- (i) *Multi-period*. RSP generalizes the TSP by extending the planning horizon to n days, thereby forming a multi-period problem.
- (ii) *Time-constrained*. In each period, i.e. on each day, the salesman is allowed to “roam” for no more than a certain number of hours. We refer to this time limit as the maximum tour duration constraint.
- (iii) *Selective*. The salesman needs to decide which nodes to visit so as to realize an activity. In other words, not every node is visited and not every node hosts an activity.

- (iv) *Absence of a fixed depot node, co-existence of open and closed tours.* Tours do not have to start and end at the same node. The only requirement is that today's tour originates where yesterday's tour terminated. Hence, the salesman has also to decide where to stay overnight at the end of each day.
- (v) *Time-dependent rewards.* Each node is associated with a time-dependent reward which changes linearly according to the day of the hosted activity in that node and the recency of the previous activity in the same node. This is a challenging issue which is mainly attributed to this problem.
- (vi) *Multiple visits.* There exist a subset of nodes which may host more than one activity during the campaign, hence can be visited more than once.

The RSP can be defined as follows. Consider a set of cities $\mathbf{N} = \{0, 1, \dots, n\}$ including a fictitious city (indexed 0), a set of cities $\mathbf{V} = \{1, \dots, n\}$ including a starting city (indexed as 1) and a set of days $\mathbf{T} = \{1, \dots, \tau\}$. On each day $t \in \mathbf{T}$, any city $i \in \mathbf{V}$ can be visited either to collect the associated reward from it or while in transit without reward. A nonnegative prize of π_i called *base reward* is specified for each city $i \in \mathbf{V}$. The base reward can depend on several factors such as the city population. Moreover, the actual reward earned by having an activity in city i on day t depends on two other factors:

- Factor 1. The number of remaining days denoted by $(\tau - t)$ until the end of the campaign, i.e. until the election day.
- Factor 2. In case a city hosts more than one activity, the number of days passed since the previous activity in the same city, denoted by s where $1 \leq s \leq t - 1$.

The traveling cost between each pair of cities is known and given by c_{ij} , $\forall i, j \in \mathbf{V}$ where c_{ij} denotes the cost of driving (or flying where applicable) from city i to city j . The traveling time between each pair of cities is also known with certainty and given by d_{ij} , $\forall i, j \in \mathbf{V}$. The time spent by the salesman (also referred to as the *campaigner* in the sequel) for an activity in city $i \in \mathbf{V}$ is shown by σ_i , $i \in \mathbf{V}$. The maximum tour duration applicable to the tour of each day is denoted by T_{\max} . This time limit imposes an implicit threshold on the number of cities that can be visited in any given day.

There is also an explicit limit α on the number of activities that can be realized per day. For the fictitious city $i = 0$, the activity duration, the base reward, the traveling costs and times are all set to zero. The campaign period starts in the central city $i = 1$ in the morning of day $t = 1$ and ends in the evening of day $t = \tau$. At the end of a day $t \in \mathbf{T}$, the campaigner stays overnight in some city $i \in \mathbf{V}$. Note that waking up or staying overnight in city i does not necessarily mean that there will be a reward collection in that city. One final remark should be made about periodic returns to the campaign base $i = 1$. The salesman cannot be away from the campaign base for more than κ consecutive days.

A distinctive feature of the RSP is that there are three possible types of daily tours during the campaign.

Type 1 tours: Multi-city closed tour

The campaigner starts the day (wakes up) in city i on day t , leaves i and visits at least one more city scheduled for that day. At the end of the day, he returns to the same city i to stay

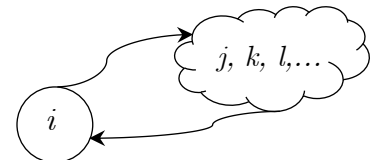


Figure 1.1 Type 1 tour

overnight. Type 1 tour is a closed tour starting and ending at city i , and involving at least one more city other than city i (see Figure 1.1).

Type 2 tours: Single-city tour

The campaigner wakes up in city i on day t , spends the whole day in the same city collecting the reward and stays overnight in the same city. In Type 2 tour, we assume that the campaigner goes from city i to a fictitious city denoted by ‘0’ and returns from ‘0’ to i . This tour is therefore treated as a closed tour starting and ending at city i (see Figure 1.2).

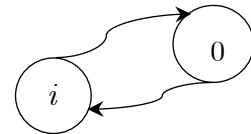


Figure 1.2 Type 2 tour

Type 3 tours: Multi-city open tour

The campaigner wakes up in city i on day t , and goes to another city j . In between cities i and j , he may visit one or more cities, or may directly travel from i to j where he stays overnight. Type 3 tour is an open tour starting in city i and ending in city j , as shown in Figure 1.3.



Figure 1.3 Type 3 tour

In order to highlight the importance of having both open and closed routes during the campaign, we build a toy instance containing six cities, two days, and a daily maximum tour duration of 14 hours as illustrated in Figure 1.4. The travel times in *italic* and the activity times are written on the arcs and next to the nodes, respectively, both in hours. As shown in Figure 1.4, the tour of the first day starts in city i , and includes three activities in cities i , j , and k . The campaigner returns to the starting city i at the end of day 1 without

having any more activities there. The return to city i on day 1 grants him/her enough time to visit more than one far city (m and n) the next day.

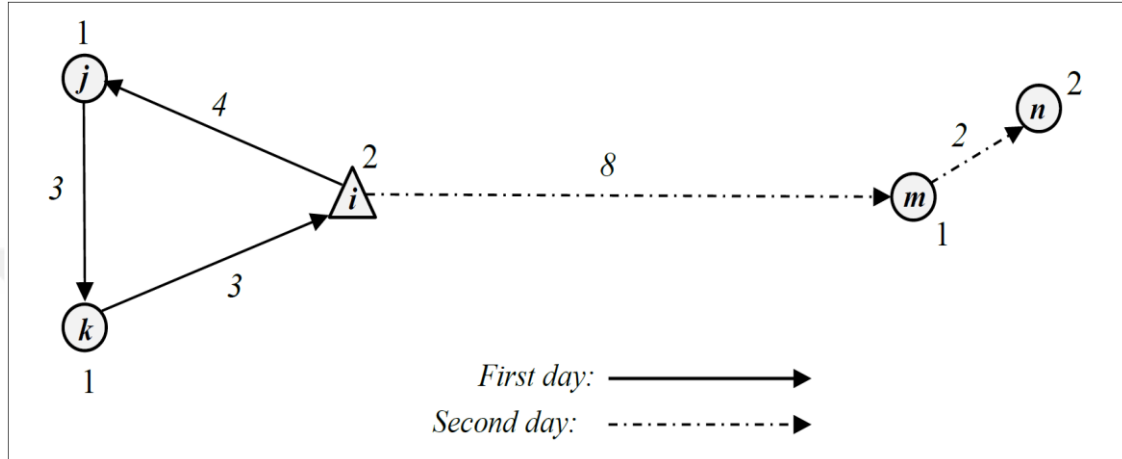


Figure 1.4 An instance with both closed and open tours

By ignoring the meeting times σ_i , taking the campaign duration as $\tau = 1$, and setting the maximum tour duration and each city's base reward to sufficiently large values, e.g. $\sigma_i = \max_{j \in \mathbf{N}} \{d_{ij}\} + \max_{j \in \mathbf{N}} \{d_{ji}\}$ and $T_{\max} = n \max_{i,j \in \mathbf{N}} \{d_{ij}\}$, a given TSP instance can be reduced to the associated MPTPP instance in polynomial time. TSP is a well-known \mathcal{NP} -hard combinatorial optimization problem (Garey and Johnson, 1979). This leads to the conclusion that the MPTPP is also \mathcal{NP} -hard, and thus cannot be solved in polynomial time to optimality.

Motivated by this challenge, we propose for the RSP a simple but efficient two-phase matheuristic method which we call *Finding Daily Optimal Routes* (FDOR). For each period of the planning horizon, FDOR decomposes the RSP

into a pair of subproblems, namely a city selection problem in Phase I and a modified prize-collecting TSP to be solved optimally in Phase II. We experimented with three different city selection approaches so as to arrive at an effective, yet efficient selection scheme. Our proposed metaheuristic can provide for medium- and large-size RSP instances a promising bundle of accommodation and meeting schedules that are complemented by daily routing plans. Actually, FDOR achieves this in remarkably short solution times. This way it can greatly facilitate campaign planners' decision-making in their election logistics efforts.

To improve the solution of FDOR, we adapt the basic variable neighborhood search (VNS) algorithm to RSP. We propose another solution methodology to address RSP and its application in election logistics. We call this method *granular skewed variable neighborhood tabu search* (GSVNTS). VNS is a recently developed metaheuristic for tackling combinatorial optimization problems. The main idea is to change neighborhoods within local search in a systematic way (Hansen and Mladenović, 2001). VNS metaheuristics have been successfully implemented in different combinatorial optimization problems and their real-life applications, e.g., the vehicle routing, the traveling salesman, and the p -median (García-López et al. 2002). We modify the basic VNS by incorporating the granularity and the skewness in it. FDOR constructs the initial feasible solution. In the local search step, a tabu search heuristic is used to avoid cycles. We also consider a variety of rich neighborhood structure to explore as many solutions as possible. The detailed descriptions of FDOR and GSVNTS are given in Section 5 and Section 6.

To the best of our knowledge, this is the first time the RSP is explored in depth and tackled. Our contribution is fivefold:

1. The investigation of a new logistical problem arising in several areas including election logistics.
2. The development of a novel mixed-integer linear programming (MILP) formulation for this new problem.
3. A real-life application of the problem to election logistics covering 81 provinces and 12 highly populated towns of Turkey, supported by an extensive scenario analysis.
4. The development of a new two-phase metaheuristic to solve large-size instances.
5. The development of a granular skewed variable neighborhood search complemented with a tabu search to find optimal or near to optimal solutions for RSP.

The remainder of this thesis is organized as follows. In Section 2 we review the related literature. In Section 3, we present the mathematical formulation of RSP. In Section 4 we investigate the application of RSP in election logistics. The scenario analysis is discussed in Section 5. In Section 6 we present the proposed two-phase metaheuristic approach FDOR. The proposed metaheuristic is presented in Section 7. We discuss our computational results in Section 8 on the basis of a case study involving a lot of cities and towns from Turkey. Finally, Section 9 summarizes our results and recommends future research directions.

Chapter 2

LITERATURE REVIEW

In this chapter, we survey the studies related with roaming salesman problem where we point out the differences between our problems and the related studies. Moreover, we survey the matheuristics introduced to similar problems. Furthermore, we address the studies which utilize VNS as a solution methodology.

2.1 Traveling salesman problem variants

RSP has been introduced to the literature recently by Shahmanzari et al. (2018). It is derived from the well-known *traveling salesman problem* (TSP) which is probably the most famous and oldest \mathcal{NP} -hard combinatorial optimization problem in the literature. A widely accepted and often cited classification of TSP and its variants has been presented in Gutin and Punnen (2007).

The first TSP variant that is closely related to RSP is the *periodic traveling salesman problem* (PTSP). Many variations of TSP assume that traveling occurs in one period only. However, PTSP relaxes this assumption by expanding the travel period to m days such that each city is visited at least once, while some cities require multiple visits. There is only one salesman available every day. The goal is to generate a tour for each of m days that will

meet the visit frequency of each city and minimize the total traveling distance throughout the whole planning horizon. The first mathematical formulation of PTSP can be found in Cordeau et al. (1997).

The other TSP variants resembling RSP include *the prize collecting traveling salesman problem* (PCTSP), *the profitable tour problem* (PTP), and *the orienteering problem* (OP). We briefly describe these three variants here. They are jointly referred to as the generic class of *TSP with profits* (TSPP). Problems belonging to TSPP class have been surveyed systematically in the seminal paper by Feillet et al. (2005) where the name TSPP was coined for the first time. The authors stated that TSPPs arise in a wide range of situations, including realistic TSPs, job scheduling, freight transportation, or they occur indirectly as a subproblem in solution approaches dedicated to other routing problems. TSPP is by definition the monocriterion version of a bicriteria extension of TSP where the two criteria are the maximization of a profit measure and the minimization of travel costs. The basic characteristics of this generic problem are as follows.

- (i). There is a value (like a profit or prize) associated with each vertex of the underlying graph.
- (ii). A feasible solution is not required to visit all vertices.
- (iii). A vertex can be visited at most once.
- (iv). The distance (cost) matrix is nonnegative and satisfies the triangle inequality.

Being the first member of this class, PCTSP was originally introduced by Balas and Martin (1985) and formally defined in Balas (1989) to model the scheduling of the daily operations of a steel rolling mill. In PCTSP there is a traveling salesman who travels between vertices i and j at cost c_{ij} , earns a prize p_k from every visited vertex k and pays a penalty γ_h for each unvisited vertex

h. The aim is to find a circuit, i.e. a tour that minimizes the sum of travel costs and penalties while collecting a total profit at least as high as a preset minimum value π_{\min} . Here, a feasible circuit either in PCTSP or in the other TSPP variants visits each vertex at most once. The minimum profit collection constraint can be viewed as a knapsack-like constraint. Feillet et al. (2005) note that the majority of PCTSP papers deal with problems which have zero penalty terms.

Another name coined for PCTSP is the *quota TSP* (QTSP) which was first studied in Awerbuch et al. (1998). Structural properties of PCTSP related to TSP polytope and to the knapsack polytope were presented by Balas (1989, 1995) where families of facet-inducing inequalities were identified. Bounding procedures based on different relaxations were developed by Fischetti and Toth (1988) and Dell'Amico et al. (1995). The lower bound obtained according to the latter paper was used in a follow-up study by Dell'Amico et al. (1998) as the starting point of a Lagrangian heuristic which is capable of finding a feasible upper bound to the problem. A branch-and-cut algorithm was proposed for the undirected PCTSP in Bérubé et al. (2009). The authors adapted and implemented some classical polyhedral results for PCTSP and derived inequalities from cuts designed earlier for OP.

The second member of the generic TSPP class is PTP. It derives directly from the PCTSP when the objective becomes the maximization of the net profit defined as the difference between the collected prizes and the travel costs. In the presence of nonzero penalties for unvisited vertices, the sum of incurred penalties is also deducted from the total amount of collected prizes to yield the net profit. PTP was introduced first by Dell'Amico et al. (1995). Fischetti et al. (2007) called the same problem the *simple cycle problem* (SCP). Archetti et al. (2009) formulated a multi-tour version of the PTP with multiple identical

and capacitated vehicles, which they referred to as the *capacitated PTP* (CPTP).

The third problem in TSPP class is OP which is evidently the most extensively studied variant. OP seeks to find a circuit or a path on a graph with n vertices that maximizes the sum of collected prizes while containing traveling costs under a preset value C_{\min} or the total travel time within a preset limit T_{\max} . Vansteenwegen et al. (2011) argue that OP can be viewed in this regard as a combination between the *knapsack problem* (KP) and TSP. Feillet et al. (2005) point to the equivalence between the path-seeking and circuit-seeking versions of the problem. Mansini et al. (2006) designate the circuit-seeking version as the *tour orienteering problem* (TuOP).

As mentioned in Chao et al. (1996b), the name “*orienteering problem*” originates from the treasure hunt game of orienteering in which individual competitors start at an initial control point, try to visit as many checkpoints as possible and return to the control point within a given time frame. Each checkpoint has its own reward and the objective is to maximize the collected rewards. Pioneering studies of OP can be found in Hayes and Norman (1984), Tsiligirides (1984), Golden et al. (1987) and Golden et al. (1988) among others. OP was researched in the literature also under different titles such as the *selective TSP* (STSP) (see Laporte and Martello, 1990; Gendreau et al., 1998; Thomadsen and Stidsen, 2003), the *maximum collection problem* (MCP) (see Kataoka and Morito, 1988; Butt and Cavalier, 1994) and the *bank robber problem* (BRP) (see Arkin et al., 1998). OP was shown to be \mathcal{NP} -hard by Golden et al. (1987) and by Laporte and Martello (1990) with separate proofs based on simple reductions to TSP and to the Hamiltonian circuit problem, respectively.

Applications in the literature of this selective routing problem span a wide range of areas. Several examples are orienteering competitions, routing technicians to service customers at geographically distributed locations, time-restricted fuel delivery to households with different urgency scores, athlete recruiting from high schools for a college team, pickup or delivery services with private fleets requiring the selection of only a subset of available customers, trip planning for tourists visiting a city or a region, and fish scouting where a subset of fishing grounds are sampled to maximize the value of catch rate assessments.

A well-studied multi-vehicle extension of OP is the *team orienteering problem* (TOP). The problem first appeared in a paper by Butt and Cavalier (1994) under the name *multiple tour maximum collection problem* (MTMCP) where all tours have an identical starting and terminal node designated as the depot. The authors proposed a greedy construction heuristic for its solution. The first exact solution method developed to tackle TOP is due to Butt and Ryan (1999). The authors were able to solve problems with up to 100 vertices using a column generation-based procedure when the number of vertices in each tour remains relatively small. Another exact algorithm is due to Boussier et al. (2007). The authors proposed a branch-and-price scheme that starts with column generation and couples it with branch-and-bound. For performance enhancement, they applied a heuristic tree-search method derived from constraint programming and different pre-processing rules that can be interpreted as branching rules specifically adapted to the problem. Boussier et al.'s exact algorithm is capable of solving TOP instances with 100 vertices in under two hours where each vehicle selects up to 15 vertices to visit. The authors modified their exact solution technique so as to solve also to the *selective vehicle routing problem with time windows* (SVRPTW).

The first heuristic method for TOP was proposed by Chao et al. (1996a). The first metaheuristics were proposed by Tang and Miller-Hooks (2005) and Archetti et al. (2007). The former authors developed a tabu search heuristic embedded in an adaptive memory procedure; the latter developed two variants of a tabu search heuristic (one using only feasible solutions, the other one accepting also infeasible solutions) and a pair of slow and fast variable neighborhood search (VNS) methods. These pioneering metaheuristics were soon followed by Ke et al. (2008)'s ant colony optimization (ACO) approach which was capable of outperforming the previous algorithms in both solution speed and accuracy. The speed factor of the state of the art of the TOP has been improved dramatically by the guided local search (GLS) framework of Vansteenwegen et al. (2009a) and later by the skewed VNS framework of Vansteenwegen et al. (2009b) both of which relied on a combination of intensification and diversification procedures. A number of variants of the OP and TOP have been introduced to the routing literature in the past decade. They address additional diverse aspects ranging from time windows to time-dependent or stochastic travel times to capacitated vehicles to stochastic profits and to a combination thereof. The reader is referred to Gunawan et al. (2016) for an inclusive review of the studies of the OP, its extensions and applications that were published after 2009.

Within the generic class of TSPPs, the variant that seems most relevant and similar to our MPTPP is the multi-period OP with multiple time windows (MuPOPTW) introduced by Tricoire et al. (2010) for a real-world sales representative planning problem. A software distribution company which sells decision support systems for salesman and marketing departments needs to plan the visits to existing and potential customers by each representative over a one-week period. There is a list of mandatory customers who should be visited

on a regular basis and another list of optional customers located nearby who should be also considered and probably integrated into the schedules of the sales representatives. The authors solve MuPOPTW for a given representative with the aim of determining which of the mandatory and optional customers to visit on which day. Some of the customers have one or two time windows per day which restrict the timing of the visit, and there exist even a few customers who have a different time window for every day.

MuPOPTW in Tricoire et al. (2010) resembles our MPTPP in that each day of the planning horizon is associated with a separate tour. MPTPP differs from MuPOPTW considerably due to the following aspects:

- (a) In MuPOPTW the tour of each day starts and ends at the same central node. The mathematical model proposed by the authors can handle also the case where the representative makes a several-day trip across the country and stops every night in previously fixed hotels such that the ending point for day t matches the same location as the starting point for day $t + 1$. However, even in that case, the terminal node (i.e. the depot) of each tour is known in advance. In contrast, in the RSP this is unknown.
- (b) In MuPOPTW, a customer node is visited at most once whereas RSP allows certain nodes to be visited more than once.
- (c) Moreover, rewards collected from customer nodes in MuPOPTW do not change over time while in RSP their magnitude depends on the day and frequency of the visit.

2.2 Matheuristics

Recent progress in CPU technologies and commercial solvers results in solving different mixed integer linear programming models to optimality or near to optimality in a small amount of CPU time. This leads to design matheuristic approaches, a heuristic that incorporate stages where mathematical programming models are solved.

In the literature, there are a couple of articles that use matheuristic methods in order to solve TSP variants. A unified matheuristic approach based on the variable neighborhood search is proposed by Lahyani et al. (2017) for solving multi-constrained traveling salesman problems with profits. Their method combines different removal and insertion routing neighborhoods. In Prins et al. (2007) a matheuristic approach is proposed where the original problem is decomposed into two phases; location decisions and routing. The location decision problem is solved as a facility location problem. Then a tabu search builds the routes using the given facility sets.

Halvorsen-Weare and Fagerholt (2013) propose a routing and scheduling problem emerging in naval logistics. Their method separates the scheduling decisions from the routing decisions. The routing problem is solved through a local search heuristic and the scheduling problem is solved through the exact solution of a MILP formulation. Cacchiani et al. (2014) propose a two-phase matheuristic approach for the problem of determining the size of the waste bins located on the streets and planning the daily routes of waste collector vehicles. They propose a different solution method where a variable neighborhood search heuristic finds the daily route and an MILP model solves the problem of determining the optimal size of the waste bins. A review of different heuristic methods including matheuristics can be found in Salhi, (2017).

2.3 Variable Neighborhood Search

The variable neighborhood search (VNS) is introduced by Mladenović and Hansen (1997) for solving large instances of combinatorial optimization problems. It represents a dynamic framework for constructing heuristics by systematically changing neighborhood structures during the search procedure. In other heuristic methods such as simulated annealing, genetic algorithm, and tabu search, specific strategies are considered to escape the local optimum in the search space. This is in VNS done by changing the neighborhood structures repeatedly. VNS lie in three simple facts: (i) A local optimum with respect to one neighborhood structure is not necessarily a local optimum with respect to another neighborhood structure. (ii) A global optimum is a local optimum with respect to all neighborhood structures. (iii) Based on many empirical pieces of evidence, it is shown that a large majority of the local optima are slightly close to each other local optima for many problems (Kirkpatrick and Toulouse, 1985).

In the literature, VNS is used for several TSP variants. The first VNS with its basic implementation for Euclidean TSP can be found in Hansen and Mladenović (1999). Guided VNS method is introduced by Burke et al. (2001) for the Asymmetric TSP (ATSP). Carrabs et al. (2007) apply VNS for the Pickup and Delivery TSP. The influence of the neighborhood structures for VNS method in Generalized TSP is studied by Hu and Raidl (2008). Mansini and Tocchella (2009) develop a multi-start VNS for the traveling purchaser problem under budget constraints.


VNS is also used to solve the standard versions of the vehicle routing problems (VRP). Crispim and Brandao (2001) solve the VRP with backhauls (VRPB) using a variable neighborhood descent (VND). It is a variant of VNS

where exploration of different neighborhood structures is performed in a deterministic way. Its benefit is again based on the fact that various neighborhood structures do not have the same local minimum in most cases. Therefore, the problem of trapping in the local optima can be resolved by changing the neighborhoods in a deterministic way. Rousseau et al. (2002) apply a VND by taking advantage of various neighborhood structures to VRP. Bräysy (2003) develop a reactive VNS for the VRP with time windows. Polacek et al. (2004) propose a VNS for the multi depot vehicle routing problem (MDVRP) with time windows.

In many applications, VNS performs better when it is combined with another metaheuristic. Melechovsky et al. (2005) merge VNS with Tabu Search (VNTS) for the location-routing problem (LRP). Repoussis et al. (2006) apply a greedy randomized VNTS for the VRP with time windows (VRPTW). An effective VNS is proposed by Kytöjoki et al. (2007) for large-scale instances of VRP. Geiger and Wenger (2007) propose VNS complemented with an interactive resolution method for VRP. Fleszar et al. (2009) develop a VNS method for the open vehicle routing problem (OVRP). Liu and Chung (2009) propose a VNTS for VRPB with inventory. Polat et al. (2015) propose a perturbation-based VNS for solving VRP with simultaneous pickup and delivery with a time limit. Todosijevic et al. (2017) propose a general VNS for the swap-body vehicle routing problem. A VNS algorithm for production routing problems is introduced by Qiu et al. (2018).

In the literature, there are a couple of extensions for VNS. The first extension is the reduced VNS (RVNS). The main goal in RVNS is to reduce the calculation time of the local search step by selecting random neighborhood structures and updating them, if a better solution is found. The RVNS is shown to be significantly efficient when a quick solution is needed, regardless of its

distance from the global optimum (Hansen et al. 2010). Another extension of VNS is the variable neighborhood decomposition search (VNDS). It includes a sequential approximation method where the local search procedure is not applied within the whole solution space (Hansen and Mladenović, 2001). The skewed VNS (SVNS) method is another extension of VNS. It addresses the problem of exploring valleys far from the incumbent solution. (Hansen and Mladenović, 2001) In fact, SVNS enhances the exploration of faraway valleys by modifying the objective function value with an evaluation function.



Chapter 3

NOTATION AND FORMULATION

In this section, we study the mathematical formulation of the Roaming Salesman Problem (RSP). The RSP described in Section 1 can be formulated as a mixed integer linear program. We first provide the notation followed by the formulation and the explanation of the new constraints which we devised.

3.1 Notation

Index Sets:

$\mathbf{N} = \{0, \dots, n\}$ Set \mathbf{V} joined by city '0' which denotes a fictitious city with all associated costs, rewards and meeting duration being zero.

$\mathbf{V} = \mathbf{N} \setminus \{0\}$ The set of cities to be considered for collecting rewards throughout the campaign period where city $i = 1$ denotes the campaign base.

$\mathbf{T} = \{1, \dots, \tau\}$ The set of τ days comprising the campaign period.

Parameters:

c_{ij} Traveling cost from city i to j where $c_{ii} = 0$.

d_{ij}	Traveling time from city i to city j where $d_{ii} = 0$.
π_i	The base reward of city i .
σ_i	The activity duration in city i .
α	Maximum number of activities allowed each day.
T_{\max}	Maximum tour duration (in hours) in each daily tour.
κ	Maximum number of consecutive days during which the campaigner is allowed to be away from the campaign base.
K	The base reward depreciation coefficient (factor) applied in successive activities held in the same city.
\bar{K}	Normalization coefficient multiplied with the collected rewards to make traveling costs and daily rewards compatible.

Decision Variables:

X_{ijt}	Binary variable indicating if arc (i, j) is traversed on day t ($i, j \in \mathbf{N}$, $t \in \mathbf{T}$) with $X_{iit} = 0$.
L_{it}	Binary variable indicating if the campaigner does not enter, but only leaves city i in day t . If $L_{it} = 1$, then the campaigner departs from city i on day t and does not come back. This indicates that the tour on day t is Type 3 with i as the starting city (source) of the tour.
E_{it}	Binary variable indicating if the campaigner does not leave, but only enters city i in day t .

If $E_{it} = 1$, then the campaigner enters city i on day t and does not leave again. This means the tour on day t is Type 3 with i being the ending city (terminal) of the tour.

S_{it} Binary variable indicating if the campaigner stays overnight (sleeps) in city i by the end of day t . Note that $S_{10} = 1$ since the campaign starts in the base city ‘1’.

Z_{it} Binary variable indicating if the campaigner holds an activity in city i on day t and collects the associated reward.

FM_{it} Binary variable indicating if the first activity in city i is performed on day t .

R_{its} Binary variable indicating if city i accommodates two consecutive activities on day t and day $(t - s)$ with no other activity in between. Since $1 \leq s < t$, we have $R_{its} = 0$ for $t \leq s \leq \tau$.

U_{it} A continuous nonnegative variable used in the Modified Lifted Miller-Tucker-Zemlin subtour elimination constraints (referred to as ML-MTZ inequalities). It is used to determine the order of visit for city i on day t .

3.2 Mixed integer linear programming formulation

The RSP can be formulated as follows:

$$\begin{aligned}
\max. \text{ NET BENEFIT} = & \\
& \sum_{i \in \mathbf{N}} \sum_{t \in \mathbf{T}} \pi_i \frac{\tau - t + 1}{\tau} FM_{it} + \sum_{i \in \mathbf{N}} \sum_{t \in \mathbf{T}} \sum_{1 \leq s < t} \pi_i \frac{\tau - t + 1}{\tau} \times \frac{s}{K\tau} R_{its} \\
& - \bar{K} \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} \sum_{t \in \mathbf{T}} c_{ij} X_{ijt}
\end{aligned} \tag{3.1}$$

Subject to:

$$\sum_{j \in \mathbf{N}} X_{ijt} \leq 1 \quad i \in \mathbf{N}, t \in \mathbf{T} \tag{3.2}$$

$$\sum_{j \in \mathbf{N}} X_{jit} \leq 1 \quad i \in \mathbf{N}, t \in \mathbf{T} \tag{3.3}$$

$$\sum_{i \in \mathbf{V}} Z_{it} \leq \alpha \quad t \in \mathbf{T} \tag{3.4}$$

$$\sum_{i \in \mathbf{V}} Z_{it} \geq 1 \quad t \in \mathbf{T} \tag{3.5}$$

$$\sum_{i \in \mathbf{V}} \sigma_i Z_{it} + \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} d_{ij} X_{ijt} \leq T_{\max} \quad t \in \mathbf{T} \tag{3.6}$$

$$FM_{i1} = Z_{i1} \quad i \in \mathbf{V} \tag{3.7}$$

$$FM_{it} \leq Z_{it} \quad i \in \mathbf{V}, t \in \mathbf{T} \setminus \{1\} \tag{3.8}$$

$$FM_{it} \leq 1 - Z_{iu} \quad i \in \mathbf{V}, t \in \mathbf{T} \setminus \{1\}, 1 \leq u < t \tag{3.9}$$

$$\sum_{j \in \mathbf{N}} X_{ijt} - \sum_{j \in \mathbf{N}} X_{jit} = L_{it} - E_{it} \quad i \in \mathbf{N}, t \in \mathbf{T} \tag{3.10}$$

$$L_{it} + E_{it} \leq 1 \quad i \in \mathbf{N}, t \in \mathbf{T} \tag{3.11}$$

$$\sum_{i \in \mathbf{N}} (L_{it} + E_{it}) \leq 2 \quad t \in \mathbf{T} \tag{3.12}$$

$$S_{i(t-1)} \leq S_{it} + \sum_{j \in \mathbf{N}} \frac{L_{jt} + E_{jt}}{2} \quad i \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.13)$$

$$\sum_{j \in \mathbf{N}} \frac{L_{jt} + E_{jt}}{2} + S_{i(t-1)} \geq S_{it} \quad i \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.14)$$

$$S_{i(t-1)} \leq L_{it} + S_{it} \quad i \in \mathbf{V}, t \in \mathbf{T} \setminus \{1\} \quad (3.15)$$

$$S_{0t} = 0 \quad t \in \mathbf{T} \quad (3.16)$$

$$S_{it} \geq X_{i0t} \quad i \in \mathbf{V}, t \in \mathbf{T} \quad (3.17)$$

$$S_{i(t-1)} \geq X_{i0t} \quad i \in \mathbf{V}, t \in \mathbf{T} \setminus \{1\} \quad (3.18)$$

$$X_{i0t} = X_{0it} \quad i \in \mathbf{V}, t \in \mathbf{T} \quad (3.19)$$

$$E_{it} \leq S_{it} \quad i \in \mathbf{V}, t \in \mathbf{T} \quad (3.20)$$

$$S_{it} \leq \sum_{j \in \mathbf{N}} X_{ij(t+1)} \quad i \in \mathbf{V}, 1 \leq t < \tau \quad (3.21)$$

$$\sum_{i \in \mathbf{V}} S_{it} = 1 \quad t \in \mathbf{T} \quad (3.22)$$

$$\sum_{k=t}^{t+\kappa} S_{1k} \geq 1 \quad 1 \leq t \leq \tau - \kappa \quad (3.23)$$

$$Z_{it} \leq \sum_{j \in \mathbf{N}} X_{ijt} + E_{it} \quad i \in \mathbf{V}, t \in \mathbf{T} \quad (3.24)$$

$$Z_{it} \leq \sum_{j \in \mathbf{N}} X_{jit} + L_{it} \quad i \in \mathbf{V}, t \in \mathbf{T} \quad (3.25)$$

$$(\alpha + 1)S_{j(t-1)} + (\alpha + 1)(1 - X_{ijt}) + U_{jt} \geq U_{it} + 1 \quad i, j (i \neq j) \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.26)$$

$$U_{it} \leq \alpha + 1 \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (3.27)$$

$$U_{it} \leq \sum_{j \in \mathbf{N}} \sum_{k \in \mathbf{N}} X_{jkt} + 1 \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (3.28)$$

$$U_{it} \geq S_{i(t-1)} \quad i \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.29)$$

$$(\alpha + 1)(1 - S_{i(t-1)}) + 1 \geq U_{it} \quad i \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.30)$$

$$U_{it} \geq \sum_{j \in \mathbf{N}} X_{ijt} \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (3.31)$$

$$U_{it} \geq S_{it} + \sum_{j \in \mathbf{N}} X_{ijt} \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (3.32)$$

$$U_{it} \leq (\alpha + 1) \sum_{j \in \mathbf{N}} X_{ijt} + (\alpha + 1) \sum_{j \in \mathbf{N}} X_{jit} \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (3.33)$$

$$R_{its} \leq Z_{it} \quad i \in \mathbf{N}, 2 \leq t \leq \tau, 1 \leq s < t \quad (3.34)$$

$$R_{its} \leq Z_{i(t-s)} \quad i \in \mathbf{N}, 2 \leq t \leq \tau, 1 \leq s < t \quad (3.35)$$

$$\sum_{k=t-s+1}^{t-1} Z_{ik} \leq s(1 - R_{its}) \quad i \in \mathbf{N}, 3 \leq t < \tau, 2 \leq s < t \quad (3.36)$$

$$R_{its} = 0 \quad i \in \mathbf{N}, t \in \mathbf{T}, t \leq s \leq \tau \quad (3.37)$$

$$R_{ius} \leq 1 - FM_{it} \quad i \in \mathbf{V}, t \in \mathbf{T} \setminus \{1\}, t < u < \tau, u - t < s < u \quad (3.38)$$

$$R_{its} \geq Z_{i(t-s)} + Z_{it} - \sum_{k=t-s+1}^{t-1} Z_k - 1 \quad i \in \mathbf{V}, 3 \leq t \leq \tau, 2 \leq s < t \quad (3.39)$$

$$X_{ijt}, L_{it}, E_{it}, S_{it}, Z_{it}, FM_{it}, R_{its} \in \{0,1\} \text{ and } U_{it} \geq 0 \quad (3.40)$$

The MILP model in (3.1)-(3.40) has $n^2\tau + \frac{1}{2}n\tau^2 + \frac{11}{2}n\tau + 3\tau$ binary variables, $(n+1)\tau$ continuous variables and

$\frac{1}{6}n\tau^3 + 2n\tau^2 + \frac{3}{2}n^2\tau + \frac{62}{3}n\tau + \frac{45}{2}\tau + \frac{1}{2}\tau^2 - n^2 - 3n - \mathcal{K} - 4$ constraints. Note that the meeting indicator variables Z_{it} , FM_{it} and R_{its} are defined for $i \in \mathbf{V}$ since the fictitious city cannot host a meeting. The objective function (3.1) seeks to maximize the difference between the collected rewards and the incurred routing costs. After consulting with campaign executives, we assume that the rewards are linearly depreciated as we get closer to the end of the campaign. The set of constraints (3.2)-(3.6) and (3.40) are adopted from the TSP literature (Öncan et al., 2009). However, the remaining constraints (3.7)-(3.39) are novel constraints which we developed specifically for this problem. A brief on these constraints is provided here; more details on the new constraints will be discussed in the next subsection.

Constraints (3.2)-(3.6) and (3.40) adopted from the literature

The set of inequalities (3.2) and (3.3) are typical selective TSP equations limiting the numbers of incoming and outgoing arcs to one for each node in \mathbf{N} . Constraints (3.4) impose an explicit upper bound α on the total number of daily activities ($\alpha \leq n$). Constraints (3.5) force the campaigner to perform at least one activity on each day t . Constraints (3.6) ensure the maximum daily tour duration is not violated. Binary integrality and nonnegativity constraints on the respective decision variables are defined in (3.40).

A brief on the new constraints (3.7)-(3.39)

Equality constraints (3.7) ensure that the first activity indicator variable and the activity indicator variable for day 1 must be equal. Constraints (3.8) set an upper bound for FM_{it} , thereby establish the coupling between FM and Z . Due to the maximization sense of optimization in the objective, the model

will try to set all FM_{it} variables to 1 as much as possible. Thus, there is no need for loose upper bound constraints on FM_{it} . Constraints (3.9) guarantee that if the first activity in city i was held on day t , then there cannot be an activity on an earlier day u , $u < t$.

Constraints (3.10) couple the binary decision variables X , L and E . Constraints (3.11) ensure that if the campaigner enters a city i on day t and does not leave it the same day, then $E_{it} = 1$ and $L_{it} = 0$. Likewise, if he exits a city i on day t and does not return to it the same day, then $E_{it} = 0$ and $L_{it} = 1$. According to constraints (3.12) the sum of the variables L and E over all cities on a given day cannot exceed two. In fact, this sum will be two only in a tour of Type 3, i.e. in an open tour.

Constraints (3.13) and (3.14) force the campaigner to stay overnight in the source i on day t if there is a closed tour that day. Constraints (3.15) make sure that terminal cities for days t and $(t-1)$ will be the same if there is a closed tour on day t . Constraints (3.16) set the variables S_{0t} to zero since the campaigner can never stay overnight in the fictitious city '0'. Constraints (3.17)-(3.18) are added to prevent the inclusion of the fictitious city in Type 1 and Type 3 tours. Along with constraints (3.19) they capture the presence of a Type 2 tour as follows: When the campaigner '*goes*' from city i to the fictitious city (namely city 0) on a given day t , then he directly '*returns*' from there the same day ($X_{i0t} = X_{0it} = 1$). Then he must also stay overnight in city i in both days t and $(t-1)$. In other words, the tours of both days must terminate in city i . This way the campaigner actually spends the whole day t in city i which points to the presence of a Type 2 tour.

The set of constraints (3.20) ensure that if the campaigner enters city i on day t and does not depart from there the same day, then he must stay overnight (sleep) in city i . This means that i must be the terminal city of the tour on day t . Constraints (3.21) guarantee that if the campaigner sleeps in city i on day t , he must depart from there the next day. Equalities (3.22) ensure that the campaigner can sleep in only one city every night. Constraints (3.23) prevent the campaigner from being away from the campaign base (city ‘1’) for more than κ consecutive days. The set of inequalities (3.24) and (3.25) assure that in order for a city i to host an activity on a given day t it must be visited that day in either of the three types of tours. When there is no visit to i , there is no activity in i either.

Constraints (3.26)-(3.33) are *Modified Lifted Miller-Tucker-Zemlin* inequalities (ML-MTZ) for subtour elimination adapted to RSP. The disaggregated constraints (3.34)-(3.35) provide the logical coupling between the binary variables R_{its} and Z_{it} . When $R_{its} = 1$, city i has to host an activity in both days t and $(t-s)$. If either day holds no activity in city i , then R_{its} will be forced to zero. Inequalities (3.36) ensure that if city i accommodates two activities in days t and $(t-s)$ and no other activity in between (i.e. if $R_{its} = 1$), then all corresponding Z_{ik} variables for k days in the interval $[t-s+1, t-1]$ should be zero.

Constraints (3.37) signify the domain restriction on the definition of the variables R_{its} . Constraints (3.38) make sure that if the first activity in city i is held on day t , then there cannot be a pair of activities on days u and $(u-s)$ where u comes after t and $(u-s)$ comes before t . The lower bounds on the variables R_{its} in (3.39) may seem unnecessary since their coefficients in the objective function to be maximized are all strictly positive. However, (3.39)

serve as valid inequalities and contribute affirmatively to the solution speed of the model.

3.3 More details on the newly-developed constraints

We will elaborate some of the constraints which we introduced to model the real-life logistic problem RSP.

Type 1 Tour Constraints: (3.10)-(3.12) and (3.13)-(3.14)

As discussed in Section 1, there are three possible types of daily tours. Type 1 indicates closed tours where the campaigner wakes up in city i in day t , visits and collects rewards in other cities (or in the starting city), comes back to city i and stays there overnight. So, city i

becomes both the source and the terminal node of the tour, see Figure 3.1. Constraints (3.10) couple the logical binary variables E_{it} (enter

the city and do not leave it the same day) and L_{it} (leave the city and do not return to it the same day) with the binary routing variables X_{ijt} , and ensure linear dependence between L and E . Constraints (3.11) guarantee that either E_{it} or L_{it} can be nonzero, but not both. As a result, constraints (3.10), (3.11) and (3.12) together ensure that if the campaigner enters a city i , but does not leave it, then we have $E_{it} = 1$ and $L_{it} = 0$. In addition, they ensure that if the campaigner exits, but does not re-enter city i on the same day, then we have $E_{it} = 0$ and $L_{it} = 1$. Constraints (3.10) can be interpreted as the following:

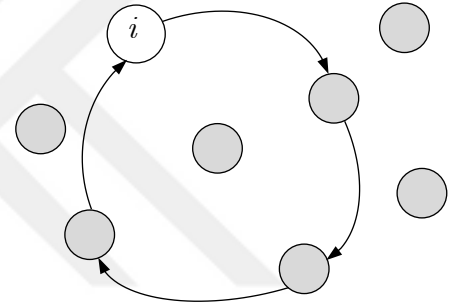


Figure 3.1 Type 1 tours

outgoing degree_(i) - incoming degree_(i) = $L_{it} - E_{it}$. If the campaigner enters and then exits city i , or if he does not visit city i at all on day t , then both L_{it} and E_{it} will be zero. Consequently, on a given day t one can at most leave one city and enter another city.

Constraints (3.13) and (3.14) ensure that the starting city (source) is the same as the ending city (terminal) in day t if the tour of day t is a closed tour. Also the term $\sum_{j \in \mathbf{N}} \frac{L_{jt} + E_{jt}}{2}$ being zero means that there is a loop, i.e. a

closed tour in day t . The existence of a loop means that the campaigner begins his tour in city i in the morning and comes back to the same city by the end of the day. So, constraints (3.13) and (3.14) say that if there is a closed tour, the campaigner stays overnight exactly in the same city which he stayed in overnight the day before. That is to say that constraints (3.13) enforce $S_{it} = 1$ if the campaigner wakes up in city i in day t , and returns to i to sleep there. In that case, all L_{it} and E_{it} will become zero leading to a closed tour which begins and terminates in city i in day t . If there is no closed tour, there will be only one city i that the campaigner exits and does not enter, thus $L_{it} = 1$. Likewise, there will be only one city j that he enters and does not exit, thus $E_{jt} = 1$. Consequently, the values of L_{it} and E_{jt} for all other cities will be

equal to zero due to (3.12), and we will have $\sum_{j \in \mathbf{N}} \frac{L_{jt} + E_{jt}}{2} = 1$.

Type 2 Tour Constraints: (3.16)-(3.19)

To model specific days in which the campaigner does not travel to any city, we define a fictitious city '0'. All rewards and costs associated with this

city are set to zero. Constraints (3.16) prevent the campaigner from staying overnight in this city. Constraints (3.19) force him to exit the fictitious city ‘0’ in case he has entered it. Constraints (3.17) and (3.18) are considered for the cases where the campaigner wakes up in city i , realizes an activity there and sleeps in the same city without leaving it. These two sets of constraints ensure that if he goes from city ‘0’ to city i (or from i to ‘0’), then he must sleep in city i .

Type 3 Tour Constraints: (3.20)-(3.21)

In Type 3 tours the campaigner wakes up in city i in day t , possibly visits several other cities, and sleeps in another city j , see Figure 3.2. The set of constraints (3.20)-(3.21) ensure that if the campaigner does not leave city i in day $(t+1)$, i.e. if the outgoing degree of city i is zero, then he cannot have slept in i in day t (the day before). He may go to the fictitious city, but should depart from the starting city i under any circumstance.

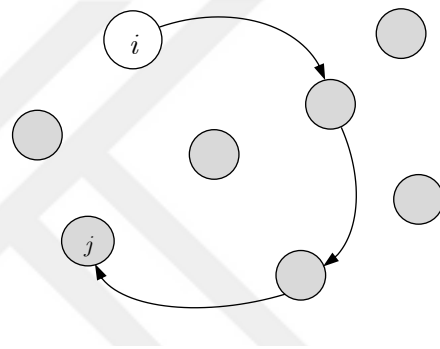


Figure 3.2 Type 3 tours

Subtour Elimination Constraints: (3.26)-(3.33)

In order to ensure that every feasible route contains only one sequence of visited cities, we introduce constraints (3.26)-(3.33) for the purpose of subtour elimination. We do not consider lasso subtours in our model. In order to formulate Lifted Miller-Tucker-Zemlin subtour elimination constraints (L-MTZ) originally proposed by Desrochers and Laporte (1991), we consider two alternatives.

Alternative 1

For day t suppose there is an open tour starting in city i_0 . In this case, the corresponding values of the decision variables are set as follows:

$$S_{i_0(t-1)} = 1, S_{i_0t} = 0, E_{i_0t} = 0, \sum_{j \in \mathbf{N}} X_{ji_0t} = 0 \text{ and } L_{i_0t} = 1. \text{ Note that by}$$

setting $L_{i_0t} = 1$, both E_{i_0t} and $\sum_{j \in \mathbf{N}} X_{ji_0t}$ become equal to 0. We use this

information as well as the information acquired in the closed tour alternative to incorporate a new dummy variable into the L-MTZ.

Alternative 2

For day t suppose there is a closed tour starting in city i_0 . In this case, the corresponding known values of variables are as follows:

$$S_{i_0,t-1} = 1, S_{i_0,t} = 1, E_{i_0,t} = 0, L_{i_0,t} = 0 \text{ and } \sum_{j \in \mathbf{N}} X_{i_0jt} + \sum_{j \in \mathbf{N}} X_{ji_0t} = 2.$$

We assume that the campaigner can visit at most \bar{n} cities in each day. U_{it} becomes k if city i is the k^{th} visited city ($i, k = 1, 2, \dots, n$). The inequalities (3.26) are L-MTZ subtour elimination constraints which state that there should be only a single tour covering visited cities and not two or more disjointed tours that only collectively cover these cities. They also guarantee that if there is an arc between cities i and j while city j is not the starting city, U_{jt} will be greater than U_{it} by 1 unit only.

Reward Function Constraints: (3.34)-(3.39)

Note that after defining constraints (3.34) and (3.35), we actually do not need the constraints (3.38), as the set of constraints (3.9) and (3.35) guarantee that if the first activity is held on day t , no previous activity may be performed before day t . In order to verify the equations (3.34)-(3.39), let us consider the following possibilities. Suppose that the first activity in city i was held on day t , i.e. $FM_{it} = 1$ and hence (i) $R_{its} = 0$ for $1 \leq s \leq t-1$ according to (3.38). This is also guaranteed by constraints (3.9) and (3.35). If $FM_{it} = 1$, then there cannot be an earlier activity on day $(t-s)$. This leads to (ii) $Z_{iu} = 0$ for $1 \leq u \leq t-1$ due to (3.9). Again if $FM_{it} = 1$, then there cannot be an activity on an earlier day u , $u \leq t-1$.

Eventually, we have (iii) $R_{ius} = 0$ for $2 \leq u \leq t-1$, $1 \leq s \leq u-1$. The equalities (iii) also follow from constraints (3.9) and (3.35). Finally, if $FM_{it} = 1$ again, then there cannot be a pair of activities on days u and $(u-s)$ where u comes after t and $(u-s)$ comes before t , i.e. (iv) $R_{ius} = 0$ for $t+1 \leq u \leq \tau-1$, $1 \leq u-s \leq t-1$. This is the same domain definition as the one in constraints (3.38). In other words, the equalities (iv) are implied also by constraints (3.38).

3.4 An alternative formulation for satisfying maximum tour duration constraint

An alternative way of satisfying the maximum daily tour duration is to introduce the continuous decision variable A_{it} which indicates the arrival time to city i on day t . Such a formulation is especially useful for problems with time-windows. Such a formulation can also be important if the schedule of

coaches or flights are incorporated into the model or the time slots of the day are considered in the reward function. However, based on our preliminary empirical testing of both formulations, it was found that constraints (3.6) provides better results, see Section 8.

$$A_{it} \leq T_{\max}(1 - S_{i(t-1)}) \quad i \in \mathbf{V}, t \in \mathbf{T} \setminus \{1\} \quad (3.41)$$

$$A_{jt} \geq A_{it} + \sigma_i Z_{it} + d_{ij} - T_{\max}(1 - X_{ijt} + S_{j(t-1)}) \quad i, j \in \mathbf{V}, t \in \mathbf{T} \setminus \{1\} \quad (3.42)$$

$$A_{jt} \leq A_{it} + \sigma_i Z_{it} + d_{ij} + T_{\max}(1 - X_{ijt} + S_{j(t-1)}) \quad i, j \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.43)$$

$$0 \leq A_{it} \leq T_{\max} - \sigma_i Z_{it} \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (3.44)$$

$$A_{it} + \sigma_i Z_{it} + d_{ij} \leq T_{\max} + M(2 - S_{j(t-1)} - S_{jt}) \quad i \in \mathbf{V}, t \in \mathbf{T} \setminus \{1\} \quad (3.45)$$

$$A_{it} \leq T_{\max} \left(\sum_{j \in \mathbf{V}} X_{jit} + \sum_{j \in \mathbf{V}} X_{ijt} \right) \quad i \in \mathbf{V}, t \in \mathbf{T} \quad (3.46)$$

The set of constraints (3.41) ensure that the arrival time for city i on day t will be zero if the salesman stays overnight on day $t-1$. Upon arrival in city j , the travel time between city i and city j and the activity time in city j are considered in constraints (3.42) and (3.43). Inequalities (3.44) impose the lower and upper bounds of A_{it} . Constraints (3.45) are the general maximum tour duration definition. These are binding for open tours. The set of constraints (3.46) are also binding for closed tours.

3.5 Added valid inequalities

In addition to the original constraints of the problem, we include the following valid inequalities:

$$\sum_{t \in \mathbf{T}} FM_{it} \leq 1 \quad i \in \mathbf{V} \quad (3.47)$$

$$L_{it} \leq 2 - S_{i(t-1)} - S_{it} \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (3.48)$$

$$E_{it} \leq 2 - S_{i(t-1)} - S_{it} \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (3.49)$$

$$X_{ijt} + X_{jit} \leq 1 + S_{it} + S_{jt} \quad i, j (i < j) \in \mathbf{N}, t \in \mathbf{T} \quad (3.50)$$

Valid inequalities (3.47) ensure that the first activity for each city can occur at most once during the campaign period. Valid inequalities (3.48) and (3.49) state that if the campaigner stays overnight in the same city on days t and $(t-1)$, then the tour on day t will be a closed tour; hence, the corresponding variables L_{it} and E_{it} must be zero. Valid inequalities (3.50) guarantee that if cities i and j are not terminal cities on day t , there should not be a cyclic tour between them. These constraints are proved to be significantly effective. We provided the computational evidence in Section 8.

3.6 An arc-based formulation for subtour elimination

We also tested an arc-based TSP model for our problem RSP. It is adopted from the single-commodity flow formulation originally developed by Gavish and Graves (1978) for TSP. Their formulation uses nonnegative flow variables to indicate the amount of goods flowing from node i to node j after collecting or dropping the load at i . We found that the node-based formulation built

upon the L-MTZ is relatively faster than the arc-based formulation of Gavish and Graves. Details on the CPU time comparison between the two models are not reported here but can be collected from the author if required. The arc-based formulation for subtour elimination is provided below.

Single Commodity Flow Formulations:

New decision variable:

F_{ijt} : A continuous variable used in Modified Gavrish-Grave Subtour Elimination constraints indicating the flow on arc i - j on day t , $F_{ijt} \in R$. Note that $F_{iit} = 0$.

$$\begin{array}{l}
 \left. \begin{array}{l} \\ \\ \end{array} \right\} \begin{array}{l} \\ \\ \end{array} \left\{ \begin{array}{l} \\ \\ \end{array} \right. \\
 \text{Closed Tour} \\
 \sum_{j \in \mathbf{N}} F_{jit} + 1 \leq \sum_{j \in \mathbf{N}} F_{ijt} + nS_{i,t-1} + S_{it} \quad i \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.51) \\
 \sum_{j \in \mathbf{N}} F_{jit} + 1 \geq \sum_{j \in \mathbf{N}} F_{ijt} - nS_{i,t-1} + S_{it} \quad i \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.52) \\
 \sum_{j \in \mathbf{N}} F_{ijt} \leq n(1 - S_{it}) \quad i \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.53) \\
 \sum_{j \in \mathbf{N}} F_{jit} \leq n(2 - S_{i,t-1} - \delta_t) \quad i \in \mathbf{N}, t \in \mathbf{T} \setminus \{1\} \quad (3.54)
 \end{array}$$

The main idea in the arc-based formulation is to benefit from the flow among nodes for eliminating subtours. In case city i is visited, the outflow of this city should be higher than its inflow ($outflow_{(i)} - inflow_{(i)} \geq 1$). Therefore,

constraints (3.51)-(3.54) also verify the before-mentioned statement when the salesman does not sleep in city i the previous night (day $t-1$).

$$F_{ijt} \geq X_{ijt} - S_{i,t-1} \quad i, j \in \mathbf{N}, t \in \mathbf{T} \quad (3.55)$$

Constraints (3.55) set the lower bound for flow $F_{ijt} \in R$ such that, if the salesman goes from city i to city j , the flow should be equal to at least 1.

$$F_{ijt} \leq (n-1)X_{ijt} \quad i, j \in \mathbf{N}, t \in \mathbf{T} \quad (3.56)$$

$$F_{jit} \leq 1 + n(2 - S_{i,t-1} - X_{ijt}) \quad i, j \in \mathbf{N}, t \in \mathbf{T} \quad (3.57)$$

Constraints (3.56) and (3.57) ensure that flow (F_{ijt}) cannot exceed the number of total visited arcs each day.

$$F_{ijt} \geq 0 \quad i, j \in \mathbf{N}, t \in \mathbf{T} \quad (3.58)$$

Equations (3.58) are the non-negativity constraint.

Chapter 4

AN APPLICATION TO ELECTION LOGISTICS

The Roaming Salesman Problem has several applications in logistics and scheduling problems. One of the most suitable contexts for the application of RSP is election logistics. The problem proposed and solved here deals with determining daily routes for a traveling party leader (the politician) who speechifies in various cities during a given campaign period until elections. This is a new problem that we call the “Multi-Period Traveling Politician Problem” (MPTPP), motivated by extensive real-world applications.

MPTPP which we investigate in this study can be considered a novel version of the Roaming Salesman Problem (RSP). Reminiscent of the salesperson in the RSP, the MPTPP revolves around a politician who holds meetings in various cities during a given campaign period. Both problems can be considered to generalize the classical version of the traveling salesman problem (TSP) by extending the planning horizon to days; hence, they both correspond to a multi-period problem.

MPTPP can be described as follows. On a graph with static edge costs and time-dependent vertex profits, the MPTPP seeks a closed or open tour for each day of a campaign period with the objective of maximizing the net benefit. The party leader is not required to visit all cities making the problem selective. Moreover, s/he can stay overnight in any city to start the tour of the next day.

This means that, similarly to RSP, there are no any fixed departure (central) nodes in daily tours.

We consider the 80 cities (provinces) of Turkey plus a campaign center, namely the capital city Ankara as well as 12 towns. At the time of the June 2015 election, Turkey had 81 cities and 85 electoral zones where İstanbul was comprised of three zones, İzmir and Ankara of two zones each. Each city is associated with a dynamic base reward and a fixed meeting duration. During the campaign period, the party leader cannot be out of the campaign center for more than κ consecutive days. In addition, the total length of the talks and travel times between cities on the same day cannot exceed T_{\max} hours.

The proposed model utilizes a multifaceted reward function. The reward of a meeting in a city is linearly depreciated according to the meeting date and the recency of the preceding meeting in the same city. The reward function determines the reward of each city considering four factors: i.) Population of that city. ii.) The party's vote rate in the previous election and the criticality of the city. iii.) The number of remaining days until the election. iv.) The number of days passed since the previous meeting in the same city. As we get closer to the election day, the reward of crowded cities decreases significantly. On the other hand, successive meetings in a city are severely depreciated. This prevents the crowded cities from being visited multiple times within short time intervals.

We first present an example followed by the data collection and the way the rewards are computed.

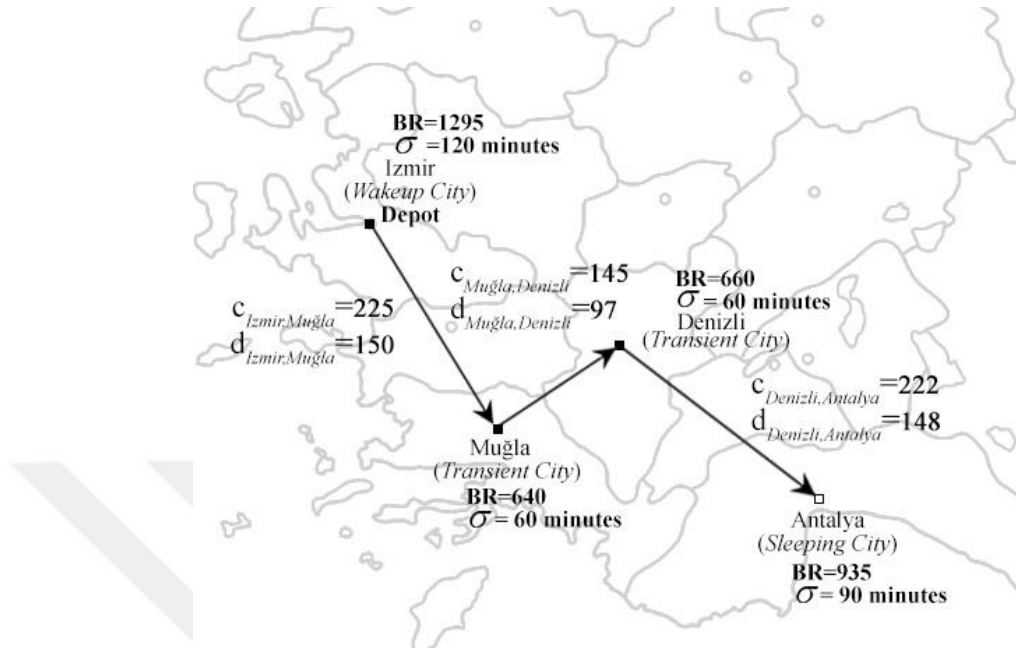


Figure 4.1 A simple example presented to illustrate the problem structure.

4.1 A real case example

To further explain the MPTPP setting, we present a simple example of a “one-day” tour as illustrated in Figure 4.1. In this example, we assume that the politician wakes up in İzmir (source city). The daily available hours are assumed to be from 09:00 a.m. to 10:00 p.m. meaning that the maximum tour duration is 13 hours. The base reward amount and the meeting duration of each city are provided in Figure 4.1.

For this simple example, we show a feasible schedule to explain the settings of the problem. At the beginning of the day (i.e. at 09:00 a.m.), the politician holds a meeting in İzmir and leaves the source city at 11:00 a.m. with a collected reward of 1295. Next, he/she visits Muğla at 01:30 p.m. by passing through Aydın without holding a meeting. The politician leaves Muğla at 02:30

p.m. while the net benefit is 1710 ($1295 - 225 + 640$) and arrives in Denizli at 04:07 p.m. The leaving time there is 05:07 p.m. where collected net benefit is 2225 ($1710 - 145 + 660$). Finally, the politician arrives in Antalya at 07:35 p.m. where the talk duration is 90-minute. At 09:05 p.m. the daily tour is finished by collecting a total net benefit of 2938. Since there is no time left to visit neighbor cities, the politician stays overnight in Antalya. The next morning, he/she starts the tour by leaving Antalya (most probably with holding no meeting there due to the penalty imposed by the reward function) to roam other cities.

4.2 Data collection and analysis

The time limit T_{\max} which is 14 hours (12 hours) in the summer (in the winter) imposes an implicit threshold on the number of cities that can be visited in any given day. Each city can accommodate at most one meeting a day. There can be an upper bound (such as two or three) on the total number of meetings held in each city during the campaign period. The meeting durations including preparation and holding times range from 60 to 120 minutes depending on the population of the host city. For the three biggest cities, namely İstanbul, Ankara and İzmir, it is 120 minutes. For cities with less than a million population it is 60 minutes, and for all other ones, this duration is 90 minutes. Another point that needs mentioning is the periodic returns to the campaign base Ankara. The politician cannot be away from the capital city for more than κ consecutive days where we choose $\kappa \in \{4, 5, 6, 7\}$.

4.3 Preprocessing of the symmetric time/cost matrix

We assume that the politician can travel either by bus or by airplane. The unit of traveling on the road by bus is assumed 1.50 TL/km where TL refers to Turkish Lira. For those cities with an airport and a bus travel time of more than 270 minutes (4½ hours) from one another, the faster travel option (either airplane or party bus) is chosen. In the calculation of the traveling costs between those origin/destination pairs for which flying proves to be a faster option, we assume that the politician flies with at least four other party executives. We determined the cheapest available ticket prices accordingly. The travel cost and time matrices have been computed after the investigation of the road travel and flight times as well as the ticket prices in Google Maps and TurkishAirlines.com, respectively.

We computed the parameters c_{ij} and d_{ij} to avoid defining a new binary variable to capture the mode of travel, as this would dramatically increase the complexity of the mathematical model. We define three new city sets and four new parameters for the preprocessing of the symmetric cost/time matrix.

- \mathbf{I}_H : The set of cities with hub airport.
- \mathbf{I}_{NA} : The set of cities with no airport.
- \mathbf{I}_{NHA} : The set of cities with non-hub airport.
- t_{ij}^{road} : Time required in minutes for driving from city i to city j .
- t_{ij}^{fly} : Time required in minutes for flying between the airports of cities i and j .
- $cost_{cc(i),A(i)}$: Cost of going from the city center to the airport of city i .

$t_{cc(i),A(i)}$: Time required in minutes for going from the city center to the airport of city i .

Step 1

If there is a direct flight from city i to city j and if $t_{ij}^{road} > 270$, then we calculate actual flight times t_{ij}^{actual} as follows:

$t_{ij}^{actual} = t_{cc(i),A(i)} + t_{ij}^{fly} + t_{A(j),cc(j)} + t^{delay}$ with t^{delay} included to consider possible delays.

Step 2

Suppose $k \in \mathbf{I}_{\text{NHA}}$ and $j \in \mathbf{I}_{\text{NA}}$ where $t_{kj}^{road} > 270$. We calculate $t_{kj}^{alt} = t_{k,i(j)}^{actual} + t_{i(j),j}^{road}$ where $i(j)$ is the nearest city with hub airport to j , i.e. $i(j) = \arg \min_{i \in \mathbf{I}_{\text{H}}} \{t_{ij}^{road}\}$. If $t_{kj}^{alt} < t_{kj}^{road}$, then we set $t_{kj}^{new} = t_{kj}^{alt}$ and $cost_{kj}^{new} = cost_{kj}^{alt} = cost_{cc(k),A(k)} + cost_{k,i(j)} + cost_{i(j),j}$ as the new travel times and costs in our symmetric cost/time matrix.

Computation of the objective function

A simple normalization coefficient is used to make the rewards and traveling costs compatible in the objective function (3.1). Given the index sets and the decision variables in Section 3, the depreciation in (3.1) is defined as $\frac{m-t+1}{m}$. The actual reward accruing from meetings are calculated according to the following rules. Further details are provided in the next section.

Rule 1: The earlier a meeting in the campaign period, the higher its reward.

Rule 2: The shorter the time difference between two successive meetings in the same city, the lower the reward earned from the latter meeting.

Rule 3: The sooner there is a follow-up (repeat) meeting in the same city, the more its actual reward is depreciated.

4.4 Time-dependent rewards

In this section we explain the reward calculation and the categorization of cities in Turkey from the main opposition party's perspective.

The proposed model utilizes a multifaceted reward function. Initially, a nonnegative prize of π_i (*base reward*) is specified for holding a meeting in each city $i \in \mathbf{V}$ where π_i depends on two factors:

Factor 1: Population of city $i \in \mathbf{V}$ (Pop_i).

Factor 2: Ratio of votes of the politician's party (PP).

In addition, the actual reward earned in city $i \in \mathbf{N}$ on day $t \in \mathbf{T}$ is based on two further factors:

Factor 3: Number of remaining days denoted by $(m - t)$ until the end of the campaign.

Factor 4: Number of days passed since the previous meeting in the same city, denoted by s where $1 \leq s \leq t - 1$.

The first two factors directly affect the base reward π_i whereas the remaining two make the reward time-dependent. Each factor is explained below.

4.4.1 Factor 1: Population

Population is one of the most decisive factors in determining the importance of a city in an election campaign. A base reward of 100 is assigned to all cities initially. Each city's population is divided by the minimum population of all cities and multiplied by a city-dependent multiplier. This multiplier is taken as 3.0 for the top seven (most populated cities), but 2.0 for İstanbul to close the drastic gap between the reward of İstanbul and other cities. The remaining cities are assigned a value of 5.0. The result is then multiplied by a *Criticality Factor* (CF_i) where the operator $\llbracket \cdot \rrbracket$ rounds its argument to the nearest integer number.

$$\pi_i = CF_i \times \left(100 + \left\llbracket \frac{Pop_i}{Min.Pop.} \right\rrbracket \times Multiplier_i \right) \quad (4.1)$$

Figure 4.2 represents the population of all 81 provinces in Turkey.

4.4.2 Factor 2: Ratio of votes and Criticality Factor

A useful information is to incorporate the importance of a given city given its previous voting pattern. We define four criticality categories to label the importance of a city from the perspective of the PP. Different towns or electoral zones of a given city are mutually assigned to the same criticality category. We introduce the *Criticality Factor* given for city i (CF_i). Categories are defined as follows:

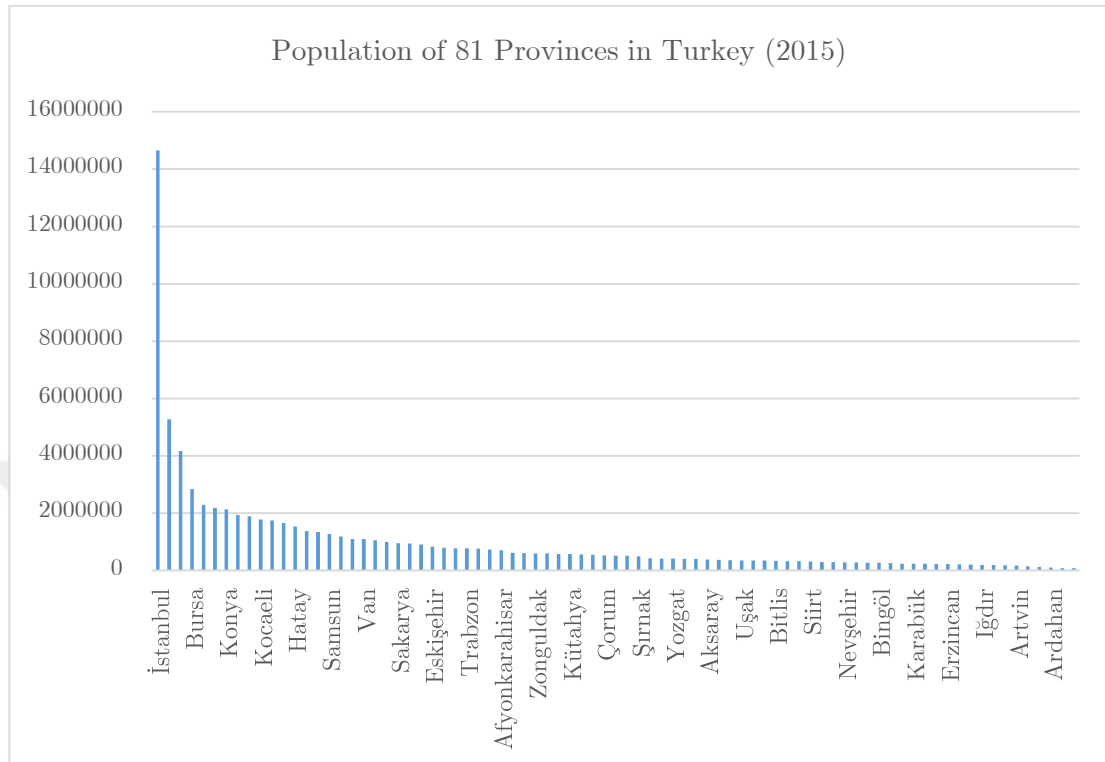


Figure 4.2 Population of 81 provinces in Turkey

Category 1: Noncritical Cities

These are the cities in which the number of seats won by the PP would not change even when the number of its votes changes by $\pm 20\%$. We set $CF_i = 2$ for $i \in \text{Noncritical Cities}$.

Category 2: Negative Critical Cities

In these cities, a 20% increase in the PP votes does not affect its seat number in the parliament (the number of PP deputies elected from those

cities). However, a 20% decrease would cause PP to lose at least one seat. We set $CF_i = 3$ for $i \in \text{Negative Critical Cities}$.

Category 3: Positive Critical Cities

In positive critical cities, PP would gain at least one more seat in the event of a 20% increase in the vote count of the past election. However, there exists no risk of losing any seat in the event of a 20% loss in the votes. We set $CF_i = 4$ for $i \in \text{Positive Critical Cities}$.

Category 4: Positive-and-Negative Critical Cities

The situation is most sensitive in cities of category 4 where both a 20% increase and a 20% decrease in the vote count would impact the party's current seat number in the parliament. Hence we set $CF_i = 5$ for $i \in \text{Positive-and-Negative Critical Cities}$.

The motivation for choosing these CF values is to assign high rewards to highly populated cities, but doing so at a decreasing rate. Another motivation is to close the enormous gap between metropolitan cities and other midsize cities of Turkey. For instance, İstanbul, despite its ~15 million population, should not earn thrice as much base reward as Ankara just because of having thrice as much population.

4.4.2.1 Criticality Analysis

In order to find the effect of variation in the number of votes on the number of deputies in the parliament, the data of June 2015 election has been analyzed for all cities. In our criticality analysis, we first simulated the election procedure according to the actual vote counts registered in the election of June 2015. We were able to reproduce exactly the same seat distributions in all 85

electoral zones of Turkey which shows the validity of the implemented simulation. Next, we evaluated each city by decreasing and increasing the votes of PP in that city by 20%. Finally, we categorized cities as discussed above. The reward statistics are provided in Table 4.1.

Table 4.1 Statistics of rewards in criticality categories

	Noncritical	Negative Critical	Positive Critical	Positive-and- Negative critical
# of cities	42	19	11	9
Avg Reward	268	444	564	1,193
Min. Reward	210	315	460	800
Max Reward	440	675	680	2,370

To illustrate the effect of the CF let us consider for example Samsun and Kastamonu in the Black Sea Region. We have $\pi_{Samsun} = 540$ and $\pi_{Kastamonu} = 500$ although Samsun's population is more than three times Kastamonu's population. The base reward of Kastamonu almost catches up with Samsun because Kastamonu is a positive-critical city whereas Samsun is a negative-critical city. We include only those cities of Turkey which have a base reward value π_i higher than 300 in the set $\mathbf{V} = \{1, \dots, n\}$ where the minimum is 210 as listed in Table 4.1. This leads to $n = 51$ cities to be considered in our MPTPP model. As highlighted earlier, MPTPP has obviously a selective nature where not all cities in $\mathbf{N} = \{0, 1, \dots, n\}$ need to be included in the meeting plan. The base rewards of all 81 cities of Turkey are illustrated in Figure 4.3 below.

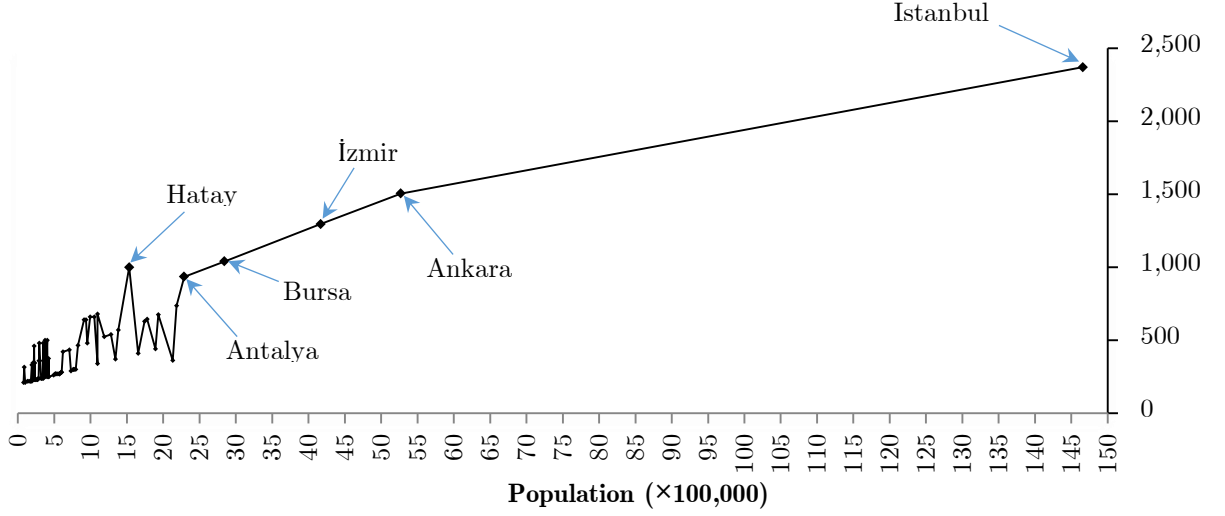


Figure 4.3 Base rewards (π_i values) of all 81 provinces.

4.4.3 Factor 3: Number of remaining days until the election day

We assume that as we get closer to the end of the campaign, the influence of meetings will decrease. In order to inflate the base rewards with the increasing number of remaining days until the elections, we develop the following formula to represent the reward.

$$Reward_i(t) = \pi_i \frac{m - t + 1}{m} \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (4.2)$$

If PP decides to reverse the effect of Factor 3, one can modify the formula in (4.2) by setting $Reward_i(t) = \pi_i \frac{t}{m}$. The actual reward of a meeting would then be the lowest on the first day and the highest on the last day of the campaign.

4.4.4 Factor 4: Number of days passed since the previous meeting

In order to prevent the model from visiting highly-rewarded cities frequently, we severely penalize repeat meetings. To inflate the base rewards with the increasing number of days passed since the last meeting, the following formula is adopted.

$$Reward_i(t, s) = \pi_i \frac{m-t+1}{m} \times \frac{s}{Km} \quad i \in \mathbf{N}, t \in \mathbf{T} \quad (4.3)$$

where s represents the number of days passed since the last meeting and K ($K \geq 1$) is a prespecified depreciation factor for repeat meetings. Note that the criterion of depreciation is not the number of meetings held in city i so far, but the recency of the previous meeting.

4.5 Supplementary assumptions in MPTPP

In this section, we introduce several operational assumptions pertinent to the meeting tours in MPTPP. We propose the associated constraint equations (4.4)-(4.6) below, which have not been included in the original RSP model in (3.1)-(3.40).

$$\sum_{i \in \text{Big Cities}} Z_{it} \leq 1 \quad t \in \mathbf{T} \quad (4.4)$$

$$R_{it1} = 0 \quad i \in \mathbf{V}, t \in \mathbf{T} \setminus \{1\} \quad (4.5)$$

$$\sum_{t \in \mathbf{T}} Z_{it} \leq \beta_i \quad i \in \mathbf{V} \quad (4.6)$$

The first supplementary assumption formulated in (4.4) is that a daily tour cannot involve more than one big city where the description ‘*big city*’ applies to İstanbul, Ankara and İzmir—namely the top three cities with respect to population size. The second assumption gives rise to the constraints (4.5) which state that it is not permitted to make two meetings in the same city on two consecutive days. The third assumption brings about a maximum number of meetings allowed in a given city i during the entire campaign period. This maximum number is denoted by the parameter β_i in (4.6)



Chapter 5

SCENARIO ANALYSIS

In order to highlight the unique features of MPTPP, we test our model on different scenarios. In this chapter, we present the details of the developed scenarios. In Section 5.1, we describe the details of four developed extreme scenarios. In Section 5.2, we describe the scenario with an alternative reward function.

5.1 Scenario analysis level 1: Extreme scenarios

We investigated the following four scenarios with $T_{\max} = 14$ hours as the maximum tour duration (Shahmanzari et al. 2018).

Scenario 1: All-inclusive base scenario, referred to as the *base scenario*.

Scenario 2: The base scenario with the additional restriction of at most one meeting in each candidate city throughout the campaign. This additional restriction is actually a relaxation on *Scenario 1*.

Scenario 3: The base scenario where the objective function involves only collected rewards and no traveling costs. In addition, the politician needs not to return to the capital city periodically.

Scenario 4: The base scenario with only closed daily tours originating and terminating at the capital city every day.

5.1.1 Scenario 1: Model Full-MILP (Base Scenario)

The politician's campaign starts and ends in Ankara. Thus party leader starts his campaign in Ankara in the morning of day 1. For simplicity, we treat day 0 as the fictitious day.

- Party leader cannot be away from Ankara for more than $\kappa = 5$ days in a row. In other words, s/he returns to Ankara at least once every $\kappa + 1$ days.
- Other than starting and ending in Ankara on day 1 and day τ , respectively, there is no restriction as to where to end up (sleep) the tour at a given day and start the tour of the next day (wake up). Thus, the tour of a given day t can comprise either an open route or a closed route.
- Each day accommodates one daily tour only. Starting the daily tour in a city i , holding a meeting there, and staying in the same city overnight is also an option. This is modeled as a closed route from city i to the fictitious city (city 0) and back to city i with no traveling costs.
- There is a "hard" maximum tour duration (T_{\max}) constraint in place which prohibits daily tours in excess of 14 hours.
- When the travel time from city i to city j does not exceed $4 \frac{1}{2}$ hours, the party leader will be transferred by bus. Otherwise, the transfer happens by airplane. We calculate the time of air travel as the sum of actual flight time, one hour for VIP check-ins, and the travel time by bus between the departure and arrival city centers and their respective operational airports.
 - The resulting travel time is compared to the travel time by party bus. Whichever option takes shorter is chosen as the preferred mode of transfer. The travel cost from city i to city j is calculated according to

the chosen mode of transfer. In case the airplane option is chosen, we multiply the ticket fare of the economy class by 5 assuming that party leader flies with four other party executives every time s/he boards an airplane.

· Candidate cities are divided into three segments: Big Cities (Istanbul, Ankara, and Izmir) where the number of meetings hosted is limited to three, two and one, respectively (parameter β_i). These count values may change after discussion with other party leaders and other party technocrats in charge of party campaign planning.

- A daily tour cannot involve more than one Big City.
- Two meetings in the same city on two consecutive days are also forbidden leading to valid inequality.
- There cannot be any day without a meeting.
- The number of meetings held each day is limited to four, i.e. $\alpha = 4$ in the original RSP constraints (3.4).
- As the problem is selective, not all cities have to be included in the campaign program.

- The objective function is the same as (3.1). The earlier the meeting, the higher the reward. We have computed base reward for each city i according to the city's population and the so-called criticality factor (CF_i). The latter factor, namely CF is derived from the sensitivity of the number of seats (deputies in the Grand National Assembly of Turkey) party has won from each city in the previous two general elections. CF is namely a quantitative measure for the sensitivity to a possible change by $\pm 20\%$ in the number of votes received.

- If the meeting in city i is held on day 1, then we collect Full π_i from city i . Otherwise, π_i is depreciated linearly with time. Hence, if the

meeting is held on day τ (the last day of the campaign), the collected RP will be only (π_i / τ) . The depreciation factor for a meeting held on day t is therefore equal to $\frac{\tau - t + 1}{\tau}$.

- Likewise, repeat meetings are also depreciated. Here the depreciation criterion is not the number of meetings held in city i so far, but the recency of the previous meeting. The lesser the number of days passed since the previous meeting in city i , the higher the depreciation of the reward that can be collected from a repeat meeting. Accordingly, if the prior meeting in the same city was held s days ago, the depreciation factor is s/τ .

5.1.2 Scenario 2: Model Full-1Meet

Scenario 2 is derived from Scenario 1 by revoking the option of multiple meetings in big and midsize cities. Since each city can host at most one meeting during the campaign, the model simplifies drastically as follows: All 2-index binary variables representing first meetings and all 3-index binary variables representing repeat meetings are now void. The definition and coupling constraints involving those variables also become void. The model diminishes to $n^2\tau + 4n\tau + 3\tau$ binary variables and

$\frac{3}{2}n^2\tau + \frac{49}{2}n\tau + 22\tau - n^2 - 7n - \mathcal{K} - 4$ constraints. The net benefit definition comprising the objective function is simplified as shown in (5.1). It is worth noting that the optimal solutions (thus the optimal objective values) of Scenario 1 and Scenario 2 may be identical.

$$\begin{aligned}
NET\ BENEFIT(Full-1Meet) = & \\
\sum_{i \in \mathbf{N}} \sum_{t \in \mathbf{T}} \pi_i \frac{\tau - t + 1}{\tau} Z_{it} - \bar{K} \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} \sum_{t \in \mathbf{T}} c_{ij} X_{ijt} & \quad (5.1)
\end{aligned}$$

5.1.3 Scenario 3: Model Rew-Only

Scenario 3 is derived from Scenario 1 by two modifications: (i) The necessity to periodically return to the capital Ankara at least once every κ days is lifted. The politician has full freedom to hop from one city to another as he/she sees fit. He/she can stay overnight in any city. Yet the campaign is still going to start in Ankara on day 1. (ii) Traveling costs are discarded from the objective function. This fundamental change motivates the politician to roam between all candidate cities without worrying about the cost of traveling.

5.1.4 Scenario 4: Model Alt-1Depot

Scenario 4 is derived from Scenario 1 by a fundamental paradigm shift in which the politician wakes up in the capital city Ankara every morning and returns there to sleep by the end of every day. This implies that each daily trip is going to be a closed tour with Ankara being the depot of the trip.

We adapt the single-commodity flow formulation of Gavish and Graves (1978) to this scenario as follows: Node-based load variables U_{it} indicating the order of visit to each city are replaced by arc-based continuous variables F_{ijt} indicating the flow from one city i to another city j on day t . The reason for using single commodity flow-based formulation only in scenario 4 is due to the fact that all daily tours are closed tour in this scenario and there is a single known depot. This reduces the problem to a multi-period selective TSP with a single depot which is the campaign base. Based on the results of the pilot tests,

the flow-based formulation finds a better solution compared with the node-based formulation.

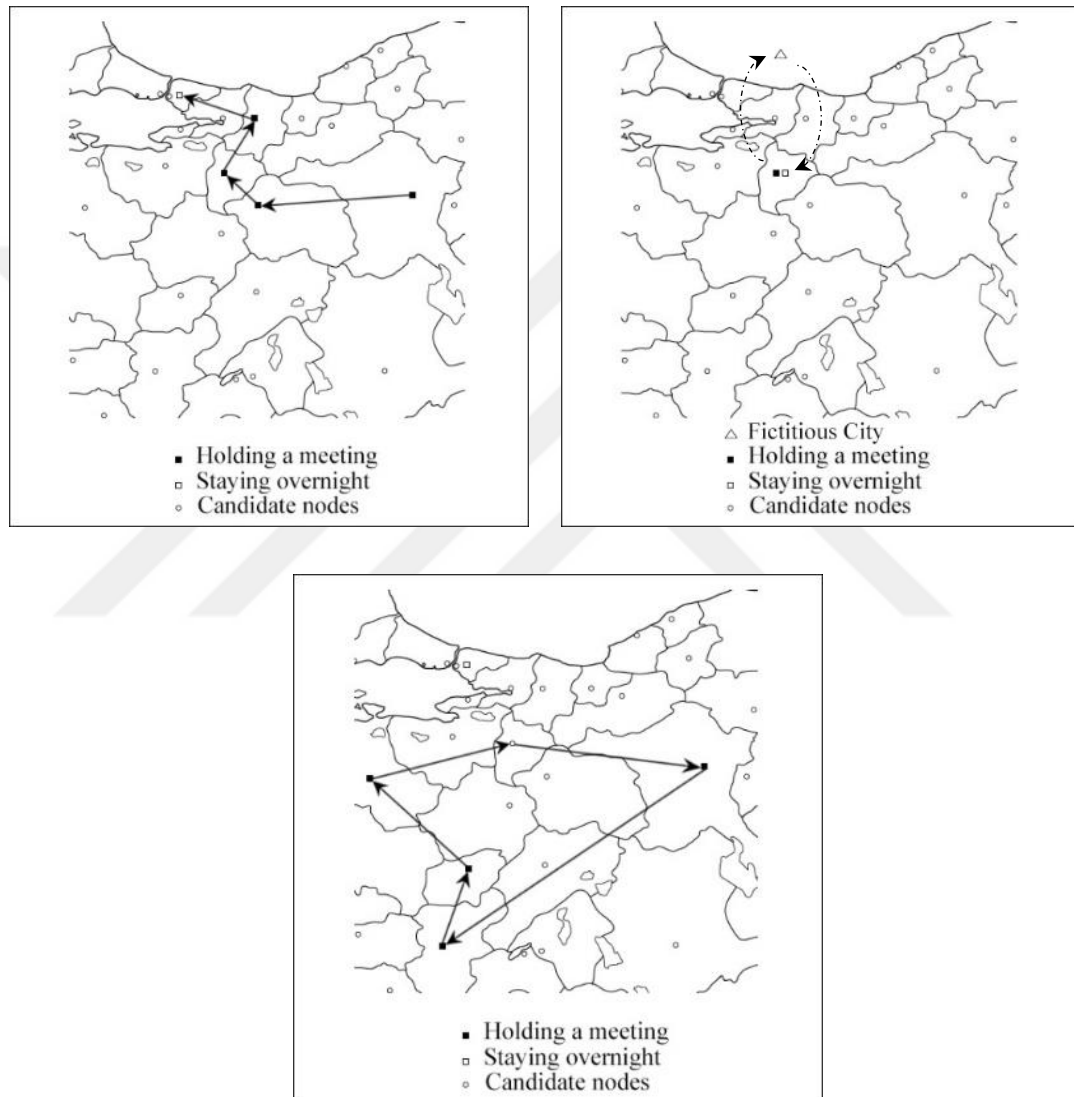


Figure 5.1 Different tours in the first three scenarios.

Flow balance equations and tight bounds on the flow variables are incorporated into the new model, which serve as subtour elimination constraints. They replace the ML-MTZ inequalities (3.26)-(3.33) in the proposed MPTPP model. The newly introduced binary variables E_{it} , L_{it} and S_{it} which keep track of open and closed tours are also dismissed. Consequently, the related coupling constraints that involve these variables become void.

Note that *Alt-1Depot* is a much more restrictive model than *Full-MILP* since it does not allow open tours and requires the politician to return to Ankara at the end of every day. Clearly, the optimal objective value of *Alt-1Depot* is a valid lower bound on *Full-MILP*.

5.2 Scenario analysis level 2: Alternative reward function

Thus far we assumed that earlier meetings produce higher rewards. In an alternative scenario, we reverse the direction of the reward function such that the actual reward increases as we approach the election day, i.e. the end of the campaign period. In this case, the objective function formulation will change as follows:

$$\begin{aligned}
 \max. \text{ NET BENEFIT} = & \\
 & \sum_{i \in \mathbf{N}} \sum_{t \in \mathbf{T}} \frac{t + \tau}{\tau} \pi_i FM_{it} + \sum_{i \in \mathbf{N}} \sum_{t \in \mathbf{T}} \sum_{1 \leq s < t} \frac{s(t + \tau)}{k\tau^2} \pi_i R_{its} \\
 & - \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} \sum_{t \in \mathbf{T}} c_{ijt} X_{ijt}
 \end{aligned} \tag{5.2}$$

Chapter 6

THE PROPOSED TWO-PHASE MATHEURISTIC

The idea of using a matheuristic approach to tackle large-size instances is motivated by observing the results of the variable fixation scenario. In this scenario, for those instances where we were able to solve them to optimality, we convert the optimal values of the binary decision variables S_{it} , L_{it} , and E_{it} to parameters with the values equal to the optimal value. The decision variable S_{it} indicates if the politician stays overnight in the city i by the end of day t . Binary variable L_{it} indicates if the politician does not enter, but only leaves the city i in the day t where binary variable E_{it} indicates if the politician does not leave, but only enters the city i in day t . For the remaining instances, we convert the values of the best feasible solutions of the mentioned variables to parameters. After performing this conversion, we solve the model again. The results are presented in Table 6.1. Boldface figures point to proven optimality achieved by the commercial solver GUROBI.

We observe that the CPU time of the variable fixation scenario has been reduced significantly. It can be comprehended that the difficulty of this problem is much more related to the scheduling and accommodation part rather than the routing part. Therefore, we decided to design a two-phase method where scheduling and accommodation part of the problem is solved

out of the mathematical formulation. In such approach, the mathematical formulation will take care of the routing part only.

Table 6.1 Comparison of original MPTPP and Variable Fixation Scenario

Instance	MPTPP			Variable Fixation Scenario		
	Best	Gap(%)	CPU(s)	Best	Gap(%)	CPU(s)
15C7D	17240	0.0	551.3	17240	0.0	0.1
15C10D	18759	0.0	30458.5	18759	0.0	0.1
21C7D	19138	0.0	6705.3	19138	0.0	0.8
21C10D	21904	6.9	86400.0	21904	0.0	1.1
30C7D	29427	0.0	20670.3	29427	0.0	5.3
30C10D	35013	6.0	86400.0	35013	0.0	3.5
40C7D	30086	4.1	86400.0	30195	0.0	25.0
40C10D	36409	12.6	86400.0	36409	0.0	211.3
51C7D	41087	9.9	86400.0	41182	0.0	95.8
51C10D	45667	22.4	86400.0	45810	0.0	424.6

To solve MPTPP, we tested the model on small instances where the commercial solvers are able to find the optimal solution in a reasonable amount of time. To deal with large-size instances we propose a two-phase matheuristic, which is named Finding Daily Optimal Routes method (FDOR). This matheuristic consists of two primary components; a city selection and a route generation. The route generation phase utilizes an integer program to build optimal route among selected cities. FDOR is an integer programming based heuristic which decomposes the original mixed-integer linear programming formulation into as many subproblems as the number of days, where using each

subproblem depends on how frequently the campaign base is to be visited throughout the campaign duration. Therefore, for those days where the politician requires to visit the campaign center, FDOR model 1 (FDORM1) will be used and for remaining days FDOR model 2 (FDORM2) will.

The main idea in FDOR is to select a (sub)set of candidate cities to be solved by either FDORM1 or FDORM2 using one of the three *city selection* approaches, namely the deterministic approach, the greedy approach, and the pseudo-random approach. The high-level algorithm of FDOR is provided in Algorithm 1.

Algorithm 1 The high-level definition of FDOR

Do the following for each day of the planning horizon

Phase 1:

- (a) Sort the cities in the decreasing order of their updated rewards.
- (b) Choose λ cities using one of the following city selection strategies:
 - Deterministic City Selection (DCS): Select all available cities.
 - Greedy City Selection (GCS): Select top λ cities.
 - Pseudo-Random City Selection (PCS): Select λ cities pseudo-randomly.

Phase 2:

- (a) Solve a TSPP for the given cities of Phase 1:
 - FDORM1: Politician should stay overnight in the campaign base.
 - FDORM2: Regular days.
 - (b) Update the rewards.
-

We will explain these approaches in the next subsection. Once the candidate cities are selected for each day, FDOR solves a daily *Selective Prize Collecting Travelling Salesman Problem* (SPCTSP) using either FDORM1 or FDORM2. The detailed pseudo code of the FDOR is explained in Algorithm 2. The new notations are first provided below:

Additional Notation

\mathbf{C}_t :	Set of candidate cities for day $t \in \mathbf{T}$.
λ :	Number of candidate cities.
π_t :	Set of updated rewards of day $t \in \mathbf{T}$.
K :	The base reward depreciation coefficient.
η_t :	Depot (starting) node of day $t \in \mathbf{T}$.
φ_t :	Terminal (ending) node of day $t \in \mathbf{T}$.
γ :	Campaign base.
ω_i :	Number of meetings in city $i \in \mathbf{N}$ during campaign period.
s_i :	Number of days since the last meeting in city $i \in \mathbf{N}$.
S_t :	Solution of day t .
S^* :	Solution of the whole campaign.
$B(S_t)$:	The Net Benefit of solution S_t .
$B(S^*)$:	The total Net Benefit of the original problem.

Algorithm 2 The pseudo code of FDOR

Initialization:

```

1:  $S^* = \emptyset, B(S^*) = 0, \delta_t = \emptyset, \omega_i = 0$ 
2: Reward calculation:
3: For  $t = 1 : \tau$ 
4:   If  $t = 1$  Then
5:      $\pi_t \leftarrow \pi_i$  // Every city gets its own base reward.
6:      $\eta_t = \gamma$  // Campaign starts from campaign base.
7:   Else If
8:     If  $\omega_i = 0$  then // This is the first meeting in city  $i$ .
9:        $\pi_i = \pi_i \frac{\tau - t + 1}{\tau}$ 
10:    Else If // This is a repeated meeting in city  $i$ .
11:       $\pi_i = \pi_i \frac{\tau - t + 1}{\tau} \times \frac{s_i}{K\tau}$ 
12:     $\pi_t \leftarrow \pi_i$ 
13:    End If
14:     $\eta_t = \varphi_{t-1}$  // Depot node of day  $t$  is equal to terminal node of day
     $t - 1$ .
15:  End If
16: Phase 1:
17:  $\mathbf{C}_t \leftarrow \text{City Selection Approach}(\lambda, \pi_t)$  // Select  $\lambda$  cities from  $\mathbf{N}$ .
18: Phase 2:
19: If  $\gamma \notin \{\varphi_{t-1}, \varphi_{t-2}, \dots, \varphi_{t-\kappa}\}$  Then // Force the politician to visit  $\gamma$  as a
    terminal node.
20:    $\text{FDORM1}(\eta_t, \mathbf{C}_t, \pi_t, \gamma) \rightarrow B(S_t), S_t, \varphi_t, \omega_i$ 
21: Else If // Solve a SPCTSP.
22:    $\text{FDORM2}(\eta_t, \mathbf{C}_t, \pi_t) \rightarrow B(S_t), S_t, \varphi_t, \omega_i$ 

```


23: **End If**

24: $S^* \leftarrow S_t$

25: $B(S^*) \leftarrow B(S^*) + B(S_t)$

26: **End For**

27: Return $B(S^*)$ and S^* as the best objective value and the best feasible solution of MPTPP, respectively.

28: **Output:** A feasible solution comprised of τ daily tours.

Algorithm 2 describes the components of FDOR. Updated rewards and number of meetings in each city are initialized as zero. Afterward, the reward of each city is calculated by taking into account the current meeting day t and the recency of previous meetings which may have been held before day t . Once rewards of all cities are updated, one of three *city selection* methods is called to select a subset of cities to be considered for the second phase.

As discussed, in FDOR, we develop two mathematical formulations and call them iteratively to solve daily SPCTSPs. The first model is called when the politician needs to return to the campaign base. The second model is called on regular days where the politician is free to start from and end up the daily tours in any node. FDORM1 is designed to generate daily routes where the starting city is any city including campaign base (the city that politician needs to visit every κ days) and the campaigner is required to stay overnight in campaign base as well at the end of that day. The routes of such days can be either an open tour or closed tour. FDORM2, on the other hand, is developed for those days where the politician is not required to return the campaign base.

The feasibility of the solution is guaranteed both with respect to the Maximum Tour Duration constraint, the Maximum Single Trip Time and also

with respect to the Maximum count of daily meetings. In the sequel, we present a mixed integer linear programming (MILP) formulation for FDORM1 and FDORM2.

6.1 Mathematical formulation of FDORM1 and FDORM2

Decision variables:

- X_{ij} : Binary variable indicating if arc (i,j) is traversed, where $X_{ii} = 0$.
 Z_i : Binary variable indicating if city i hosts a meeting.
 U_i : A continuous nonnegative variable used in the lifted Miller-Tucker-Zemlin Subtour Elimination Constraints (referred to as MTZ inequalities) determining the order of visit for city i .

$$\max. \text{ Daily NET BENEFIT} = \sum_{i \in \mathbf{N}} \pi_i Z_i - \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} c_{ij} X_{ij} \quad (6.1)$$

Subject to:

$$\sum_{i \in \mathbf{N}} \sigma_i Z_i + \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} d_{ij} X_{ij} \leq T_{\max} \quad (6.2)$$

$$\sum_{j \in \mathbf{N}} X_{ij} = \sum_{k \in \mathbf{N}} X_{ki} \quad i \in \mathbf{N}, i \neq \eta, \gamma \quad (6.3)$$

$$\sum_{j \in \mathbf{N}} X_{j,i} \leq 1 \quad i \in \mathbf{N} \quad (6.4)$$

$$\sum_{j \in \mathbf{N}} X_{i,j} \leq 1 \quad i \in \mathbf{N} \quad (6.5)$$

$$U_j \geq U_i + X_{ij} - (\alpha + 1)(1 - X_{ij}) \quad i, j \in \mathbf{N}, j \neq \eta \quad (6.6)$$

$$U_i \leq \sum_{j \in \mathbf{N}} \sum_{k \in \mathbf{N}} X_{jk} + 1 \quad i \in \mathbf{N} \quad (6.7)$$

$$0 \leq U_i \leq (\alpha + 1)Z_i \quad i \in \mathbf{N} \quad (6.8)$$

$$U_\eta = 1 \quad (6.9)$$

$$\sum_{j \in \mathbf{N}} X_{j,\eta} + \sum_{j \in \mathbf{N}} X_{\gamma,j} = 0 \quad (6.10)$$

$$\sum_{j \in \mathbf{N}} X_{\eta,j} + \sum_{j \in \mathbf{N}} X_{j,\gamma} = 2 \quad (6.11)$$

$$Z_i \leq \sum_{j \in \mathbf{N}} X_{ji} \quad i \in \mathbf{N}, i \neq \eta \quad (6.12)$$

$$\sum_{i \in \mathbf{N}} Z_i \leq \alpha \quad (6.13)$$

$$X_{ij} \in \{0,1\} \quad i, j \in \mathbf{N} \quad (6.14)$$

$$Z_i \in \{0,1\} \quad i \in \mathbf{N} \quad (6.15)$$

$$U_i \geq 0 \quad i \in \mathbf{N} \quad (6.16)$$

In the above formulation, the objective function (6.1) maximizes the net benefit of a tour while deducting travel costs from collected rewards. Constraint (6.2) ensures that the length of the tour does not exceed the maximum tour duration. The set of constraints (6.3) guarantee that if the politician enters any city, except the depot and the campaign base, he/she should leave there. Constraints (6.4) and (6.5) are typical selective TSP inequalities which impose the incoming and outgoing degree of each node. The set of constraints (6.6) and (6.7) are node-based MTZ sub-tour elimination constraints (Miller, et al., 1960). The lower bound and upper bound of continuous variable U are determined in constraints (6.8) and (6.9). Equalities (6.10) and (6.11) force the politician to leave the depot and to stay overnight in the campaign base. The inequalities (6.12) couple binary decision variable Z and X and ensure that

there will be no meeting in non-visited cities. Such a definition results in holding a meeting in every city that politician enters, except depot. Constraint (6.13) ensures that there will be no more than α meetings. Finally, binary integrality and nonnegativity constraints on the respective decision variables are defined in (6.14) – (6.16).

Compared to the original formulation of the MPTPP, FDORM1 is a significantly easier problem. Since the day of the meetings and the recency of them are calculated in the reward calculation step of FDOR, there is no need to include extra binary decision variables like FM and R (in the original formulation of the MPTPP) to capture either first or repeated meetings. Also, the starting node of each period is known due to the fact that the terminal node of the previous period is known. Therefore, there is no need for binary decision variables L , E and S (in the original formulation of the MPTPP) to track the terminal node of the previous day. Excluding these variables results in a simple yet effective model.

The mathematical formulation of FDORM2 is similar to FDORM1 except for that constraints (6.10) and (6.11) are replaced by:

$$\sum_{j \in \mathbf{N}} X_{\eta, j} = 1 \quad (6.17)$$

$$\sum_{j \in \mathbf{N}} X_{ij} \leq \sum_{k \in \mathbf{N}} X_{ki} \quad i \in \mathbf{N}, i \neq \eta \quad (6.18)$$

$$\sum_{j \in \mathbf{N}} X_{ij} + \sum_{j \in \mathbf{N}} X_{ji} - 1 \leq Z_i \quad i \in \mathbf{N} \quad (6.19)$$

Constraint (6.17) ensures that the politician leaves the depot. Constraints (6.18) allow the model to generate either an open tour or a closed tour. Finally,

the constraint (6.19) couples binary decision variables X and Z . In the formulation of both FDORM1 and FDORM2, we couple binary decision variables X and Z using equations (6.12).

FDORM1 has n^2 binary variables, n continuous variables, and $2n^2 + 7n + 2$ constraints. FDORM2 has n^2 binary variables, n continuous variables, and $2n^2 + 9n$ constraints. Such a decomposition results in a significant reduction in terms of the number of variables and the number of constraints as shown in Table 6.2.

Table 6.2 Reduction of original MILP formulation by using FDORM1 and FDORM2

	Original Formulation	FDORM1	FDORM2
# of binary variables	$n^2\tau + \frac{1}{2}n\tau^2 + \frac{11}{2}n\tau + 3\tau$	n^2	n^2
# of continuous variables	$(n+1)\tau$	n	n
# of constraints	$\frac{1}{6}n\tau^3 + 2n\tau^2 + \frac{3}{2}n^2\tau + \frac{62}{3}n\tau + \frac{45}{2}\tau + \frac{1}{2}\tau^2 - n^2 - 3n - \mathbf{K} - 4$	$2n^2 + 7n + 2$	$2n^2 + 9n$

Compared with the number of decision variables and constraints in the original formulation of the problem, FDORM1 and FDORM2 are relatively and significantly easier problems to solve. Such a reduction is achieved by

decomposing the original formulation of the problem into as many subproblems as the number of the planning time horizon.

6.2 City selection approaches

We investigated the following three selection strategies:

- (i) Simply choose $\lambda = |\mathbf{N}|$, (i.e. $\mathbf{C}_t = \mathbf{N}$ and $|\mathbf{C}_t| = n, t \in \mathbf{T}$.)
- (ii) Select the λ cities with the highest updated rewards.
- (iii) Pseudo-randomly select the λ cities based on their updated reward.

Choosing an appropriate set of the cities for the second phase of the algorithm is the most important step in FDOR to produce a high-quality solution. The most straightforward strategy would be selecting all cities in \mathbf{N} ; However, by increasing the number of cities, this strategy may increase the computational complexity of the second phase significantly. Another strategy can be sorting all the cities in \mathbf{N} in the decreasing order of their updated rewards at the beginning of each period. Next, select top λ cities from this list and pass it to the second phase. Such a method requires a sensitivity analysis of choosing an appropriate value for the parameter λ . The last strategy would be a pseudo-random city selection approach, where λ cities are selected from the list of the cities depending on their updated reward. Below we explain mentioned strategies in details.

6.2.1 Deterministic City Selection

In Deterministic City Selection (DCS) approach, all cities in the set \mathbf{N} are selected to be considered in the second phase of the FDOR at every iteration. Therefore, the value of the parameter λ is equal to n . Consequently, the *City Selection Approach*(λ) returns \mathbf{C}_t where $\mathbf{C}_t = \mathbf{N}$ and $|\mathbf{C}_t| = n$.

6.2.2 Greedy City Selection

The main idea in Greedy City Selection (GCS) approach is to sort all cities in the decreasing order of their updated reward in phase 1. Once the sorted list of the cities is generated, the algorithm select top λ cities from this list. Algorithm 3 presents the pseudo-code of GCS.

Algorithm 3 The pseudo code of GCS

New

Notation *Definition*

δ'_t : List of the cities in the decreasing order of their reward.

Input: λ, δ_t

Output: \mathbf{C}_t

- 1: $\delta'_t \leftarrow$ Sorted list of the cities in \mathbf{N} in the decreasing order of their reward
- 2: $\mathbf{C}_t = \{\}$
- 3: **while** $|\mathbf{C}_t| \leq \lambda$ **do**
- 4: $i = \delta'_t[1]$
- 5: $\mathbf{C}_t \leftarrow i$
- 6: $\delta'_t = \delta'_t / \{i\}$
- 7: **end while**

The value of the parameter λ , which indicates the number of the cities that should be selected from the sorted list of the cities at the beginning of each day, is set to 15. We came up with this value by performing a comprehensive sensitivity analysis of parameter λ . The results of the sensitivity analysis are provided in the section 8.

6.2.3 Pseudo-random city selection

Selection of the cities in GCS approach is based on their updated reward, which is a logical selection criterion that results in choosing highly-rewarded cities at each iteration. On the other hand, due to the dynamic property of the objective function, which makes the rewards time-dependent and recency dependent, such a selection may not result in finding the optimal solution.

Obviously, the random selection of the cities does not result in achieving an optimal or a high-quality solutions as well. It may even lead to an infeasible solution, where the travel times of all cities from depot violate the maximum tour duration constraint. One moderate approach to consider these challenges is to select cities pseudo-randomly. Algorithm 4 presents the pseudo-code of Pseudo-random city (PCS) selection.

Algorithm 4 The pseudo code of PCS

<i>New</i>	
<i>Notation</i>	<i>Definition</i>
p_i :	The weighted probability of city i .
P :	Array of weighted probabilities of cities.
ε :	a random number between 0 and $\sum_{i \in \mathbf{N}} p_i$
Input:	λ, δ_t
Output:	\mathbf{C}_t
1:	$\mathbf{C}_t = \{\}$
2:	for $i = 1 : n$ do
3:	$p_i = \frac{\delta_i}{\sum_{i \in \mathbf{N}} \delta_i}$
4:	$P \leftarrow p_i$
5:	end for
6:	while $ \mathbf{C}_t \leq \lambda$ do
7:	$i \leftarrow \text{Pseudo-random selection}(\rho, P)$
8:	$\mathbf{C}_t \leftarrow i$
9:	$\varepsilon \leftarrow \text{uniform}(0, \sum_{i \in \mathbf{N}} p_i)$
10:	end while

The PCS approach randomly selects λ cities from \mathbf{N} where each city has a known probability of selection. All probabilities of the cities together sum to 1. Once the probability vector is generated, PCS computes the discrete cumulative density function (CDF) of rewards which corresponds to the vector of cumulative sums of the rewards. Next, a random number in the range

between 0 and the sum of all probabilities (in this case, 1) is generated. The corresponding value of this random number in the discrete CDF array is the weighted random city.

If PCS is used in the city selection step of the FDOR, we need to repeat the algorithm for *iter* times and pick the best solution and objective value to ensure solution diversity of FDOR. The necessary modifications in the main body of the FDOR are shown in Algorithm 5.

Algorithm 5 The pseudo code of FDOR–PCS

<i>New</i>	
<i>Notation</i>	<i>Definition</i>
Z_{all} :	Set of objective values.
S_{all} :	Set of all solutions.
S_{iter} :	Solution of the $iter^{th}$ iteration.
$Z(S_{iter})$:	Objective value of the $iter^{th}$ iteration.
Output:	A feasible solution comprised of τ daily tours.
1:	$\delta_t \leftarrow$ Reward calculation
2:	while $iter < Max_Iter$ do
3:	for $t = 1 : \tau$
4:	$\mathbf{C}_t = \{\}$
5:	Phase 1:
6:	$\mathbf{C}_t \leftarrow PCS(\lambda, \delta_t)$
7:	Phase 2:
8:	if $\varphi_{t-1}, \varphi_{t-2}, \dots, \varphi_{t-\kappa} \neq \gamma$ then
9:	FDORM1($\eta_t, \mathbf{C}_t, \delta_t, \gamma$) $\rightarrow Z(S_t), S_t, \varphi_t, \omega_i$

```

10:         else if
11:             FDORM2( $\eta_t, \mathbf{C}_t, \delta_t$ )  $\rightarrow Z(S_t), S_t, \varphi_t, \omega_i$ 
12:         end if
13:          $S_{iter} \leftarrow S_t$ 
14:          $Z(S_{iter}) += Z(S_t)$ 
15:     end for
16:      $Z_{all} \leftarrow Z(S_{iter})$ 
17:      $S_{all} \leftarrow S_{iter}$ 
18:      $iter ++$ 
19: end while
20:      $Z(S^*) = \text{Max}(Z_{all})$ 
21:      $S^* = S_{iter}$ , where  $Z_{iter} = Z(S^*)$ 
22:     Return  $Z(S^*)$  and  $S^*$  as the best objective value and the best feasible
        solution of MPTPP, respectively.

```

The FDOR-PCS method generates $iter$ solutions and pick the best one as the ultimate output of the algorithm. The value of $iter$ is set to 5 in our experiments. We will explain the computational results in Section 8.

Chapter 7

A GRANULAR SKEWED VARIABLE NEIGHBORHOOD TABU SEARCH

The roaming salesman problem is a generalization of the well-known traveling salesman problem (TSP). Garey and Johnson (1979) prove that TSP is a strongly \mathcal{NP} -hard combinatorial optimization problem. The roaming salesman problem (RSP) is also \mathcal{NP} -hard since it is more complex than the traditional TSP. Therefore, it cannot be solved in polynomial time to optimality. Motivated by this challenge, we propose a hyper-heuristic algorithm that can solve the large size instances of the problem in a reasonable amount of CPU time. When we survey the literature, we observe that the variable neighborhood search (VNS) algorithm is very successful in solving routing problems, see Polacek et al. (2007), Liu et al. (2009), Polacek et al. (2008), Hemmelmayr et al. (2009), Polat et al. (2015), Sarasola et al. (2016), and Todosijevic et al. (2017). In most of these studies, an extended version of VNS is used, rather than using a basic version of VNS. Since the RSP is a large-scale optimization problem, we are motivated to propose an extended VNS complemented with a Tabu Search (TS) algorithm. Furthermore, to the best of our knowledge, the VNS has not been applied to the RSP in the literature since it is a new problem.

This section describes the primary steps of the implemented algorithm. First, we explain the basic VNS. Then, we introduce a greedy algorithm to construct an initial feasible solution and compare it with the proposed two-phase matheuristic. Next, the general framework of the proposed approach is explained. Finally, the granularity and the skewness used in the approach is discussed.

7.1 Variable Neighborhood Search

The variable neighborhood search (VNS) was proposed by Mladenović and Hansen (1997). It is a metaheuristic approach which is applied to different combinatorial optimization problems. The basic idea in VNS is to change the neighborhoods in a systematic way within a local search procedure. It searches for the best solution among different neighborhood structures. VNS method heavily relies upon the following fact:

Fact 1 A local optimum of one neighborhood structure is not necessarily a local optimum for another neighborhood structure.

Fact 2 A global optimum is the local optimum with respect to all neighborhood structures.

Fact 3 In many combinatorial optimization problems, the local optima with respect to one or multiple neighborhoods are fairly close to each other.

Let us denote with $N_k, (k = 1, \dots, k_{\max})$, a set of pre-defined neighborhood structures, and with $N_k(x)$, a set of solutions in the k^{th} neighborhood of x .

Unlike many local search heuristics where $k_{\max} = 1$, VNS uses multiple neighborhood structures.

Once the neighborhood structures are determined, the main steps of the algorithm start. Starting from any initial feasible solution, a random solution is generated in the first step, which is called shaking. This is followed by applying a local search method. Once a new incumbent solution is obtained, the procedure starts with the first neighborhood again; otherwise, the local search is performed with the next neighborhood structure.

Typically, the neighborhoods are nested, which means the next neighborhood is larger than the previous one and it contains the previous one. The pseudo code of the basic VNS is shown in Figure 7.1, where $N_k, (k = 1, \dots, k_{\max})$, is the set of pre-selected neighborhoods. Figure 7.1 illustrates the basic VNS scheme.

The stopping criteria can be an explicit time limit, a threshold on the number of iterations, or a limit on the number of iterations without improvements. Interested readers are advised to read Mladenović and Hansen (1997) and Hansen and Mladenović (2001) for a more detailed explanation of the basic VNS.

Initialization. Select the set of neighborhood structures $N_k, (k = 1, \dots, k_{\max})$,
that will be used in the search;
Find an initial solution x ;
Choose a stopping condition;

Repeat the following until the stopping condition is met :

- (1) Set $k \leftarrow 1$;
- (2) Repeat the following steps until $k = k_{\max}$:
 - (a) Shaking. Generate a point x' at random from k^{th} neighborhood of x
 $(x' \in N_k(x))$;
 - (b) LocalSearch. Apply some local search method with x' as initial
solution;
Denote with x'' the so obtained local optimum;
 - (c) Move or not. If this local optimum x'' is better than the incumbent,
or if some acceptance criterion is met , move there
 $x \leftarrow x''$,
and continue the search with $N_1 (k \leftarrow 1)$; otherwise,
set $k \leftarrow k + 1$;

Figure 7.1 Steps of the Basic VNS (Hansen and Mladenović, 2001)

“*Shaking*”, “*Local Search*”, and “*Moving*” are the main three blocks of the VNS. In the ‘shaking’ step, a random solution is generated from the neighborhood of the current solution. Next, the local search follows. In the Moving step, the objective value of the current solution is compared with the objective value of the incumbent solution. If there is an improvement, this solution is accepted and the algorithm goes back to the first step; otherwise,

the algorithm goes to the shaking step with using the next neighborhood structure.

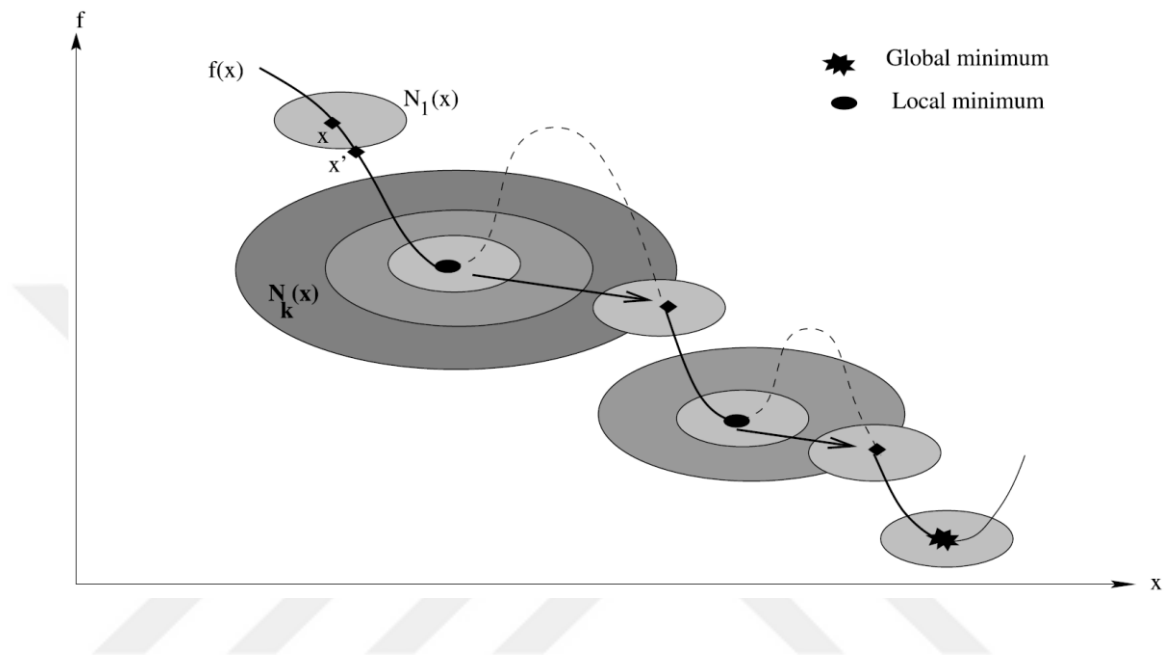


Figure 7.2 Basic VNS scheme (Hansen et al. 2010)

Our proposed approach consists of the following components:

- Initial solution construction,
- Neighborhood structures,
- Granular neighborhoods,
- Shaking procedure,
- Local search,
- Strategic oscillation,
- Acceptance decision criteria (Skewed moves),
- Termination criteria.

In the next subsections, we elaborate on the different components of the proposed VNS for the RSP.

7.2 Solution Representation

Since RSP involves both costs and rewards, we employ a vector-based representation which indicates the order of the visited cities in the sequence with the corresponding meetings. Specifically, we represent a solution in the VNS method in two two-dimensional arrays. The first $(\tau \times 1)$ array represents the order of the visits for the cities each day. For instance, $[5, 8, 1, 9]$ means the tour of the day is started with city 5. Then, city 8 is visited. Then, city 1 is visited. Finally, the tour is finished by visiting city 9. The second $(\tau \times 1)$ array represents the meetings held at each period. The order of the meetings within a day does not affect the collected rewards.

As an example, consider a campaign period with $n = 20$ and $\tau = 4$. The following matrices represent a solution of RSP:

$$a_{routes} = \begin{bmatrix} [6, 15, 5, 10] \\ [10, 4, 14, 9] \\ [9, 12, 1, 13] \\ [13, 8, 7, 19] \end{bmatrix}$$

$$a_{meetings} = \begin{bmatrix} [6, 15, 5] \\ [10, 4, 14] \\ [9, 12, 1, 13] \\ [8, 7, 19] \end{bmatrix}$$

a_{routes} indicates that the order of the visits for four days is $[6, 15, 5, 10]$, $[10, 4, 14, 9]$, $[9, 12, 1, 13]$, and $[13, 8, 7, 19]$, respectively. $a_{meetings}$ indicates

that the meetings of the first day are held in cities [6, 15, 5], the meetings of the second day is held in cities [10, 4, 14], etc.

7.3 Initial solution

We first propose a greedy approach to generate the initial feasible solution. After comparing the results of this greedy method with the two-phase matheuristic proposed in Section 6, we employ the latter one to construct an initial solution. The reason for choosing the two-phase matheuristic lies in the fact that the quality of the initial solution heavily affects the performance of the VNS (Hansen et al. 2010). Based on the experimental evidence, it can be said that a better initial solution significantly decreases the CPU time of the VNS. The details of the greedy approach are discussed in the next subsection. The initial feasible solution is represented with S_0 in the main algorithm.

7.3.1 Exhaustive search of the candidate cities

In this algorithm, the initial feasible solution is produced quickly and will be fed to the main loop of the VNS algorithm to make the improvements in the next steps. We seek to assign the cities to the days rapidly. The main idea is to assign the highly-rewarded cities to the early days of the campaign due to the characteristics of the reward function. To assign the cities to the days, we perform an exhaustive search. Therefore, we call this algorithm the *Exhaustive Search of the Candidate Cities for each day* (ESCC).

In ESCC, a sorted list of all cities is created where the sorting is with respect to the updated rewards of cities at the beginning of each day. Then, a feasible route is generated such that the maximum possible net benefit is

achieved considering limitations like maximum tour duration and necessity to return to the campaign base periodically. Then, a feasibility restoration function is utilized to make the solution feasible. The detailed pseudo code of the construction of the initial feasible solution using ESCC is described in Algorithm 6.

Algorithm 6 The pseudo code of ESCC

```

1: Initialization
2:  $T \leftarrow \tau$ 
3:  $\text{MaxSingleTripTime} \leftarrow 300$ 
4:  $\text{MaxTourDuration} \leftarrow T_{\max}$ 
5:  $\text{DailyMax} \leftarrow \alpha$ 
6:  $\text{WakeupCity}(1) \leftarrow \eta$ 
7: For  $t=1:T$ 
8:   If  $t=1$  Then
9:      $\text{Rewards} \leftarrow \pi$ 
10:  Else If
11:    Calculate the reward of each city by taking into account the current
    meeting day  $t$  and the recency of the previous meetings which
    may have been held before day  $t$ .
12:  End If
13:  Sort all cities eligible for hosting a meeting on day  $t$  in descending
    order of their rewards into the array  $\text{HRC}_t$ 
    // HRC stands for Highest Reward Cities
14:   $\text{WakeupCity}(t) \leftarrow i$  ;

```

```

15:   Top6={i};
16:   k←1;
17:   While |Top6|<6 Do
18:     j ← HRCi[k];
19:     If i=j OR tij>MaxSingleTripTime Then
20:       k←k+1;
21:       continue;
22:     End If
23:     Top6←Top6 ∪ {j};
24:     k←k+1;
25:   End While
26:   Step 2:
27:   Given Top6, permute all 5!=120 possible tours and select the one with
       the lowest total traveling cost. Break ties arbitrary.
           //each possible permutation represents an open tour for day t.
28:   Call the selected tour BestTour(t)=[1*,2*,3*,4*,5*,6*]
           // 1* is going to be the wakeup city i.
29:   If BestTour(t) is infeasible w.r.t. MaximumTourDuration then
30:     Crop BestTour(t) from its right and starting at city 6* until it
       becomes time-feasible
31:   End If
32:   Set WakeupCity(t+1)=last visited city in BestTour;
33: End For
34: Output: A feasible initial solution comprised of T daily tours where the tour
       of day 1 starts in  $\eta$ 

```

In the ESCC, we assume that there will be a meeting in every visited city of daily tour other than the wakeup city. The wakeup city of day t never hosts a meeting; However, being the sleeping city of day $(t - 1)$, it always hosts a meeting in day $(t - 1)$. We also assume that each daily tour is either of Type-1 or Type-3 tours. Therefore, no closed tour are allowed in this algorithm. As discussed, all cities but the wakeup city visited during a daily tour host a meeting. The feasibility of the ESCC solution is guaranteed with respect to the Maximum Tour Duration constraint, the Maximum Single Trip Time and also with respect to the Maximum count of daily meetings.

7.4 Neighborhood structures

In VNS method, the decision of choosing an appropriate neighborhood structure, and the way they are arranged play a significant role in the success of the algorithm. Apparently, the search space of neighborhoods influences the efficiency of the approach. Most probably, a large neighborhood includes the global optima while a small neighborhood may not cover the global optima. On the other hand, the computational effort increases while the neighborhoods are enlarged.

We conduct preliminary experiments to choose neighborhood structures where various neighborhood structures along with different sequences are tested. We employ eleven neighborhood structures for VNS algorithm: 1-Add, 1-Drop, Drop-Add, 1-1 Exchange Non-Visited, 1-1 Exchange Intra Route, 1-0 Relocate, 2-0 Relocate, 1-1 Swap, 2-2 Swap, 1-1-1 Swap (Triple Rotation) and 1-1-1-1 Swap (Quadruple Rotation). The first four neighborhoods resemble the selective nature of the problem, thus, are called city selection neighborhoods. The 1-1 Exchange Intra Route is called Intra-Route Neighborhood. The

remaining neighborhoods are called Inter-Route Neighborhoods since they include different routes of various days. In the following, we describe each neighborhood structure in details.

1-Add

Given a city not included in any route of the current solution, the city is inserted in the best position using the cheapest insertion approach. The city will be positioned in the first possible day as it may result in obtaining higher objective value. This position can be before the depot node, between two nodes, or after the terminal node. An example of 1-Add operator is depicted in Figure 7.3.

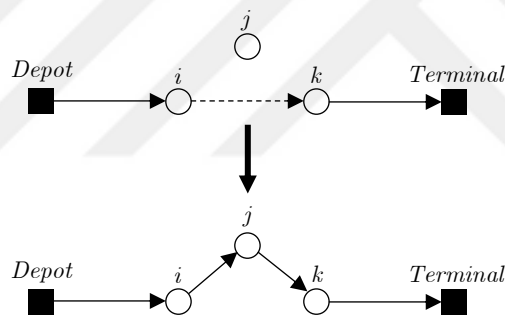


Figure 7.3 The 1-Add operator

1-Drop

Given a city included in the current solution, it is removed from the route. If this city is the last visited city of the campaign, it will be simply removed. If the city is a depot node, the chain feasibility of the whole tour ensures the connectivity between daily tours. If the city is a transient city, the predecessor and successor of this city are connected. The primary candidates for removal are the visited cities with the lowest reward. See Figure 7.4.

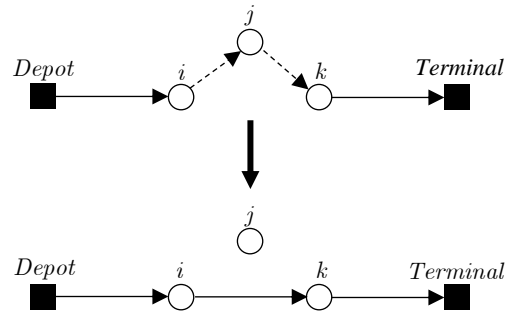


Figure 7.4 The 1-Drop operator

Drop-Add

Given a randomly-selected route, The Drop-Add operator drops one node from that route. Then, a non-visited city is added to another randomly-selected route. See Figure 7.8.

1-1 Exchange Non-Visited

Given a randomly-selected route, a node is selected at random and its position is interchanged with a non-visited city. This move is illustrated visually in Figure 7.6.

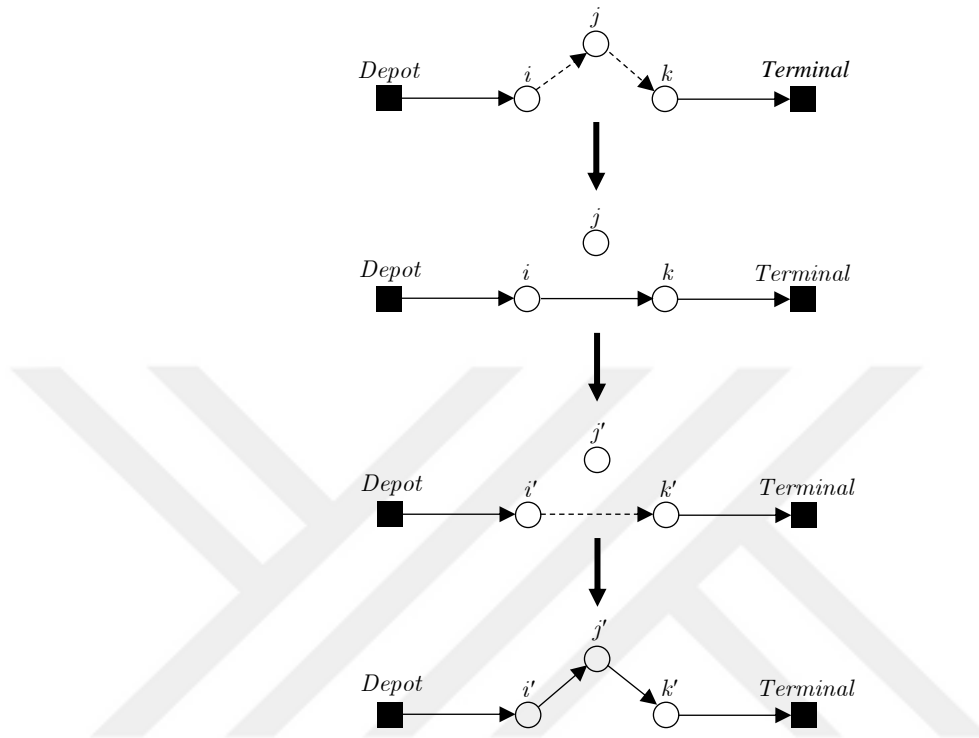


Figure 7.5 The Drop Add operator

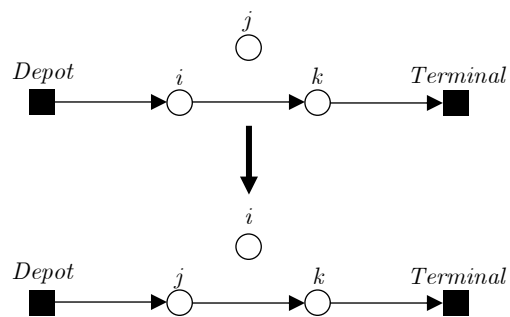


Figure 7.6 The 1-1 Exchange Non-Visited operator

1-1 Exchange Intra route

Given a randomly-selected route, two nodes are randomly selected and their positions are interchanged. The selected nodes can be depot node, terminal node, or transient node. See Figure 7.7.

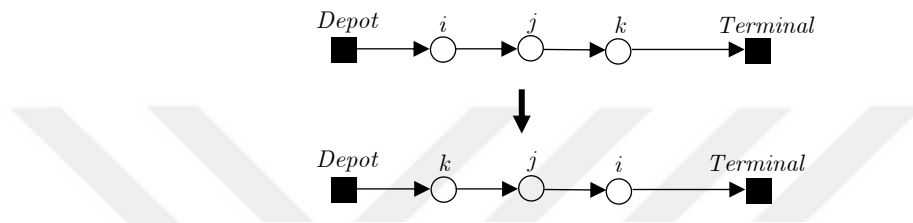


Figure 7.7 The 1-1 Exchange Intra Route operator

1-0 Relocate

In this operator, given a daily route, a randomly-selected city is relocated from one daily route to another daily route between two consecutive cities, as a depot node or as a terminal node. See Figure 7.8.

2-0 Relocate

In this move, a randomly-selected node and its successor are removed from their positions and inserted in a different route. This move is represented visually in Figure 7.9.

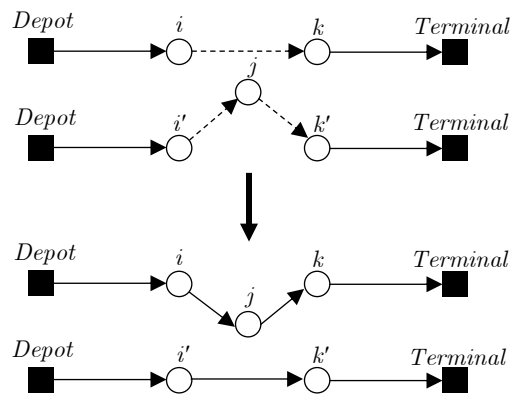


Figure 7.8 The 1-0 Relocate operator

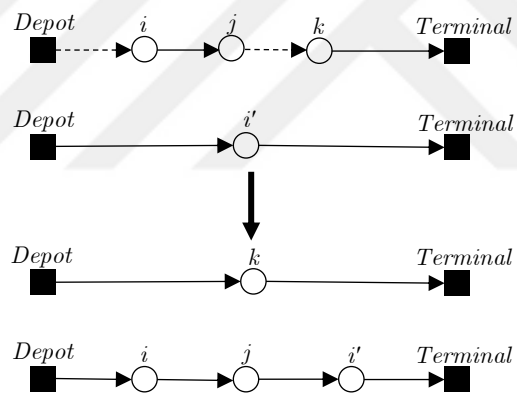


Figure 7.9 The 2-0 Relocate operator

1-1 Swap

Given two cities from two different daily routes, their positions are swapped.

Figure 7.10 illustrates this move.

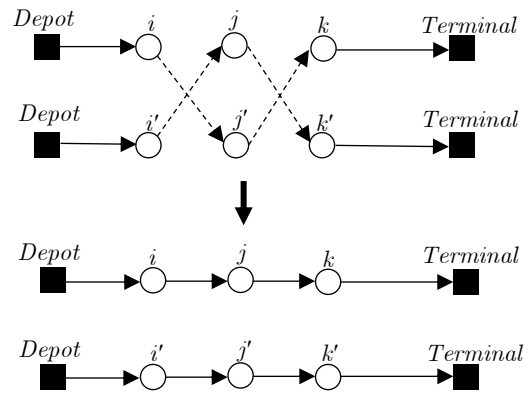


Figure 7.10. The 1-1 Swap operator

2-2 Swap

Given two cities on different daily routes, the positions of the first city and its successor are exchanged by the second city and its successor. 2-2 Swap is represented visually in Figure 7.11.

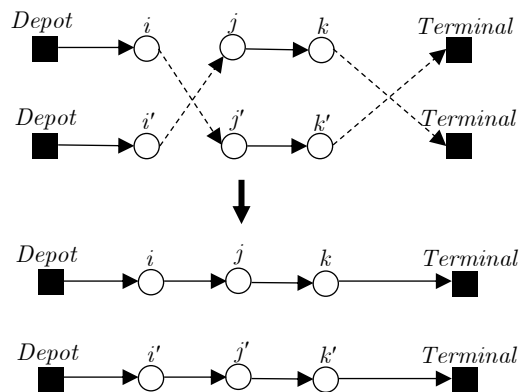


Figure 7.11 The 2-2 Swap operator

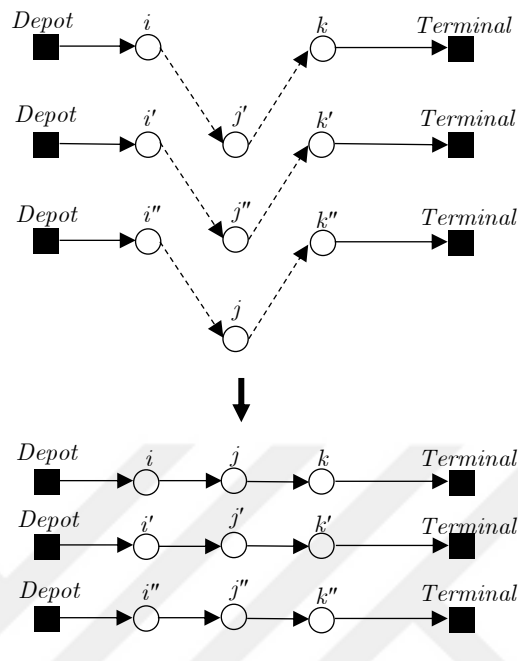


Figure 7.12 The Triple Rotation operator

Triple Rotation (1-1-1 Swap)

Given three cities in different daily routes, it executes a sequence of three moves where every city is relocated to the location of the next in line. See Figure 7.12.

Quadruple Rotation (1-1-1-1 Swap)

Given four cities in different daily routes, it executes a sequence of four moves where every city is relocated to the location of the next in line. See Figure 7.13.

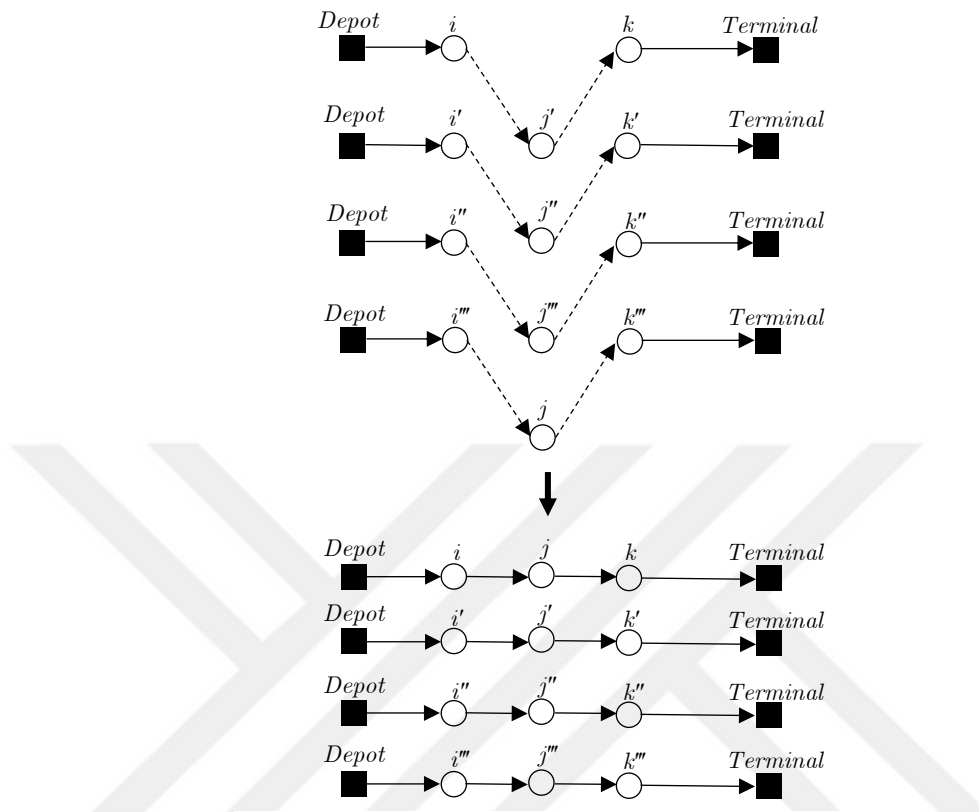


Figure 7.13 The Quadruple Rotation operator

7.5 Granular neighborhoods

The neighborhoods of the GSVNTS algorithm is fairly large and exploration of all of them leads to a cumbersome task. GSVNTS requires to explore all discussed neighborhood structures at each iteration for obtaining a high-quality solution. Such an exploration inevitably increases the CPU time of the algorithm. Toth and Vigo (2003) introduce a successful method to reduce computing time while exploring the neighborhoods. They suggest the granular neighborhoods that speed up the search procedure severely. The main idea is

to explore a certain subset of promising moves while a significant number of unpromising moves are eliminated. In our implementation of GSVNTS, we capitalize on the similar idea with some modifications.

Granular Neighborhoods reduce the size of the candidate neighborhoods by depriving non-promising moves during the search procedure. In order to reduce the computing time of neighborhood search in the shaking procedure and in the local search procedure, we proposed two effective strategies, which results in a significant reduction of the computing time.

a) Tabu Moves

While inserting a city into a daily route, if the daily tour is a closed route:

Calculate the distance of the candidate city i for insertion in terms of travel time with the depot j (t_{ij}).

If $t_{ij} \leq \psi$ **then**

CONTINUE;

Else

Try another city;

Where ψ is the *Tabu Move Threshold*. If the travel time is higher than *Tabu Move Threshold*, ignore this move, as this insertion will most probably lead to a time infeasible solution.

If the daily tour is an open tour:

Calculate the minimum distance of the candidate city i for insertion in terms of travel time with the depot node j and terminal node k .

IF $\min\{t_{ij}, t_{ik}\} \leq \psi$ **Then**

CONTINUE;

Else

Try another city;

If $\min\{t_{ij}, t_{ik}\}$ is higher than the *Tabu Move Threshold*, ignore this move, as such an exchange will most probably lead to a time infeasible solution.

b) Permanent Tabu Matrix

While swapping two cities in different daily tours, it is important to consider the time- and the cost-efficiency of this exchange. For instance, in the daily route of “Istanbul – Bursa – Balıkesir”, the “Bursa \leftrightarrow Adana” swap is not a promising move because it is too far from the candidate route. Therefore, we developed an Adjacency Matrix to handle similar scenarios. First, we develop a two-dimensional adjacency matrix with 0 – 1 values. In this matrix, city i and j are adjacent if and only if there is not any candidate city k that can be inserted between i and j . Therefore, most of the neighbor cities are considered as adjacent. If city k is close enough to both city i and j such that there is a possibility of generating sub-route “ $i - k - j$ ”, then city i and j are considered as non-adjacent cities. The symmetric two-dimensional adjacency matrix is significant as it is the basis of developing three-dimensional “Matrix of Eligibility for Insertion”.

Next, a three-dimensional adjacency matrix with 0 – 1 values is developed to evaluate all possible insertion patterns. Let us call the two-dimensional matrix A and that three-dimensional matrix M . The purpose is to keep track of promising and eligible insertions of cities in between a pair of other cities. Cell $M[i, j, k]$ of the three-dimensional matrix will be 1 if and only if city i is eligible to be inserted between cities j and k . The primal property of the three-dimensional matrix is being symmetric with respect to j and k ,

i.e. $M[i,j,k] = M[j,k,i]$ for each arc (j,k) . We calculate the values of this matrix as follows.

$$M[i,j,k] = 1 \text{ if } \min\{t_{ij}, t_{jk}\} \leq \text{Granularity Threshold Value}$$

$$A[j,k] = 0, \text{ 0 otherwise.}$$

Considering the mentioned rule, city i can be inserted into the arc of (in between) cities j and k if and only if both of the following conditions are satisfied. The parameter $t_{i,j}$ shows the symmetric travel time between cities i and j .

- i. Cities j and k are known to be nonadjacent. This is checked via two-dimensional symmetric adjacency matrix at this point.
- ii. City i which is a candidate for insertion is close enough to at least one of the depots of the edge (j,k) .

The benefit of the three-dimensional matrix of eligibility for insertion is to speed up the neighborhood search within the GSVNTS algorithm by a great deal. Thanks to the two-dimensional matrix A and three-dimensional matrix M , we will eliminate unnecessary and non-promising moves to progress from the current solution to the next solution in the steps of the GSVNTS.

For instance, we do not consider inserting Istanbul in between Kırklareli and Tekirdağ. Although Istanbul is close enough to both of them, it cannot be cost- and time-saving to visit Istanbul while going from Kırklareli to Tekirdağ, because Kırklareli and Tekirdağ are adjacent cities. The same logic applies while inserting Hatay in between Antalya and Alanya.

Figure 7.14 represents another example of granular moves. Suppose we want to exchange Balıkesir with another city. We eliminate the move Balıkesir–

Van since the two cities are too far from each other. On the other hand, we permit the evaluation of Balıkesir – Tekirdağ move as it seems a promising move.



Figure 7.14 An example of granular neighborhoods

7.6 Shaking

Shaking procedure in the proposed GSVNTS method is responsible for generating a new starting point for the local search procedure. A neighborhood of an incumbent solution is determined by an operator. On the one hand, the goal of the operator is to adequately perturb the solution while keeping the good segments of it. In the basic VNS, one random solution is obtained by shaking procedure; however, in GSVNTS, multiple random solutions are generated and the best solution among them is selected to be used as an initial point of the local search procedure. This is basically the diversification strategy of our method.

The set of neighborhoods used for shaking is not necessarily the same as the local search neighborhoods. The neighborhoods used in shaking procedure must normalize the trade-off between keeping good variables of the incumbent solution and perturbing it. We consider the city selection neighborhoods for the shaking procedure. We explained the details of the neighborhoods in Section 7.4. Such neighborhoods allow the shaking procedure to explore solutions far from the incumbent more and more.

7.7 Local Search

The generated solution by the shaking is submitted to the local search approach to obtain a locally optimal solution. Recently, several local search procedures have been introduced to the routing literature which extend the scheme of iterative improvement in various ways and avoiding being trapped in a local optimum. The most well-known research of these methods can be found in Holland (1975), Kirkpatrick et al. (1983), Glover (1989,1990), Glover and Laguna (1998).

The local search procedure used in the second step of GSVNTS by carrying out a sequence of local changes in the solution obtained from the shaking procedure. The local search improves the objective value of this solution each time until a local optimum is found. To this end, an improved solution x' in the neighborhood $N(x)$ of the current solution x is obtained at each iteration, until no more improvement is obtained.

In many local search approaches, non-improving moves are allowed to avoid termination at a local minimum. Such a move is called hill climbing. However, non-improving moves increase the risk of cycling. Tabu Search (TS) uses a short-term memory to prevent moves that might result in revisiting the

recently-explored solutions. The basic VNS tries to escape from local optimum by changing the neighborhood structures as well.

We incorporate the TS into both local search and shaking steps to regulate the intensification and diversification of the approach. The local search examines all the non-tabu moves. The shaking procedure consists of applying random moves. The shaking method is allowed to generate infeasible solutions. Then, the local search is permitted to make the best non-tabu move among those that decrease the infeasibility of the solution. Hence, the local search consists of selecting the best non-tabu move.

The tabu search used in GSVNTS is based on the neighborhoods defined in Section 7.4 and 7.5. The pseudocode of the local search procedure is provided in Algorithm 7.

We describe the details of the algorithm in the following. A maximum of $iter_max$ iterations are done during this search. The tabu search works with neighborhood N_k where k is provided by the outer loop of the GSVNTS. Once the complete neighborhood N_k is constructed, the best solution of N_k is given. Next, the tabu status of this solution is checked. If the solution is a tabu, the second best solution from N_k is extracted. This process continues until all solutions from N_k being examined; otherwise, the best-known solution is used as the new solution.

Algorithm 7 Local search based on TS with steepest descent strategy

Input:

$iter_max$

S_0

$N_k(S_0)$

1. $iter \leftarrow 0$;
2. $Improve = true$;
3. **While** $Improve = true$ or $iter < iter_max$ **Do**
4. $Improve = false$;
5. $iter \leftarrow iter + 1$;
6. $Tabu_List \leftarrow null$;
7. Select the best solution S^* from $N_k(S_0) \setminus Tabu_List$;
8. **If** $Z(S^*) > Z(S_0)$ **Then**
9. $S \leftarrow S^*$;
10. Update $Tabu_List$;
11. $Improve = true$;
12. **End If**
13. **End While**
14. **Return** S^*
15. **End**

Our local search integrates the tabu conditions to avoid cycling. Such conditions enforce the search procedure to explore different regions of the solution space. These tabu tools are more powerful than the usual tabu moves since by considering a short list of tabu elements, a significant number of moves are forbidden. To this end, we create the tabu conditions for every neighborhood structure once the current solution is updated. The tabu conditions of the neighborhood structures are explained below:

1-Add: The added city cannot be dropped from the routing/meeting schedule.

1-Drop: The dropped city cannot be added to the routing/meeting schedule.

Drop-Add: The dropped city cannot be added to the routing/meeting schedule and the added city cannot be dropped from the routing/meeting schedule.

1-1 Exchange Non-Visited: Visited and non-visited cities whose positions are changed by 1-1 Exchange Non-Visited cannot be re-interchanged by the same move.

1-1 Exchange Intra Route: Cities on the same day whose positions are swapped by 1-1 Exchange Intra Route cannot be re-swapped by the same move.

1-0 Relocate: A city whose position is changed by 1-0 Relocate cannot be dislocated by the same move.

2-0 Relocate: The cities whose positions are changed by 2-0 Relocate cannot be dislocated by the same move.

1-1 Swap: Cities whose positions are exchanged by 1-1 Swap cannot be re-swapped by the same move.

2-2 Swap: The swapped chain of two cities by 2-2 Swap cannot be re-swapped by the same move.

1-1-1 Swap (Triple Rotation): The swapped chain of three cities by 1-1-1 Swap cannot be re-swapped by the same move.

1-1-1-1 Swap (Quadruple Rotation): The swapped chain of four cities by 1-1-1-1 Swap cannot be re-swapped by the same move.

Our TS approach has one aspiration criterion which consists of revoking tabu conditions for a move if it results in a higher objective value than the current incumbent. In this case we override the tabu condition and perform the move.

7.8 Penalty value for strategic oscillation

At each iteration of the local search procedure, several feasibility checks are performed to ensure the validity of the solution. The performed feasibility checks are explained below:

(a) **Chain feasibility**

Except for the last period, the terminal city of each day should be as same as the starting city of the next period. In other words, each tour must start from the terminal node of the previous day.

(b) **Maximum tour duration feasibility**

The total length of each tour should not exceed the maximum tour duration.

(c) **Return to campaign base feasibility**

The campaign base must be visited as a terminal node at least once every κ days.

(d) **Maximum number of meetings for different city categories**

There cannot be more than three meetings in big cities. This limitation reduces to two for regular cities. The remaining cities can host at most one meeting during the whole campaign.

(e) **No repeated meetings of the same city on the same day (No Subtour)**

Each city can host at most one meeting every day.

(f) **No meetings of Big Cities on the same day**

If there is a meeting in one of the big cities, there cannot be another meeting for the remaining big cities on the same day.

(g) **Maximum number of meetings per day**

There cannot be more than α meetings each day

In GSVNTS approach, some solutions generated during the exploration of the neighborhoods may turn out infeasible with respect to maximum tour duration constraint. We prevent infeasible solutions with respect to the all constraints above except the maximum tour duration constraint since the restoration of the other infeasibilities is a computationally expensive task. In case a solution turns out infeasible with respect to maximum tour duration, GSVNTS computes a new objective value that includes the original objective value as well a penalty associated with the infeasibility. For a given solution S the new objective function with penalties is expressed as:

$$Z'(S) = Z(S) - \mathcal{G}F(S), \quad (7.1)$$

where

$$\begin{aligned} Z(S) = & \sum_{i \in \mathbf{N}} \sum_{t \in \mathbf{T}} \pi_i \frac{\tau - t + 1}{\tau} FM_{it} + \sum_{i \in \mathbf{N}} \sum_{t \in \mathbf{T}} \sum_{1 \leq s < t} \pi_i \frac{\tau - t + 1}{\tau} \times \frac{s}{K\tau} R_{its} \\ & - \bar{K} \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} \sum_{t \in \mathbf{T}} c_{ij} X_{ijt}, \text{ and} \\ F(S) = & \sum_{t \in \mathbf{T}} \left[\sum_{i \in \mathbf{V}} \sigma_i Z_{it} + \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} d_{ij} X_{ijt} - T_{\max} \right]^+ \end{aligned} \quad (7.2)$$

$F(S)$ represents the violation in the time constraint. \mathcal{G} is a positive parameter, and function $[\cdot]^+ = \max\{0, \cdot\}$. If a given solution S is time feasible, then $Z'(S) = Z(S)$. Adjusting the values of the penalty parameter \mathcal{G} is crucial since selecting too high penalty parameter value prevents the algorithm from visiting infeasible solutions, whereas too low penalty parameter value will fall short of detecting feasible solutions on the GSVNTS steps. In this regard, the best option is to proceed with an adaptive update mechanism in which the penalty parameter is to be updated after each iteration in order to find feasible and infeasible solutions as

much equally often as possible. In our GSVNTS implementation, if the current solution S after each iteration is time-infeasible, we set $\mathcal{G} = \mathcal{G}(1 + \delta)$; if it is time-feasible, we set $\mathcal{G} = \mathcal{G} / (1 + \delta)$. A reasonable value for δ is 0.5 as suggested in Cordeau et al. (1997). The initial value of \mathcal{G} is set to 1 in the first iteration.

7.9 Skewed moves

Once the local search procedure found a local optimum, the decision of accepting or rejecting should be made. Thus, this solution must be compared to the incumbent solution. In the basic VNS, the acceptance criterion is straightforward. It accepts only improving moves. Although this strategy is simple and easy to implement, the search procedure may get stuck in a local optima easily. Therefore, we need an alternative strategy to accept non-improving moves which seems to be promising.

Instead of the basic VNS, we propose to incorporate the so-called Skewed Variable Neighborhood Search (SVNS) strategy introduced by Hansen and Mladenović (2001) where the problem of the exploring valleys far from the incumbent solution is considered. In SVNS, not only the objective value of a solution is evaluated but also its distance to the incumbent solution is also evaluated, favoring faraway solutions.

In this approach, the function $\rho(S, S'')$ calculates the distance between the incumbent solution S and the local optima S'' , where solutions in the far distance are favorable. This means that GSVNTS accepts the new non-improving solution if the following condition is satisfied:

$$Z(S) - \mu\rho(S, S'') < Z(S''), \quad (7.3)$$

where S'' is the best-found solution so far, and S is the current solution. The function $\rho(S, S'')$ denotes the distance between solutions S'' and S . This distance is expressed as the number of uncommon cities between S and S'' .

The selection of a proper value for parameter μ is crucial since the decision of accepting exploration of the distant valleys from incumbent solution depends on it. After detailed testing, we assign the value of 0.05 to parameter μ .

Other acceptance decisions are also proposed in the literature. Polacek et al. (2004) implemented SVNS in different routing problems where the threshold accepting is used instead of the basic SVNS's acceptance criteria. Vansteenwegen (2009b) used SVNS to solve different Team Orienteering Problems.

7.10 Termination criteria

Various termination criterions are used in the VNS implementations. The frequently used termination criteria in VNS methods are: (i) a fixed number of iterations, (ii) reaching a threshold CPU time, (iii) reaching a specific objective value and (iv) a fixed number of iterations without improvement. Among these criterions, the latter one is the most widely used. In our implementation, we used two termination criteria. As soon as one of the following termination criteria is met, the algorithm terminates:

- A fixed number of iterations without improvement.
- Reaching a pre-defined CPU time (3 hours).

The first criteria depends on the topology of the instance. For a small size instance, we can increase the number of iterations with non-improving solutions

since implementing rich neighborhood structures does not require high computational effort in small size problems, but in large problems spending more time on non-improving solutions is not rational. We set this number as specified in Table 7.1.

Table 7.1 Termination criteria of GSVNTS

# of days τ	Maximum # of non-improving iterations
$\tau \leq 7$	1000
$7 < \tau < 15$	400
$15 \leq \tau < 30$	200
$30 \leq \tau$	100

7.11 Granular Skewed Variable Neighborhood Tabu Search

The proposed GSVNTS algorithm gets its principles from the basic VNS; however, some features are added to the basic approach. As discussed, the basic VNS was introduced by Mladenović and Hansen (1997) with the fundamental idea of the systematic change of the neighborhoods within a possibly randomized local search algorithm to solve a wide range of combinatorial optimization problems.

Unlike many metaheuristics approaches, VNS uses multiple neighborhood structures for improving the initial solution. The systematic change of neighborhood structures is performed with the hope of finding better solutions in the other neighborhood structure of the current solution. Usually, successive neighborhoods are nested and their sequence should be such that each neighborhood covers a larger search space compared to the previous one.

Toth and Vigo (2003) introduce the granular neighborhoods for routing problems and embed it in a tabu search algorithm. The concept of granularity is added to GSVNTS algorithm to prevent non-promising moves and to reduce the computing time of neighborhood search accordingly. Granular neighborhood search is based on the use of limited neighborhoods to avoid moves that do not belong to the good feasible solution most probably. We explained the details of granularity in Section 7.5.

Another modification of the basic VNS used in GSVNTS is the concept of skewed VNS (SVNS) introduced by Hansen and Mladenović (2001). It modifies the basic approach to explore far neighborhoods of the incumbent solution. In SVNS, step 2(c) of the basic VNS has been modified. Thus, instead of moving to only better local optimums, SVNS accepts non-improving moves if the new solution is far enough to the incumbent solution. This strategy helps the algorithm to explore non-visited areas of solution space. The details of skewed moves are explained in Section 7.9.

In GSVNTS, the strategy of changing neighborhoods is in a deterministic way. That is to say, neighborhoods change in sequence. There is another variant of basic VNS where neighborhoods are changed randomly, called Randomized VNS (Repoussis et al. 2006). Unlike to Randomized VNS, we used the deterministic change of neighborhoods in our implementation.

The pseudo code of GSVNTS method is described in Algorithm 8.

Algorithm 8. GSVNTS Method

1. **Input:**
2. A set of neighborhood structures $N_k(k = 1, 2, \dots, k_{\max})$ to be used in the shaking procedure;
3. A set of neighborhood structures $N_l(l = 1, 2, \dots, l_{\max})$ to be used in the local search procedure;
4. **Parameters:** Max_Time , $MaxNonImp$, k_{\max} , l_{\max} , θ , μ ,
ShakingTimeThreshold
5. **Initialization:**
6. Find an initial feasible solution S_0 generated by FDOR–DCS with the objective value of $Z(S_0)$
7. $S^* \leftarrow S_0$
8. $Z(S^*) \leftarrow Z(S_0)$
9. $Non_Imp \leftarrow 0$
10. $ShakingTime \leftarrow 0$
11. $ShakingTimeThreshold \leftarrow \theta$
12. $l \leftarrow 1$
13. $k \leftarrow 1$
14. **While** $CPUTime() < Max_Time$ **and** $Non_Imp < MaxNonImp$ **Do**
15. ShakingSolutions = $\{(\cdot, \cdot)\}$
16. **While** $ShakingTime < ShakingTimeThreshold$ **Do** //shaking
17. ShakingSolutions \leftarrow (Shaking(S^*, k), Objective(S^*, k))
18. **End While**

```

19.    $S^* : \text{Objective}(S^*, k) \geq \text{Objective}(\bar{S}),$ 
       $\forall (\bar{S}, Z(\bar{S})) \in \text{ShakingSolutions}$ 
20.    $S' \leftarrow S^* (S' \in N_k(S))$ 
21.    $S'' \leftarrow \text{LocalSearch}(S', l)$            //Best improvement local search
22.   If  $Z(S'') > Z(S^*)$  Then                 //move or not
23.        $S^* \leftarrow S''$ 
24.        $Z(S^*) \leftarrow Z(S'')$ 
25.        $\text{Non\_Imp} \leftarrow 0;$ 
26.        $k \leftarrow 1;$ 
27.        $l \leftarrow 1$ 
28.   Else
29.        $\text{Non\_Imp} \leftarrow \text{Non\_Imp} + 1$ 
30.   If  $Z(S^*) - \mu\rho(S^*, S'') < Z(S'')$  Then //skewed moves
31.        $S^* \leftarrow S''$ 
32.        $Z(S^*) \leftarrow Z(S'')$ 
33.        $k \leftarrow 1$ 
34.        $l \leftarrow 1$ 
35.   Else
36.        $k \leftarrow k + 1$ 
37.        $l \leftarrow l + 1$ 
38.   End If
39. End If
40. End While
41. Output: A feasible solution  $S^*$  with objective value  $Z(S^*)$ 

```

Chapter 8

COMPUTATIONAL RESULTS

We test the performance of GSVNTS approach computationally. In this section, the details of the computational studies along with the results are reported.

8.1 Data sets

Since the MPTPP constraints are unique, well-known VRP and TSP instances cannot be used. Therefore, we generated 3 set of instances:

- (i) Presidential Elections I (**PE.I**): it includes 22 instances where the smallest instance includes 6 cities and 2 days and the real-world instance includes 93 cities and 40 days. The criteria for selection of cities is their base rewards.
- (ii) Presidential Elections II (**PE.II**): it includes 20 instances where the cities have been selected based on their distance with each other.
- (iii) Local Election (**LE**): it consists of three instances with 39 towns of Istanbul.

All 45 instances were generated with real-world distances and travel times among all cities and some towns of Turkey. We assume symmetric travel costs

and symmetric travel times. The naming convention of instances sheds light on the sizes of the 45 test instances and their types. An instance name ‘ **PE.I.** $nC\tau D$ ’ tells that the problem relates to presidential elections and it has n cities (excluding the fictitious city) and a planning horizon of τ days.

All instances are available at <http://shahmanzar.ir/RSP.html>. The details of instances for both dominant party and the main opposition party is represented in Appendix A.

8.2 Computational Platform and Solver specifications

Our computational tests were performed on a Dell Precision T7810 model PC equipped with one Intel Xeon[®] E5-2690 v4 2.60 GHz processor and 32 GBytes of ECC DDR3 type random access memory (RAM). Our algorithms are coded in Python 3.6. 4 (64-bit version). For the model solution and the second phase of the FDOR, among available commercial MILP solvers, we employed GUROBI 8.0.1 which is called from inside Python.

The solver specific options applied to all runs are presented in Table 8.1. The reader is referred to GUROBI User’s Manual (2018) for a more thorough explanation of these options.

Table 8.1 List of GUROBI specific options applied to all runs.

GUROBI specific options used in Python codes			
MIPGap	= 0.000	Threads	= 0
TimeLimit	= 86400	Concurrentmip	= 3
IterationLimit	= 1.e9	NumericFocus	= 3
NodeLimit	= 5.0e8	DualReductions	= 0
Nodefilestart	= 6.5	InfUnbdInfo	= 1

The relative optimality criterion (MIPGap) is set to zero as we seek proven optimality (i.e. zero gap between the best feasible and best possible solutions). MIPGap is computed as $|BFS - BPS| / |BFS| \times 100\%$ where *BFS* and *BPS* stand for the best feasible and best possible solutions, namely the tightest lower and upper bounds in a maximization problem, respectively. The CPU time limit (TimeLimit) is set to 86,400 seconds (24 hours).

The iteration limit (IterationLimit) is set to one billion. So the solution procedure will terminate after one billion iterations or 24 hours, whichever happens first. NodeLimit limits the number of nodes to be explored in the branch-and-bound tree. Nodefilestart limits the memory usage of the solver (measured in GBytes).

The options Threads and Concurrentmip turn on the multithreading (concurrent optimization) capabilities of GUROBI. When Threads is set to zero, the computing load is distributed onto all available fourteen cores (28 threads) of the processor. On the other hand, when Concurrentmip is set to three, the solver divides available threads evenly between three independent

MILP solve operations and performs them in parallel. Optimization terminates when the first solve operation completes.

In order to compare multithreading options, we tested the performance of GUROBI under different concurrent optimization configurations. We observed that `Concurrentmip=3` outperforms other configurations and finds the best feasible solution as well as the best possible solution achieving thereby the smallest optimality gap. Thus, we proceeded in our experiments with `Concurrentmip=3`. We discuss the configuration of `Concurrentmip` in more details in Section 8.2.1.

`NumericFocus` controls the degree to which the code attempts to detect and manage numerical issues. It is set to 3 since the right-hand side values of the constraint equations are relatively large in our model. `DualReductions` determines whether dual reductions are performed in the presolve step. It also helps to find out the actual status of the model solution. Finally, `InfUnbdInfo` specifies whether the LP solver will compile additional information when a model is determined to be infeasible or unbounded.

8.2.1 Concurrent MIP configuration

In order to compare the multithreading options, we analyze the performance of different concurrent optimization configurations of the MILP solver. When threads are set to zero, the computing load is allocated on all available cores of the processor. On the other hand, when `Concurrentmip` is set to 3, the solver splits all available threads evenly between 3 independent MILP solve operations and executes them in parallel. Optimization aborts when the first solve operation finishes. Table 8.2 illustrates the performance report of different

Concurrentmip configurations in Gurobi and Cplex (Concurrentmip 1, Concurrentmip 2, and Concurrentmip 3) on the same instance. We observe that when Concurrentmip is set to 3, the runtime of the Gurobi decreases significantly compared to other configurations.

Table 8.2 Comparison of Cplex with different CONCURRENTMIP configurations of Gurobi

	CPLEX	CONCURRENT MIP 1	CONCURRENT MIP 2	CONCURRENT MIP 3
Final UB	29512.39	27640.00	27724.00	27713.50
Final LB	20811.00	17867.00	20251.00	21146.50
Final Absolute Gap	8701.39	9773.00	7473.00	6567.00
Final Relative Gap (%)	0.29	0.35	0.26	0.23
BFS Finalize Time (sec)	-	31803	43224	68542

Concurrentmip 3 outperforms other configurations by finding the best feasible solution, best possible solution and the smallest gap. Therefore, we report our experimental findings with Concurrentmip 3.

8.3 Comparison of the original formulation with the alternative formulation

The comparison between alternative Maximum Tour Duration (MTD) formulation (3.41)-(3.46) and MTD constraints (3.6) is summarized in Table 8.3.

It can be seen from Table 8.3 that using the MTD constraints (3.6) improves the CPU time for all developed instances. This can be attributed to the continuous variable A that saves arrival times for all cities. In all cases,

using constraints (3.6) reduces the CPU time to solve the problem without compromising the final solution. This result led us, in this class of problems, not to pursue the alternative formulation any further.

Table 8.3 Comparison of two MTD formulation

Instance PE.I	MTD constraints (3.41)-(3.46) using A_{it}		MTD constraints (3.6)	
	Gap(%)	CPU ^a (s)	Gap(%)	CPU ^a (s)
5C2D	0.0	3.5	0.0	0.1
5C3D	0.0	3.8	0.0	0.1
7C2D	0.0	4.3	0.0	0.2
7C3D	0.0	4.7	0.0	0.4
7C4D	0.0	5.4	0.0	0.4
9C2D	0.0	15.3	0.0	0.3
9C3D	0.0	166.2	0.0	0.5
9C4D	0.0	640.7	0.0	1.3
12C3D	7.1	3600.0	0.0	5.2
12C4D	9.0	3600.0	0.0	5.8
12C5D	14.3	3600.0	0.0	6.0
15C3D	7.6	3600.0	0.0	32.1
15C4D	11.2	3600.0	0.0	214.8
15C5D	15.8	3600.0	0.0	409.5

^aIntel®Core™i5-4310U @2GHz 2.60 GHz.

8.4 Speeding up GUROBI using the results of FDOR

The commercial solvers such as GUROBI and CPLEX assume the initial values of all binary decision variables as zero. On the other hand, we know that some greedy-like and heuristic methods are used inside the black box of these solvers, especially in the pre-processing step. When we are dealing with a mixed integer linear programming problem, it is possible to assist the commercial solver to find an initial solution, by for instance including values of variables, known as a warm start. In other words, a warm start can be supplied by a feasible problem that has been previously solved. Since we have a high-quality feasible solution generated by FDOR, which executes in a small amount of CPU time, we feed the best solution of the FDOR as an initial starting solution of the original MPTPP model.

This high-quality initial solution may help the solver to start solving the problem from a better initial solution which may, though not always true, result in finding the optimal solution faster than before. Note that in large instances of MPTPP, GUROBI fails, after 24 hours, to generate a feasible solution as a lower bound of the problem. Here, the optimal values of the binary decision variables X , Z , R , FM , S , L , and E are extracted from FDOR solutions and inserted into the original MPTPP model as initial values.

In order to define these values, we only take care of those variables which their corresponding optimal value of FDOR method is equal to one. The initial values of the remaining decision variable are set to zero by default. The new results are presented in Table 8.4.

Table 8.4 The results of setting initial values of MPTPP to optimal values of FDOR

Instance	MPTPP				MPTPP with FDOR solution as the initial solution			
	LB	UB	Gap(%)	CPU(s)	LB	UB	Gap(%)	CPU(s)
PE.I								
12C5D	14575	14575	0.0	6.0	14575	14575	0.0	6.8
15C7D	17240	17240	0.0	551.3	17240	17240	0.0	572.5
15C10D	18759	18759	0.0	30458.5	18759	18759	0.0	20630.7
21C7D	19138	19138	0.0	6705.3	19138	19138	0.0	5311.0
21C10D	21904	23413	6.9	86400.0	21684	23143	6.7	86400.0
30C7D	29427	29427	0.0	20670.3	29427	29427	0.0	14071.7
30C10D	35013	37102	6.0	86400.0	35197	37226	5.7	86400.0
40C7D	30086	31317	4.1	86400.0	30122	30122	0.0	77160.8
40C10D	36409	41008	12.6	86400.0	34763	40960	17.8	86400.0
51C7D	41087	45166	9.9	86400.0	41442	44684	7.8	86400.0
51C10D	45667	55890	22.4	86400.0	46971	56169	19.5	86400.0
51C30D	47279	135554	186.7	86400.0	59885	120014	100.0	86400.0
Average			20.7	55365.9			13.1	53012.7

We observe that the solution quality is increased in almost all instances. For those instances where GUROBI was able to find the optimal solution, the CPU time is significantly decreased leading to an average improvement of 30.4%. There are, however, two instances (12C5D and 15C7D) where, the CPU time increased which could be due to restricting the search. It is also worth noting that in one instance (40C7D), GUROBI was not able to solve it to optimality, but, this variant obtained the optimal solution in less than one day. For the remaining instances, the best feasible solution is improved by

approximately 4.3% on average. Finally, the actual gap of the MPTPP model is improved by 58.7% on average.

8.5 Linear relaxation of the binary decision variables

We investigated four types of relaxations on a fairly large instance including 39 cities and 15 days. The best feasible solution (the lower bound on the true optimal solution) of the problem is reported as 21146.5 by GUROBI; However, considering the 23% relative gap, we examined whether we can find a tighter upper bound. To this end, we investigated four types of relaxations:

- i) Linear Relaxation of binary decision variables (LR),
- ii) Partial Linear Relaxation of the binary routing decision variable X (PLR1),
- iii) Semi-Full LP Relaxation with S , FM , and Z forced to be binary and all other originally binary decision variables relaxed between 0 and 1 (SFLR),
- iv) Partial Linear Relaxation of the binary decision variable S (PLR2).

The comprehensive non-relaxed model (denoted as Full Milp) has 22782 binary decision variables after the reductions performed by Gurobi at the root node of the branch and bound tree before the iterations commence. By relaxing the binary decision variable X , this number reduced to 2885 for the Partial Linear Relaxation (PLR1) version of the full model.

Table 8.5 presents the test results obtained from the five models discussed above.

Table 8.5 Comparison of different solutions of the same instance with 39 cities and 15 days

Model	MIP Solution	LP Solution	Best possible	Absolute gap	Relative gap (%)	CPU^a (s)
Full MILP	21146.5	-	27713.5	6567.0	0.23	86435.54
LR	-	65585.8	-	-	-	38.06
PLR1	26125.3	-	29143.9	3018.6	0.10	3791.84
SFLR	25345.3	-	29596.6	4251.4	0.14	43216.80
PLR2	57431.7	-	58744.2	1312.5	0.02	35537.92

^aIntel®Core™i5-4310U @2GHz 2.60 GHz.

As we observe in Table 8.5, the final upper bound for PLR1 is 291439.9 which is worse than the final upper bound of the MILP model (which is 27713.5). The lower bound of SFLR at the end of 11 hours is as high as 57431.7 and the UB is 58744.17. These are also extremely loose bounds. Therefore, the FULL MILP upper bound is found to be the tightest bound we could obtain so far.

8.6 Effect of the added valid inequalities (VI) on solution quality

Table 8.6 displays the lower and upper bounds found by GUROBI 8.0.1 and the corresponding CPU time (in seconds) for the models without VIs, with partial VI (with VI except for constraints (3.50)) and with all VI. The optimal solutions are shown in bold where the remaining figures display the best solution.

Table 8.6 Comparison of the models with and without valid inequalities

Instance	<i>All VIs OFF</i>			<i>All VIs ON except</i>			<i>All VIs ON</i>		
	Obj. Val.	Gap (%)	CPU	Obj. Val.	Gap (%)	CPU	Obj. Val.	Gap (%)	CPU
PE.I									
12C3D	12620	0.0	1.4	12620	0.0	0.9	12620	0.0	1.1
12C4D	16584	0.0	5.5	16584	0.0	2.7	16584	0.0	2.3
12C5D	14575	0.0	38.5	14575	0.0	7.3	14575	0.0	6.0
15C3D	12620	0.0	2.4	12620	0.0	1.7	12620	0.0	1.6
15C4D	14210	0.0	10.4	14210	0.0	5.9	14210	0.0	4.3
15C5D	15446	0.0	113.1	15446	0.0	37.3	15446	0.0	14.4
15C7D	17240	0.0	2477.1	17240	0.0	1528.0	17240	0.0	551.3
15C10D	18719	5.5	86400.0	18759	0.0	35355.0	18759	0.0	30458.5
21C7D	19138	0.0	17290.9	19138	0.0	5296.1	19138	0.0	6705.3
21C10D	21727	11.2	86400.0	21792	7.4	86400.0	21904	6.9	86400.0
30C7D	29427	0.0	19736.3	29427	0.0	7421.5	29427	0.0	20670.3
30C10D	32803	18.8	86400.0	33281	14.0	86400.0	35013	6.0	86400.0
Average	18759	2.9		18807	1.7		18961	1.0	

According to Table 8.6, the formulation of MPTPP is more compact when all valid inequalities are used. Based on these positive results, we opted to include all valid inequalities in our experiments. This led to an improvement in the average objective value and a reduction in the average gap. The deviation reduces from nearly 3% to just a 1% on average and the largest % deviation is just below 7%, a massive drop from the previous value of 11.2%.

8.7 Changing values of α

In all computational experiments, the default value of α is set to 4. We compare the performance of the developed mathematical formulation in terms of the different values of α . The results are listed below in Table 8.7. For each instance set, we indicate the number of nodes and number of days. For each value of α we list the CPU time in seconds as well as optimality gap.

Table 8.7 Results of the instances for different values of α

Instance	$\alpha = 1$		$\alpha = 2$		$\alpha = 3$	
	Gap(%)	CPU	Gap(%)	CPU	Gap(%)	CPU
5C2D	0.0	3.5	0.0	3.3	0.0	3.4
5C3D	0.0	3.4	0.0	3.5	0.0	3.5
7C2D	0.0	3.5	0.0	3.7	0.0	3.7
7C3D	0.0	3.7	0.0	3.8	0.0	3.8
7C4D	0.0	4.2	0.0	4.5	0.0	4.2
9C2D	0.0	3.7	0.0	3.7	0.0	3.6
9C3D	0.0	4.9	0.0	4.7	0.0	4.8
9C4D	0.0	9.4	0.0	9.4	0.0	9.5
12C3D	0.0	4.5	0.0	4.3	0.0	4.4
12C4D	0.0	14.8	0.0	15.4	0.0	16.0
12C5D	0.0	35.2	0.0	36.1	0.0	36.9
15C3D	0.0	6.1	0.0	6.2	0.0	6.2
15C4D	0.0	29.8	0.0	30.7	0.0	31.0
15C5D	0.0	34.1	0.0	35.3	0.0	36.5
21C5D	11.0	600.0	11.1	600.0	11.0	600.0
21C7D	11.1	600.0	13.7	600.0	13.8	600.0
21C10D	15.2	600.0	17.2	600.0	18.5	600.0

The results in Table 8.7 indicate that increasing the maximum number of meetings allowed to be held each day does not significantly affect the solution time. The obtained best gaps are also very close in large-sized instances.

8.8 Computational results of scenario analysis level 1

The results of the four different scenarios described above are presented in Table 8.8. The naming convention in the leftmost column of the table sheds light on the sizes of the 10 test instances. Boldface figures in the first column of each scenario in Table 8.8 point to proven optimality achieved by the commercial solver GUROBI.

Table 8.8 consists of four segments where each segment corresponds to a scenario and reports the optimal or best feasible objective value (BFS), the number of meetings held during the planning horizon (m) and the final gap reported by GUROBI. The CPU time limit in all runs was applied as 24 hours. In the first scenario *Full-MILP*, we also report the CPU time that elapsed until the lower bound on the value of the maximization objective, namely the BFS reached its final level. The average gap and CPU time are 6.19% and 29504.3 seconds, respectively. This implies MPTPP is a large-scale optimization problem even for small size instances.

In *Full-1Meet* (Scenario 2), the computational complexity of the problem is greatly reduced due to the removal of the binary variables FM_{it} and R_{ius} from the model and due to the simplified net benefit definition shown in (5.1). This simplification helps the solver find better solutions within the CPU time limit of 24 hours compared to *Full-MILP* (Scenario 1). The average gap decreases to 7.20%.

Objective values obtained in *Rew-Only* (Scenario 3) are not comparable with the ones obtained in the other scenarios since the traveling costs in the definition of net benefit are ignored. However, except in three instances, namely 40C-7D, 51C-7D and 51C-10D, the count of meetings realized in this scenario is either higher than or equal to the others. This can be ascribed to having a larger feasible solution space which occurs because of lifting the necessity to visit the campaign base every κ days.

Similarly, the politician in this scenario has more freedom to travel to remote cities that he would not visit in the base scenario due to the net benefit being negative after the deduction of traveling expenses. The average gap in Scenario 3 is 6.07%.

The results of *Alt-1Depot* (Scenario 4) are also interesting, as this scenario bears the most similar conditions to the current campaign policy of the PP. In comparison to *Full-MILP*, the commercial solver GUROBI in this scenario was able to attain optimality in one more instance (51C-7D). Table 8.8 reports an overall lesser number of meetings in Scenario 4.

It is apparent that the requirement to return to the capital city Ankara at the end of every day prevents some of the meetings which were realized in the base scenario *Full-MILP*. The average gap reported by GUROBI in this scenario is 7.78%.

Table 8.8 Results of the four scenarios.

Instance PE.I	Scenario 1				Scenario 2			Scenario 3			Scenario 4		
	<i>Full- MILP</i>	<i>m</i>	Gap (%)	t_{BFS}^a	<i>Full- 1Meet</i>	<i>m</i>	Gap (%)	<i>Rew- Only</i>	<i>m</i>	Gap (%)	<i>Alt- 1Depot</i>	<i>m</i>	Gap (%)
15C7D	17240	14	0.0	451	16000	13	0.0	22561	16	0.0	11539	10	0.0
15C10D	18759	17	0.0	10556	16299	15	0.0	26061	20	0.0	11170	13	15.4
21C7D	19138	16	0.0	2521	18117	16	0.0	23932	17	0.0	13779	12	0.0
21C10D	21904	21	6.9	24431	20850	20	0.0	28927	23	3.8	14498	15	14.6
30C7D	29427	18	0.0	9785	27576	17	3.3	33638	18	0.0	20774	13	0.0
30C10D	35013	24	6.0	61308	32210	24	9.0	38148	24	10.0	24520	17	15.9
40C7D	30086	20	4.1	11280	27023	18	16.4	32586	19	9.5	20893	13	7.1
40C10D	36409	25	12.6	44758	32210	24	9.0	39876	25	13.7	25324	18	19.7
51C7D	41087	31	9.9	71322	33366	23	17.6	33123	19	11.0	31154	19	0.0
51C10D	45667	36	22.4	58631	35314	26	16.7	41325	28	12.7	37373	26	5.1

^a CPU time in seconds spent until the reporting of the best feasible solution by GUROBI.

* The value of the best feasible solution (BFS) is provided where GUROBI is not able to prove optimality in 24 hours.

8.9 Computational results of scenario analysis level 2

The importance of holding meetings in the early days or in the last days of the campaign period should be decided by party executives indeed. Despite this fact, we present in Table 8.9 the comparison between the original and the alternative reward functions on 14 small size test instances using the base scenario *Full-MILP*. All instances are solved to proven optimality under each net benefit function. The column with the header ‘CPU (s)’ indicates the solution times in seconds reported by GUROBI.

Table 8.9 Comparison of new and original reward function.

Instances	<i>Full-MILP</i> with original reward function (3.1)			<i>Full-MILP</i> with alternative reward function (5.2)			
	PE.I	Obj. Value	Gap (%)	CPU (s)	Obj. Value	Gap (%)	CPU (s)
6C2D		7110	0.0	0.1	17441	0.0	0.1
6C3D		8181	0.0	0.1	22272	0.0	0.2
7C2D		9629	0.0	0.1	22172	0.0	0.2
7C3D		10939	0.0	0.2	26778	0.0	0.4
7C4D		11597	0.0	0.4	30339	0.0	0.7
9C2D		9695	0.0	0.3	22172	0.0	0.1
9C3D		10939	0.0	0.5	28572	0.0	0.9
9C4D		11668	0.0	1.3	32149	0.0	1.8
12C3D		12620	0.0	1.1	31726	0.0	1.3
12C4D		13584	0.0	2.3	37076	0.0	3.6
12C5D		14575	0.0	6.0	40382	0.0	60.4
15C3D		12620	0.0	1.6	32750	0.0	2.9
15C4D		14210	0.0	4.3	39496	0.0	6.1
15C5D		15446	0.0	14.4	43533	0.0	159.8

According to Table 8.9 the solution times obtained with the original reward function are better in 13 out of 14 instances. The objective values are unfortunately not comparable due to different rewards being assigned to each city in different days. The decision which reward function to adopt is to be made by the politician. Based on the results shown in Table 8.9, it can be comprehended that using the alternative reward function in (5.2) increases the computational complexity of the problem.

8.10 Tightening scheme

As shown in previous sections, the average gap of Full MILP model was too high. For larger instances like 51C 30D, the commercial solver was not able even to generate a feasible solution after 24 hours. Therefore, we need to produce the upper bound and lower bound for the problem using alternative methods. Since the objective function of MPTPP is maximization, the lower bound is produced by any feasible solution and the upper bound is produced by any type of relaxation of the original problem.

We use the procedure described in Figure 8.1 to tighten the actual optimal value of the main problem (51 city and 30 days). To obtain a valid upper bound, we tested different relaxation schemes (full linear relaxation of all binary variables and partial linear relaxation of some variables) as discussed in Section 8.5. The partially linear relaxation (PLR) scenario where the binary decision variable X is relaxed turns out to be the best upper bound.

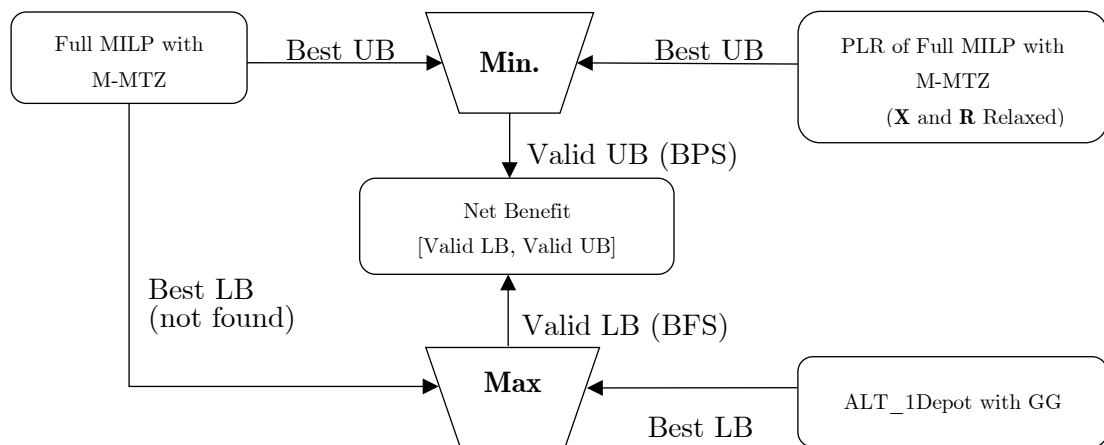


Figure 8.1 Procedure for tightening the optimal value of main problem

The actual upper bound of the problem is produced by selecting the minimum of the best upper bounds achieved by the partial linear relaxation of the Full MILP and the main problem. The valid lower bound is produced by choosing the maximum of the best lower bound of alternative one depot model and main problem. The gap is reduced to 0.68% after performing this scheme for 24 hours.

8.11 Omitting binary decision variables and parameters

In order to analyze the effect of variables FM and R on the performance of the model solution, we omit these variables from the original formulation of the problem. Then, we employ the binary decision variable Z in the objective function. Therefore, the objective function (which maximizes the time-dependent reward as a function of Z) is changed as follows:

$$\begin{aligned} \max. \text{ NET BENEFIT} = \\ \sum_{i \in \mathbf{N}} \sum_{t \in \mathbf{T}} \pi_i \frac{\tau - t + 1}{\tau} Z_{it} - \bar{K} \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} \sum_{t \in \mathbf{T}} c_{ij} X_{ijt} \end{aligned} \quad (8.1)$$

Obviously, such a formulation leads to frequent visits to highly-rewarded cities. Thus, we add the following constraints to the model accordingly:

$$\sum_{t \in \mathbf{T}} Z_{it} \leq 2 \quad i \in \mathbf{N} \quad (8.2)$$

The set of constraints (8.2) impose a limit on the number of the meetings for any city during the total campaign period. Therefore, the party leader is

forced to proceed by visiting other cities as well. Another variation is to make all costs between each pair of cities equal to zero. Since we do not assume budget constraints in our model, this assumption will definitely improve the optimal value of the problem. The comparison results are presented in Table 8.10.

Table 8.10 Comparison of the original reward function with the reward function without binary variable FM and Z

Instance PE.I	<i>Full Model with FM</i>			<i>Full Model without FM</i>		
	<i>and R</i>			<i>and R</i>		
	Obj.	Gap	CPU	Obj.	Gap	CPU
6C2D	7008	0.0	3.5	7008	0.0	3.5
6C3D	7846	0.0	3.9	9038	0.0	3.5
7C2D	7678	0.0	3.6	7678	0.0	3.7
7C3D	8947	0.0	4.4	9942	0.0	3.8
7C4D	9630	0.0	4.9	11898	0.0	4.9
9C2D	7678	0.0	3.8	7678	0.0	3.7
9C3D	8947	0.0	5.2	9942	0.0	4.7
9C4D	10297	0.0	10.2	11912	0.0	8.8
12C3D	9755	0.0	15.8	9942	0.0	8.7
12C4D	11437	0.0	39.5	12155	0.0	79.5
12C5D	12541	0.0	233.0	14165	0.0	563.2
15C3D	9897	0.0	32.1	10242	0.0	36.5
15C4D	11855	0.0	214.8	12578	0.0	491.5
15C5D	12738	2.1	600.0	15653	2.1	600.0

As expected, from Table 8.10 we find that as we omit binary variables FM and R from the original formulation of the problem, the corresponding objective value improves. This increase in the objective value is due to not penalizing repetitive meetings in highly-rewarded cities.

8.12 Performance of FDOR–DCS

Table 8.11 illustrates the results of the FDOR–DCS on PE.I instances. The column with the header ‘CPU (s)’ indicates the solution times in seconds. In all computational results, the boldface figures point to proven optimality achieved by either commercial solver GUROBI or developed algorithms.

The results of Table 8.11 indicate that FDOR–DCS is able to generate very efficient solutions in a significantly small amount of CPU time compared to the commercial solver GUROBI.

The average gap between the optimal solution (or the best known feasible solution) of the MILP and the net benefit of the FDOR–DCS is 5.03% where the average CPU time decreased considerably from 45854.27 seconds to 6.55 seconds. In other words, FDOR–DCS requires a tiny fraction, approximately 0.014% of the original algorithm.

Table 8.11 Computational Results of FDOR–DCS ($\lambda = n$)

Instance	MPTPP	Gap(%)	CPU (s)	FDOR–DCS	Gap(%)	CPU (s)
PE.I						
6C2D	7110	0.0	0.1	7110	0.0	0.08
6C3D	8181	0.0	0.1	8181	0.0	0.14
7C2D	9629	0.0	0.2	9629	0.0	0.08
7C4D	11597	0.0	0.4	11457	1.2	0.20
9C3D	10939	0.0	0.5	10788	1.4	0.13
9C4D	11668	0.0	1.3	11268	3.4	0.15
12C5D	14575	0.0	6.0	12906	11.5	0.30
15C7D	17240	0.0	551.3	16132	6.4	0.52
15C10D	18759	0.0	30458.5	17356	7.5	0.70
21C7D	19138	0.0	6705.3	17325	9.5	0.99
21C10D	21904	6.8	86400.0	20673	5.6	1.22
30C7D	29427	0.0	20670.3	27474	6.6	1.72
30C10D	35013	5.9	86400.0	32213	8.0	2.26
40C7D	30086	4.0	86400.0	28821	4.2	3.74
40C10D	36409	12.6	86400.0	34672	4.8	4.97
51C7D	41087	9.9	86400.0	36942	10.1	8.40
51C10D	45667	22.3	86400.0	43212	5.4	11.38
51C30D	47279	186.7	86400.0	59890	-26.7	14.56
70C15D	–	–	86400.0	46818	–	16.64
70C40D	–	–	86400.0	58408	–	22.26
93C30D	–	–	86400.0	68174	–	26.61
93C40D	–	–	86400.0	73574	–	27.12

According to Table 8.11, for those instances where the commercial solver was able to find the optimal solution, the FDOR–DCS finds significantly quick solutions with the average gap of 4.31%. For the remaining instances, where the commercial solver was not able to find the optimal solution after 24 hours, the average gap of FDOR–DCS solutions with the best feasible solution is 6.35%. In four instances, namely 70C15D, 70C40D, 93C30D, and 93C40D GUROBI was not able to generate a feasible solution even after 24 hours of CPU time whereas FDOR–DCS finds an effective solution in 23.15 seconds on average.

The high-quality feasible solution generated by FDOR–DCS is exploited next to tighten the actual gap of the original problem. It can also be used as an initial feasible solution of the original problem. Another observation from Table 8.11 is that by increasing the number of cities and the number of days, the CPU time of FDOR–DCS increases exponentially, which may take a significantly long time to solve large-scale instances. Therefore, two other approaches FDOR–GCS and FDOR–PCS are investigated here.

8.13 Performance of FDOR–GCS

Our approach to the FDOR–DCS exploits the ability of FDOR to solve daily prize-collecting TSPs in a small amount of CPU time. Such a performance may not be reachable in the large instances. On the other hand, providing all cities as the candidate cities for phase 2 is not guaranteed to produce better solutions compared with choosing a subset of cities. The reason lies in the fact that visiting some low reward cities as the terminal node may result in exploring a better solution space on the next day which was not possible to reach in the current day due to the maximum tour duration constraint.

In FDOR–GCS, all cities are sorted in the decreasing order of their updated reward at the beginning of each day. Next, the top λ cities with the highest rewards are selected as the candidate cities for phase 2. This strategy of city selection yields obtaining high-quality solutions by guaranteeing that high rewards cities are always included within the candidate cities.

Table 8.12 depicts the solutions for the medium-sized and large-sized instances with FDOR–GCS possessing different values of λ . In GUROBI results, boldface figures point to proven optimality achieved by the commercial solver GUROBI and in FDOR–GCS results, boldface figures point to the best solution achieved for every instance.

As it is shown in Table 8.12, for every λ and each instance, the best-achieved solution and corresponding CPU time are reported. By increasing the parameter, in most cases, the objective value of the solutions are improved while the increase in CPU time is not significant. After $\lambda = 18$, there are some improvements in the objective values of some instances but the CPU time increases significantly. This increase is much more apparent in large instances. For instance, taking 93C40D instance into account, by $\lambda = 21$ the CPU time increases by almost 30% compared to the case $\lambda = 18$. Therefore, considering the best-obtained gaps and the runtimes of the algorithm, in most instances the best solution and CPU time obtained with $\lambda = 18$.

The average gap in FDOR–GCS is 4.97% for those PE.I instances where there is a (best)known solution. The rightmost column in Table 8.12 indicates the CPU time of the best-obtained solution in seconds. The average gaps and average CPU times are reported at the last row where the smallest gap is obtained by $\lambda = 18$.

8.14 Performance of FDOR-PCS

The results in Table 8.13 provide the solutions of FDOR-PCS with different Max_Iter values. As the parameter Max_Iter increases, the CPU time is multiplied by its value. The best-obtained gaps are achieved by $Max_Iter = 20$ where the average CPU time increases up to 47.27 seconds. The boldface figures under FDOR-PCS point to the best objective value for every instance.



Table 8.12 Computational Results of FDOR–GCS

GUROBI			FDOR–GCS										Best Obj.	Best Gap (%)	BestCPU (s)
Instance PE.I	Best	CPU	$\lambda = 10$		$\lambda = 13$		$\lambda = 15$		$\lambda = 18$		$\lambda = 21$				
	Obj.	(s)	Best Obj.	CPU (s)	Best Obj.	CPU (s)	Best Obj.	CPU (s)	Best Obj.	CPU (s)	Best Obj.	CPU (s)			
12C5D	14575	6.0	12906	0.39	–	–	–	–	–	–	–	–	12906	11.45	0.39
15C7D	17240	551.3	16132	0.44	16132	0.48	16132	0.57	–	–	–	–	16132	6.43	0.44
15C10D	18759	30458.5	17356	0.42	17356	0.45	17356	0.52	–	–	–	–	17356	7.48	0.42
21C7D	19138	6705.3	17324	0.38	17324	0.49	17324	0.59	17324	0.66	17324	0.85	17324	9.48	0.38
21C10D	21904	86400.0	20673	0.49	20673	0.69	20673	0.71	20673	0.85	20673	1.04	20673	5.62	0.49
30C7D	29427	20670.3	27963	0.38	27963	0.51	27963	0.58	27963	0.75	27474	1.01	27963	4.98	0.38
30C10D	35013	86400.0	32427	0.49	32427	0.65	32427	0.81	32427	1.03	32427	1.38	32427	7.39	0.49
40C7D	30086	86400.0	28071	0.39	28074	0.59	28074	0.82	28114	1.08	28114	1.46	28114	6.55	1.08
40C10D	36409	86400.0	31656	0.59	31890	0.78	32654	1.20	34278	1.49	34278	2.05	34278	5.85	1.49
51C7D	41087	86400.0	34429	0.46	35119	0.90	36446	0.95	36446	1.33	36446	1.98	36446	11.30	0.95
51C10D	45667	86400.0	40391	0.60	42119	1.04	41538	1.23	42406	1.89	42273	2.55	42406	7.14	1.89
51C30D	47279	86400.0	53623	1.68	54336	2.42	56232	2.18	58587	3.49	56348	4.28	58587	-23.92	3.49
70C15D	–	86400.0	41946	0.70	40974	1.16	43556	1.54	44911	1.90	45752	2.36	45752	–	2.36
70C40D	–	86400.0	46723	2.13	49942	3.62	52979	2.95	53747	6.31	54235	7.60	54235	–	7.60
93C30D	–	86400.0	58833	2.04	60476	3.17	62926	3.92	63247	5.93	63085	8.03	63085	–	8.03
93C40D	–	86400.0	61390	2.34	62016	3.96	64819	4.72	65746	6.87	68307	9.94	68307	–	9.94
			Avg. Gap (%)	Avg. CPU (s)	Avg. Gap (%)	Avg. CPU (s)	Avg. Gap (%)	Avg. CPU (s)	Avg. Gap (%)	Avg. CPU (s)	Avg. Gap (%)	Avg. CPU (s)			
			7.24	0.87	6.17	1.39	5.43	1.55	3.82	2.58	4.56	3.42			

Table 8.13 Computational Results of FDOR-PCS

GUROBI		FDOR-PCS															
Instance PE.I	Best Obj.	CPU (s)	<i>Max_Iter=5</i>			<i>Max_Iter=10</i>			<i>Max_Iter=15</i>			<i>Max_Iter=20</i>			Best Obj.	Best Gap (%)	Best CPU (s)
			Best Obj.	CPU (s)	Avg Obj.	Best Obj.	CPU (s)	Avg Obj.	Best Obj.	CPU (s)	Avg Obj.	Best Obj.	CPU (s)	Avg Obj.			
12C5D	14575	6.0	12906	1.47	12906	12906	2.94	12906	12906	4.32	12906	12906	5.76	12906	12906	11.45	1.47
15C7D	17240	551.3	16103	1.72	15913	16103	3.41	15909	16103	7.44	16103	16103	9.91	16103	16103	6.60	1.72
15C10D	18759	30458.5	17234	2.34	17154	17182	6.72	17182	17182	9.96	17182	17234	9.30	16975	17234	8.13	2.34
21C7D	19138	6705.3	17324	3.86	17293	17324	7.49	17324	17324	11.29	17303	17324	14.64	17322	17324	9.48	3.86
21C10D	21904	86400.0	20673	5.07	20673	20673	10.05	20673	20673	15.18	20530	20673	20.34	20549	20673	5.62	5.07
30C7D	29427	20670.3	27046	4.32	26036	27474	8.10	26205	27963	12.24	26420	27533	16.8	26267	27963	4.98	12.24
30C10D	35013	86400.0	31321	5.87	30258	32427	11.91	30499	32368	17.26	31181	32533	23.51	31085	32533	7.08	23.51
40C7D	30086	86400.0	26313	5.10	24728	27308	8.72	26050	27927	14.18	25852	27627	17.49	26102	27927	7.18	14.18
40C10D	36409	86400.0	31047	7.92	29576	33233	14.67	31550	33107	20.80	31056	32766	30.59	31013	33233	8.72	14.67
51C7D	41087	86400.0	33446	5.85	31356	34635	11.67	32283	36077	17.29	32218	36218	22.47	33241	36218	11.85	22.47
51C10D	45667	86400.0	40670	7.49	39630	42165	15.59	39025	40896	23.34	39129	41362	30.73	39029	42165	7.67	15.59
51C30D	47279	86400.0	58633	18.53	57848	59081	37.25	56921	58904	59.24	55811	59745	105.71	57157	59745	-26.37	105.71
70C15D	–	86400.0	40750	9.85	38061	41988	18.48	38964	40848	41.09	39163	43116	54.85	39726	43116	–	54.85
70C40D	–	86400.0	51854	36.54	50022	53392	71.68	50122	54809	103.38	50831	54589	142.51	50242	54809	–	103.38
93C30D	–	86400.0	56972	27.13	53991	58458	53.72	54494	58451	82.95	54070	58493	108.73	55158	58493	–	108.73
93C40D	–	86400.0	61867	34.69	59258	62090	69.21	59681	64118	106.12	58701	61958	143.07	59145	62090	–	69.21
			Avg. Gap (%)	Avg. CPU (s)		Avg. Gap (%)	Avg. CPU (s)		Avg. Gap (%)	Avg. CPU (s)		Avg. Gap (%)	Avg. CPU (s)				
			7.73	11.10		5.99	21.97		5.70	34.13		5.66	47.27				

8.15 Comparison of performances of FDOR–DCS, FDOR–GCS, and FDOR–PCS

Table 8.14 presents the comparison of different city selection approaches in FDOR for PE.I instances. For the FDOR–GCS and the FDOR–PCS, the objective value of the best-achieved solution and its corresponding CPU time is provided. The FDOR–DCS outperforms other city selection approaches in most instances. In two instances, namely 30C7D and 30C10D, FDOR–GCS finds better solutions with higher objective values. In three other instances, 15C7D, 15C10D and 21C10D, FDOR–GCS finds the same solution as FDOR–DCS with smaller CPU time.

In general, FDOR–DCS spends more time on the second phase by considering all available cities. FDOR–GCS and FDOR–PCS consider a subset of cities, which are selected in a greedy way or pseudo-random way, to ensure solution diversity. FDOR–DCS with an average gap of 4.40% for the main set of instances, including 12 test instances with best known objective values, outperforms FDOR–GCS and FDOR–PCS with average gaps of 4.97% and 5.19%, respectively. These three selection rules were tested thoroughly using various values of λ for FDOR–GCS and several maximum iterations. The results show that FDOR–DCS is still outperforming others.

Given the good performance of FDOR–DCS, in the rest of our experiments, we just focus on this particular variant. Figure 8.2 recapitulates the information in Table 8.14 to depict the differences between the commercial solver’s results and FDOR with different selection rules.

Table 8.14. Comparison of FDOR–DCS, FDOR–GCS, and FDOR–PCS

Instance	GUROBI			FDOR–DCS		FDOR–GCS		FDOR–PCS		
	PE.I	Best	Gap (%)	CPU (s)	Obj. Val.	CPU (s)	Obj. Val.	CPU (s)	Obj. Val.	CPU (s)
12C5D		14575	0.0	6.0	12906	0.30	12906	0.39	12906	1.47
15C7D		17240	0.0	551.3	16132	0.52	16132	0.44	16103	1.72
15C10D		18759	0.0	30458.5	17356	0.70	17356	0.42	17234	2.34
21C7D		19138	0.0	6705.3	17325	0.99	17324	0.38	17324	3.86
21C10D		21904	6.8	86400.0	20673	1.22	20673	0.49	20673	5.07
30C7D		29427	0.0	20670.3	27474	1.72	27963	0.38	27963	12.24
30C10D		35013	5.9	86400.0	32213	2.26	32427	0.49	32533	23.51
40C7D		30086	4.0	86400.0	28821	3.74	28114	1.08	27927	14.18
40C10D		36409	12.6	86400.0	34672	4.97	34278	1.49	33233	14.67
51C7D		41087	9.9	86400.0	36942	8.40	36446	0.95	36218	22.47
51C10D		45667	22.3	86400.0	43212	11.38	42406	1.89	42165	15.59
51C30D		47279	186.7	86400.0	59890	14.56	58587	3.49	59745	105.71
70C15D		–	–	86400.0	46818	16.64	45752	2.36	43116	54.85
70C40D		–	–	86400.0	58408	22.26	54235	7.60	54809	103.38
93C30D		–	–	86400.0	68174	26.61	63085	8.03	58493	108.73
93C40D		–	–	86400.0	73574	27.12	68307	9.94	62090	69.21
Average					4.40%	8.96	4.97%	2.48	5.19%	34.93

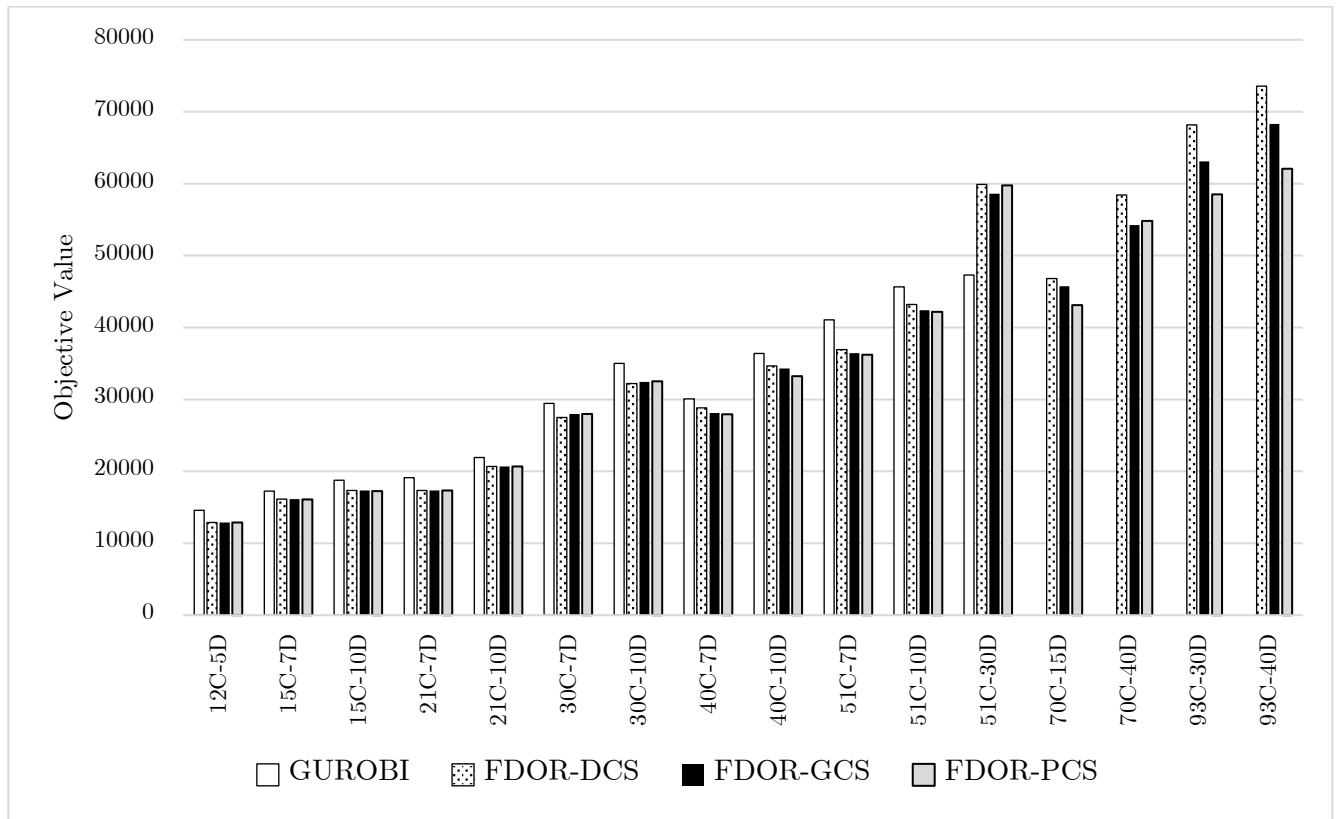


Figure 8.2 Comparison of FDOR-DCS, FDOR-GCS, and FDOR-PCS

Table 8.15 presents the comparison of routes generated by FDOR-DCS algorithm and the optimal solution for the instance 30C7D. We specify “Holding a Meeting” by (M).

Table 8.15 Comparison of routes of FDOR–DCS and optimal solution

Routes	
Optimal Solution	Day 1: Wakeup in Ankara (M) → Hatay (M) → İskenderun(M) (Sleep in İskenderun)
	Day 2: Wakeup in İskenderun → Adana (M) → Istanbul (M) → Antalya (Sleep in Antalya)
	Day 3: Wakeup in Antalya (M) → Denizli (M) → Aydın (M) → Izmir (Sleep in Izmir)
	Day 4: Wakeup in Izmir (M) → Balıkesir (M) → Bursa (M) (Sleep in Bursa)
	Day 5: Wakeup in Bursa → Istanbul (M) → Gebze (M) → Ankara (Sleep in Ankara)
	Day 6: Wakeup in Ankara (M) → Gaziantep (M) → Kahramanmaraş (M) (Sleep in Kahramanmaraş)
	Day 7: Wakeup in Kahramanmaraş → Hatay (M) → Adana (M) (Sleep in Adana)
FDOR–DCS	Day 1: Wakeup in Ankara (M) → Hatay (M) → İskenderun(M) (Sleep in İskenderun)
	Day 2: Wakeup in İskenderun → Istanbul (M) → Bursa (M) (Sleep in Bursa)
	Day 3: Wakeup in Bursa → Izmir (M) → Aydın (M) (Sleep in Aydın)
	Day 4: Wakeup in Aydın → Denizli (M) → Antalya (M) → Alanya (M) (Sleep in Alanya)
	Day 5: Wakeup in Alanya → Isparta (M) → Ankara (M) (Sleep in Ankara)
	Day 6: Wakeup in Ankara → Gaziantep (M) → Kahramanmaraş (M) (Sleep in Kahramanmaraş)
	Day 7: Wakeup in Kahramanmaraş → Adana (M) → Istanbul (M) (Sleep in Istanbul)

In order to illustrate the efficiency of the FDOR–DCS, we compared the daily net benefit (collected daily rewards minus daily travel costs) of the MPTPP optimal solution with FDOR–DCS solution for 30C7D instance in Figure 8.3. In day 1, FDOR–DCS generates the same route as the optimal solution and in day 2 and day 7, it was able to obtain higher net benefits for the same days in the optimal solution.

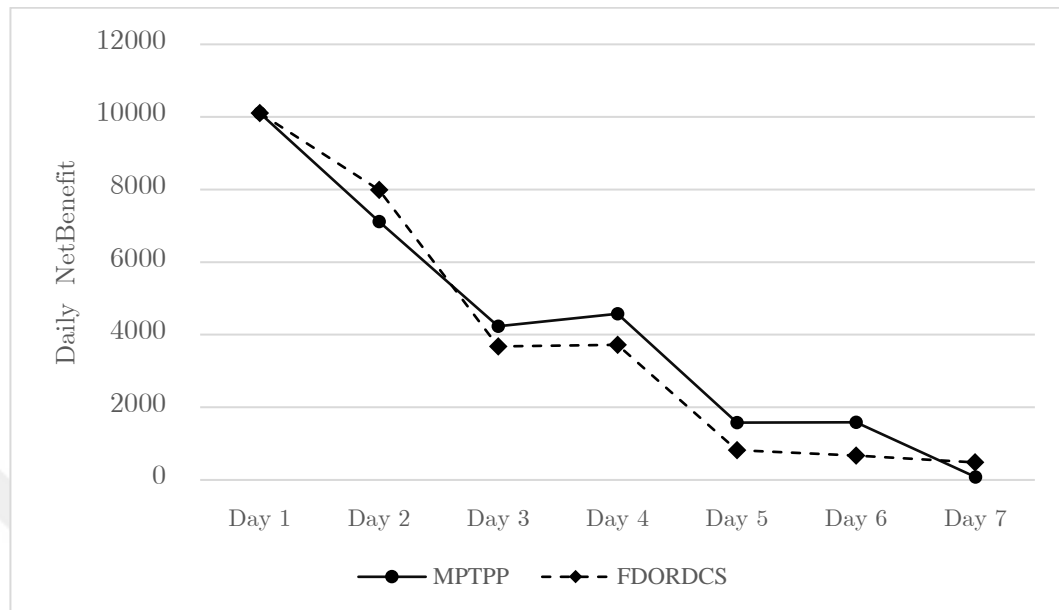


Figure 8.3 The comparison of the net benefit of FDOR and MPTPP

8.16 Comparison of GUROBI and FDOR-DCS for all three set of instances

Since it was found that FDOR-DCS finds better solutions compared to FDOR-GCS and FDOR-PCS in PE.I instances, for completeness, we compared the results of the commercial solver and FDOR-DCS for all 45 instances. Table 8.16 displays the objective value of all instances for both GUROBI and FDOR-DCS along with CPU times in seconds.

Table 8.16 Comparison of GUROBI with FDOR–DCS for all instances

Instance	GUROBI			FDOR–DCS			Instance	GUROBI			FDOR–DCS		
	Best	CPU	Gap(%)	Obj.Val	CPU	Gap (%)		Best	CPU	Gap(%)	Obj.Val	CPU	Gap (%)
PE.I							PE.II						
6C2D	7110	0.1	0.0	7110	0.1	0.0	20C5D	25118	239.2	0.0	24196	0.6	3.7
6C3D	8181	0.1	0.0	8181	0.1	0.0	20C7D	27523	1995.9	0.0	25419	0.6	7.6
7C2D	9629	0.2	0.0	9629	0.1	0.0	30C5D	16635	709.9	0.0	16052	1.5	3.5
7C4D	11597	0.4	0.0	11457	0.2	1.2	30C7D	18855	28216.8	0.0	17997	1.8	4.6
9C3D	10939	0.5	0.0	10788	0.1	1.4	30C10D	21251	86400.0	5.9	19577	2.0	7.9
9C4D	11668	1.3	0.0	11268	0.1	3.4	40C7D	32811	86400.0	20.1	31748	3.0	3.2
12C5D	14575	6.0	0.0	12906	0.3	11.5	40C10D	37851	86400.0	3.8	34267	3.6	9.5
15C7D	17240	551.3	0.0	16132	0.5	6.4	50C7D	32829	86400.0	1.6	33101	6.9	-0.8
15C10D	18759	30458.5	0.0	17356	0.7	7.5	50C10D	38098	86400.0	11.8	37389	8.5	1.9
21C7D	19138	6705.3	0.0	17325	0.9	9.5	50C15D	44098	86400.0	35.6	41687	11.0	5.5
21C10D	21904	86400.0	6.8	20673	1.2	5.6	60C7D	40480	86400.0	2.5	38105	13.8	5.9
30C7D	29427	20670.3	0.0	27474	1.7	6.6	60C10D	48270	86400.0	7.0	45446	18.8	5.8
30C10D	35013	86400.0	5.9	32213	2.2	8.0	60C20D	50559	86400.0	80.1	62869	22.8	-24.3
40C7D	30086	86400.0	4.0	28821	3.7	4.2	70C10D	42474	86400.0	13.9	40201	26.1	5.3
40C10D	36409	86400.0	12.6	34672	4.9	4.8	70C20D	43705	86400.0	112.3	51055	34.9	-16.8
51C7D	41087	86400.0	9.9	36942	8.4	10.1	70C30D	-	-	-	57065	36.2	-

Instance	GUROBI			FDOR-DCS			Instance	GUROBI			FDOR-DCS		
	Best	CPU	Gap(%)	Obj.Val	CPU	Gap (%)		Best	CPU	Gap(%)	Obj.Val	CPU	Gap (%)
51C10D	45667	86400.0	22.3	43212	11.3	5.4	80C10D	40808	86400.0	22.2	38423	38.6	5.8
51C30D	47279	86400.0	186.7	59890	14.5	-26.7	80C20D	50777	86400.0	75.1	53270	41.9	-4.9
70C15D	-	86400.0	-	46818	16.6	-	80C30D	-	-	-	57285	50.0	-
70C40D	-	86400.0	-	58408	22.2	-	80C40D	-	-	-	62576	48.7	-
93C30D	-	86400.0	-	68174	26.6	-	Average	36008	67903.6		39386	18.5	1.37
93C40D	-	86400.0	-	73574	27.1	-	LE						
Average	23094	45854.2		29682	6.5	5.03	39C7D	22361	86400.0	4.7	22164	11.7	0.8
							39C10D	26774	86400.0	18.5	27191	13.5	-1.5
							39C14D	30214	86400.0	57.5	31757	15.9	-4.8
							Average	26449	86400		27037	13.7	-1.8

8.17 Results for GSVNTS

We conducted computational experiments for GSVNTS with the instance sets discussed in Section 8.1 and present our results in this section. We compare our results with the results of the commercial solver. The results of GSVNTS are tabulated in Table 8.17. Since there is an uncertainty inside the algorithm, we run GSVNTS for 5 times and report the average objective value and average run time.

GSVNTS algorithm outperforms the commercial solvers in each instance size of each election groups, as represented in Table 8.17.

Table 8.17 Comparison of GUROBI with GSVNTS for all instance sizes

Instance	GUROBI			GSVNTS			Instance	GUROBI			GSVNTS		
	Best	CPU	Gap(%)	Obj.Val	CPU	Gap (%)		Best	CPU	Gap(%)	Obj.Val	CPU	Gap (%)
PE.I							PE.II						
6C2D	7110	0.1	0.0	7110	0.1	0.0	20C5D	25118	239.2	0.0	25118	40.0	0.0
6C3D	8181	0.1	0.0	8181	0.1	0.0	20C7D	27523	1995.9	0.0	27523	31.4	0.0
7C2D	9629	0.2	0.0	9629	0.1	0.0	30C5D	16635	709.9	0.0	16635	35.2	0.0
7C4D	11597	0.4	0.0	11597	2.4	0.0	30C7D	18855	28216.8	0.0	18855	52.8	0.0
9C3D	10939	0.5	0.0	10939	3.2	0.0	30C10D	21251	86400.0	5.9	21972	847.3	-3.3
9C4D	11668	1.3	0.0	11668	6.5	0.0	40C7D	32811	86400.0	20.1	35360	554.2	-7.7
12C5D	14575	6.0	0.0	14575	10.3	0.0	40C10D	37851	86400.0	3.8	38205	1095.0	-0.9
15C7D	17240	551.3	0.0	17240	25.6	0.0	50C7D	32829	86400.0	1.6	33156	810.2	-0.8
15C10D	18759	30458.5	0.0	18759	70.1	0.0	50C10D	38098	86400.0	11.8	39225	1271.3	-2.9
21C7D	19138	6705.3	0.0	19138	39.5	0.0	50C15D	44098	86400.0	35.6	48604	2371.6	-10.2
21C10D	21904	86400.0	6.8	21612	680.0	0.1	60C7D	40480	86400.0	2.5	40290	957.8	0.4
30C7D	29427	20670.3	0.0	29427	70.1	0.0	60C10D	48270	86400.0	7.0	48936	1581.0	-1.3
30C10D	35013	86400.0	5.9	35001	1045.2	0.0	60C20D	50559	86400.0	80.1	65640	2863.5	-29.8
40C7D	30086	86400.0	4.0	29315	864.5	2.5	70C10D	42474	86400.0	13.9	42152	2433.4	0.7
40C10D	36409	86400.0	12.6	35614	1281.9	2.1	70C20D	43705	86400.0	112.3	56730	3174.2	-29.8
51C7D	41087	86400.0	9.9	42018	822.4	-2.2	70C30D	-	-	-	61573	5824.4	-
51C10D	45667	86400.0	22.3	47259	1554.0	-3.4	80C10D	40808	86400.0	22.2	40825	2951.6	0.0

Instance	GUROBI			GSVNTS			Instance	GUROBI			GSVNTS		
	Best	CPU	Gap(%)	Obj.Val	CPU	Gap (%)		Best	CPU	Gap(%)	Obj.Val	CPU	Gap (%)
51C30D	47279	86400.0	186.7	63955	3475.2	-35.2	80C20D	50777	86400.0	75.1	65294	4830.9	-28.5
70C15D	-	86400.0	-	51878	3108.1	-	80C30D	-	-	-	59425	3574.1	-
70C40D	-	86400.0	-	60422	7150.8	-	80C40D	-	-	-	64740	6383.1	-
93C30D	-	86400.0	-	69940	5394.6	-	Average	36008	67903.6		42513	2084.1	-6.71
93C40D	-	86400.0	-	78685	8450.5	-							
Average	23094	45854.2		31544	1547.9	-2.00	LE						
							39C7D	22361	86400.0	4.7	22805	718.3	-0.1
							39C10D	26774	86400.0	18.5	28059	1158.0	-4.7
							39C14D	30214	86400.0	57.5	34732	2390.5	-14.9
							Average	26449	86400		28532	1422.2	-6.56

8.18 Performance of GSVNTS

Contrary to Gurobi, GSVNTS performs in a fairly robust fashion. The performance and average runtime of GSVNTS for all instance sets are represented in Figure 8.4 - Figure 8.9.

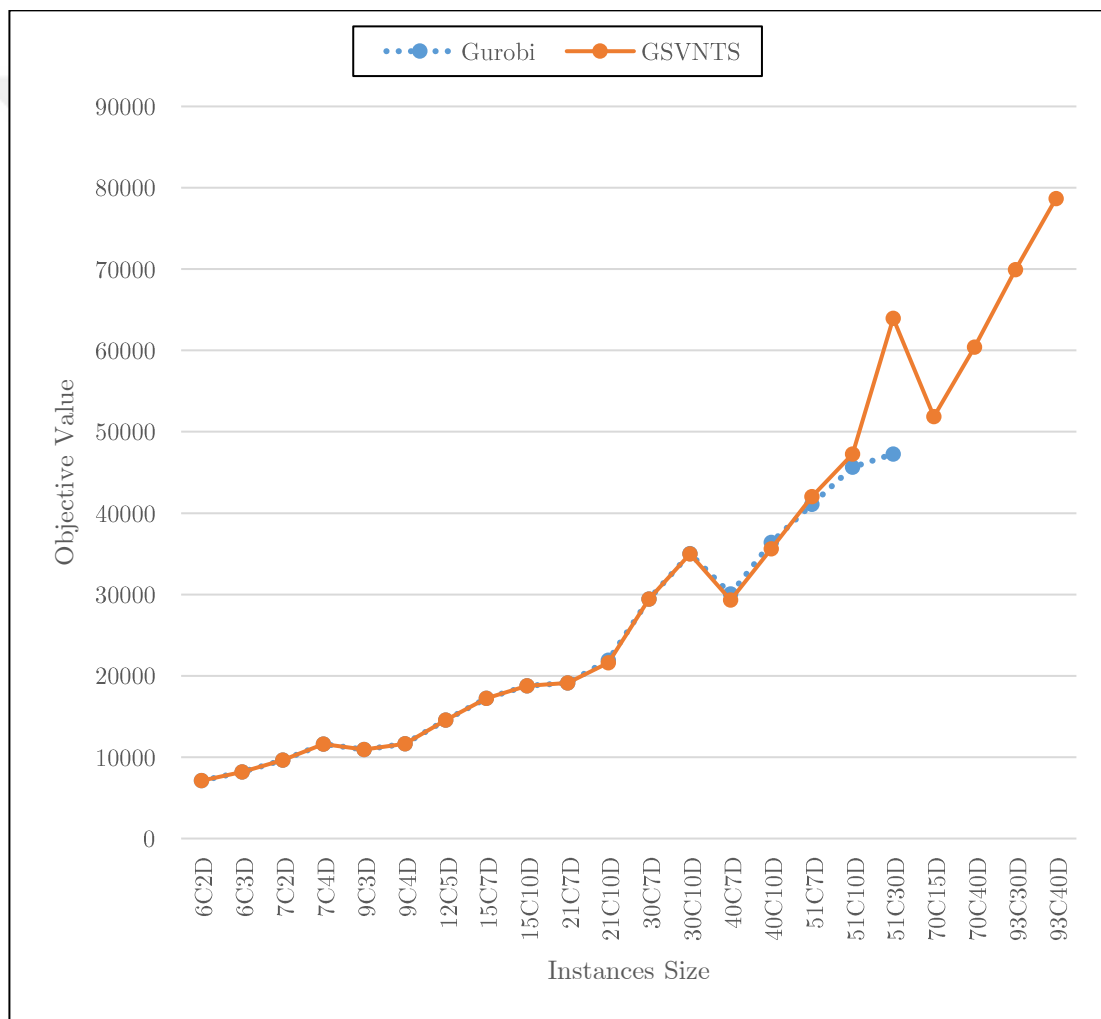


Figure 8.4 Performance of GSVNTS (PE.I)

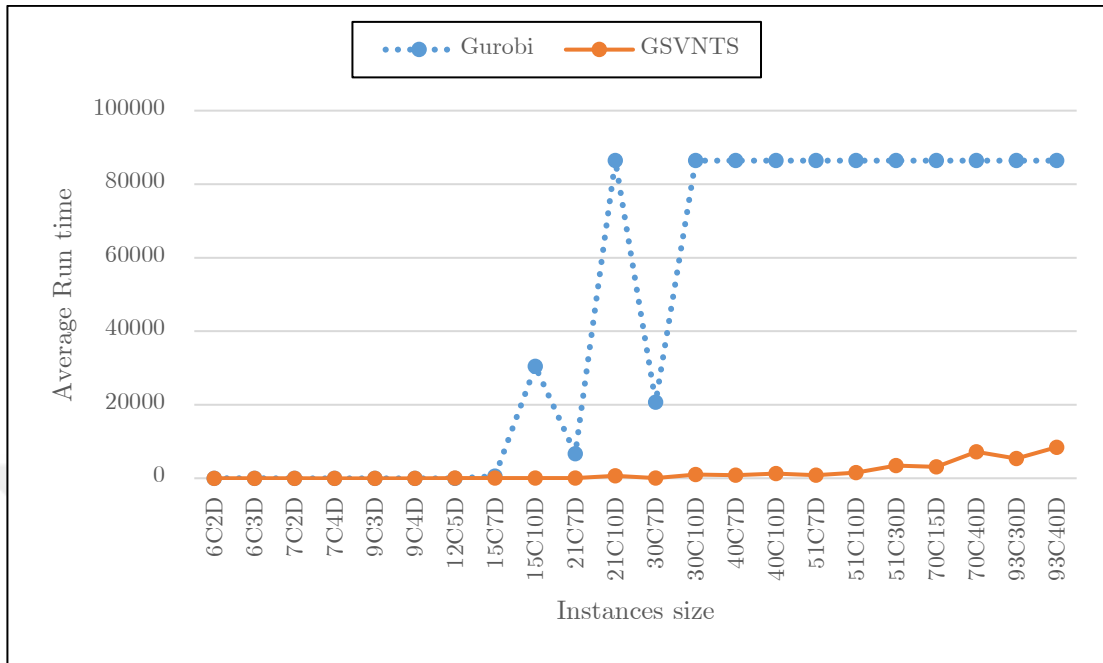


Figure 8.5 Average runtime of GSVNTS (PE.I)

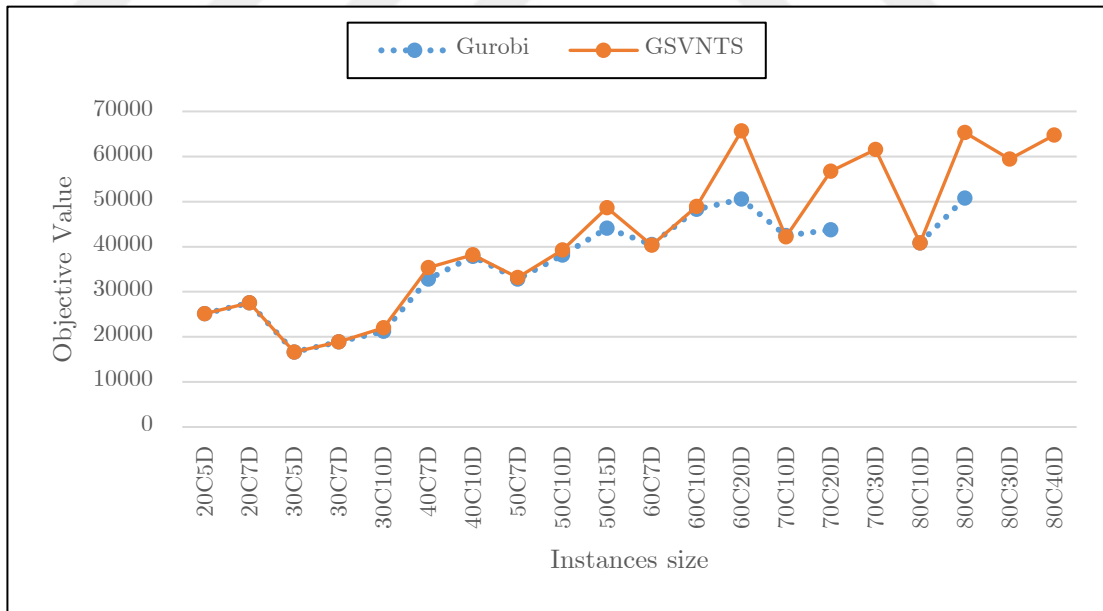


Figure 8.6 Performance of GSVNTS (PE.II)

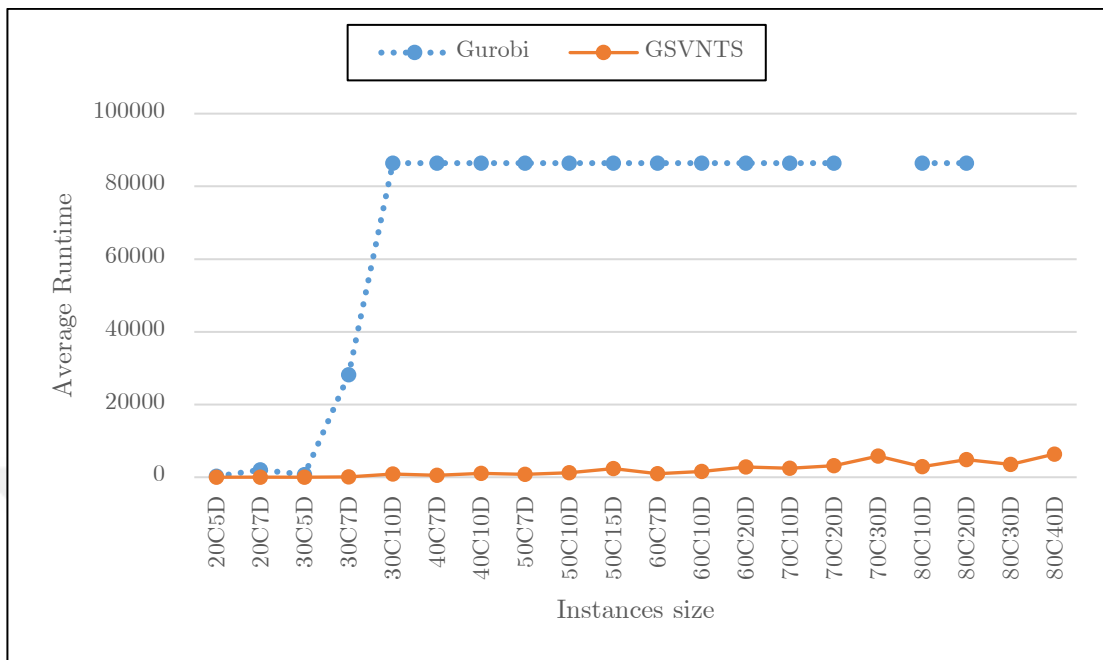


Figure 8.7 Average runtime of GSVNTS (PE.II)

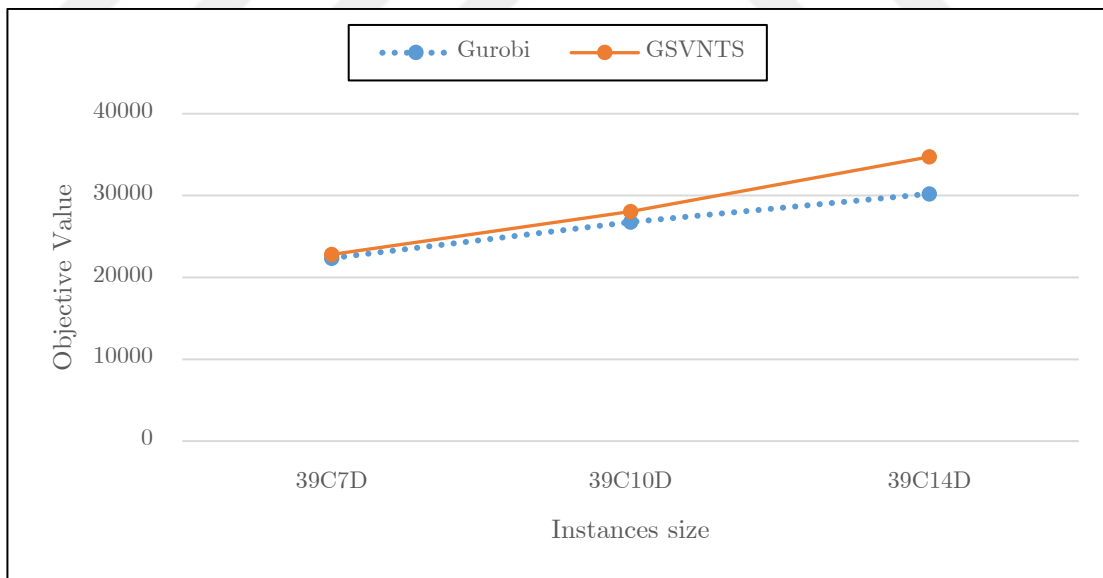


Figure 8.8 Performance of GSVNTS (LE)

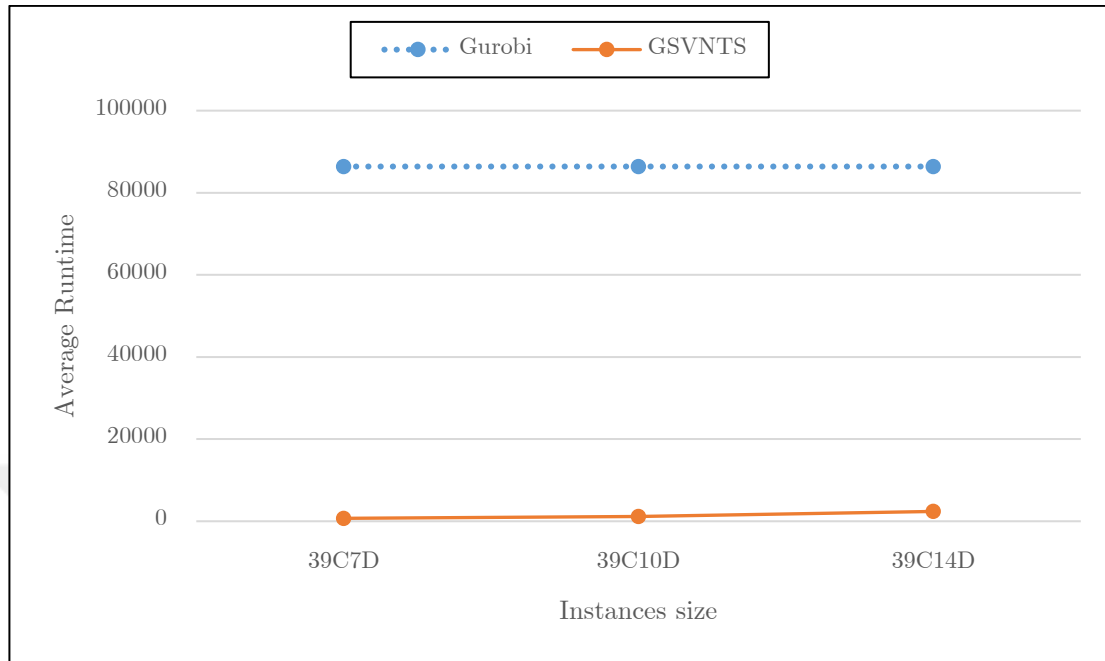


Figure 8.9 Average runtime of GSVNTS (LE)

8.19 Comparison of GSVNTS with Party's actual meeting plan

To assess the quality of the solutions proposed by GSVNTS, it is useful to compare the objective values of GSVNTS solutions for the real-life instance 70C-40D with the political party's actual plans as well. Moreover, in order to show the efficiency of GSVNTS, the results of deterministic mathematical models are proposed. Such a comparison highlight the actual need to tackle this problem. To do this, we retrieved the main opposition party's realized meetings prior to the general election in June 2015 in Turkey. In the light of these meetings, we created our large size instance with 70 cities and a campaign period of 40 days.

In order to make a fair comparison, we also define a "Reward-Only" scenario where we ignore the traveling costs and relax the following three

constraints in our assumptions. The first constraint was forcing the politician to hold at least one meeting every day. However, in the actual meeting schedule of the party, there were two meeting-free days. The second constraint was forcing the politician to end the campaign at the campaign base. We also relaxed this constraint since the actual campaign of the party back in June 2015 had not been completed in Ankara. The last constraint forced the politician to the campaign base frequently. Table 8.18 illustrates the results of GSVNTS, MPTPP and actual party's plan on the 70C-40D instance. It shows that Gurobi is not able to find an optimal solution in three days. However, the best feasible MPTPP solution reported by GUROBI bears a net benefit that is about 90% greater than the net benefit accrued by the end of the actual campaign plan of the party.

In the actual plan there are three meetings in Istanbul, Ankara, and Mersin each, two meetings in Izmir, and one meeting in the remaining cities each. However, the best feasible MPTPP solution prescribes three meetings in Istanbul, Ankara, Izmir, and Mersin each, two meetings in the majority of midsize cities such as Adana, Balikesir, Bursa, Çanakkale, Hatay, Konya, Zonguldak, Uşak, etc., and one meeting in the remaining cities. The results highlight the massive advantage of solving the MPTPP for the maximization of the net benefit obtained from an election campaign that spans an extended period.

Table 8.18 Comparison of GSVNTS with Party's actual meeting plan

		A real-life instance of 70C40D			
		Obj. Val.	Gap(%)	# of Meetings	CPU(s)
	MPTPP	LB = 46640	60.3	75	259200
	GSVNTS	58408	–	96	22.26
	Party's Plan	24534	–	77	n/a
Rew- Only	MPTPP	LB = 68399	56.1	65	259200
	GSVNTS	94044	–	102	34.15
	Party's Plan	64124	–	77	n/a

For the reward-only scenario of the problem under study, both MPTPP and GSVNTS outperform the actual party's plan. Considering the number of meetings as a performance measure, the solution obtained by GSVNTS holds 19 more meetings compared with the party's plan. This difference is more significant in the reward-only scenario where GSVNTS holds 25 more meetings. Note that in both cases, the objective values obtained by GSVNTS in less than 35 seconds are much better than the one of the party's plan which indicates the effectiveness of the scheduling and routing in the proposed algorithm.

Chapter 9

CONCLUSIONS AND FUTURE WORK

In this study, we introduce a novel logistical problem which we call the Roaming Salesman Problem (RSP). This problem can be classified as a multi-period version of the prize-collecting traveling salesman problem with dynamic profits, repeat visits to certain customer nodes, arbitrary depot nodes, and three types of time restricted tours. In addition, the campaigner can stay overnight in any arbitrary city and resume his/her daily tour there the next morning. This extraordinary feature adds another level of complexity to the model of the problem. This problem arises in election logistics where there exist no fixed depots and daily tours do not have to start and end at the same city.

We propose an innovative MILP formulation followed by an extensive scenario analysis. We also suggest an effective two-phase matheuristic approach consisting of two primary components: a city selection step and a route generation step. The proposed matheuristic (FDOR) decomposes the original mixed-integer linear programming formulation into as many subproblems as the number of days, where each subproblem depends on how frequently the campaign base is to be visited throughout the campaign duration. This decomposition strategy has led to generating next period's route without the need to track the route of each day which reduces the computational complexity of the problem massively. We tested three city selection approaches followed by associated parameter calibration experiments.

Computational results suggest that FDOR provides high-quality solutions in a significantly small amount of time.

We also present a Variable Neighborhood Search (VNS) complemented with Tabu Search (TS) for RSP. Two initial feasible solution construction algorithms are introduced. Next, this solution is improved using the proposed local search procedure. The concept of granularity is incorporated into the developed algorithm to prevent non-promising moves and thereby reduce the computing time of the neighborhood search. On the other hand, the concept of skewness modifies the basic VNS so as to explore deeper neighborhoods of the current solution by accepting nonimproving moves which lead to far enough neighboring solutions. At each iteration of the local search procedure, several feasibility checks are performed to ensure the validity of the solution. Therefore, the so-called chain feasibility of the solution is guaranteed at each iteration, which means the starting node of tomorrow's tour must match the terminal node of today's tour. The proposed GSVNTS algorithm gives competitive and acceptable results for real-world instances of RSP and its application in election logistics.

We consider a set of 95 cities and towns in Turkey and a campaign period of 40 days as our largest problem instance. Computational results using actual distance and travel time data show that the developed algorithm can find optimal and near-optimal solutions in a reasonable CPU time.

This work can be extended in many directions. RSP can be applied in different similar contexts (e.g. nurse routing problem, scheduling of music band auditions, and touristic trip planning) where the rewards are time-dependent and it is required to have different tour types during the planning horizon. Additionally, the decomposition approach we used in the proposed

matheuristic can be generalized for solving other hard combinatorial problems that could be too difficult to tackle otherwise. A relevant topic is the inclusion of rival party's meetings in the calculation of the rewards. Alternative objective functions can be evaluated as well. For instance, the party executives may want to simply hold as many meetings as possible. Another example is considering equal number of visits to the east and west sides of country. Time-windows can be considered as well to add a more realistic aspect to the problem. Moreover, alternative formulations can be investigated to improve the model solution. Finally, other powerful heuristics can be developed to improve the solution quality of the algorithm.

BIBLIOGRAPHY

- Archetti, C., Hertz, A., & Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13(1), 49-76.
- Archetti, C., Feillet, D., Hertz, A., & Speranza, M. G. (2009). The capacitated team orienteering and profitable tour problems. *Journal of the Operational Research Society*, 60(6), 831-842.
- Arkin, E. M., Mitchell, J. S., & Narasimhan, G. (1998, June). Resource-constrained geometric network optimization. In *Proceedings of the Fourteenth Annual Symposium on Computational Geometry* (pp. 307-316). ACM.
- Awerbuch, B., Azar, Y., Blum, A., & Vempala, S. (1998). New approximation guarantees for minimum-weight k -trees and prize-collecting salesmen. *SIAM Journal on Computing*, 28(1), 254-262.
- Balas, E. (1989). The prize collecting traveling salesman problem. *Networks*, 19(6), 621-636.
- Balas, E., & Martin, G. (1985). ROLL-A-ROUND: Software package for scheduling the rounds of a rolling mill, ©Balas and Martin Associates, 104 Maple Heights Road, Pittsburgh, USA.

-
- Bérubé, J. F., Gendreau, M., & Potvin, J. Y. (2009). An exact ϵ -constraint method for bi-objective combinatorial optimization problems: Application to the Traveling Salesman Problem with Profits. *European journal of operational research*, 194(1), 39-50.
- Boussier, S., Feillet, D., & Gendreau, M. (2007). An exact algorithm for team orienteering problems. *Ior*, 5(3), 211-230.
- Bräysy, O., & Gendreau, M. (2005). Vehicle routing problem with time windows, Part II: Metaheuristics. *Transportation science*, 39(1), 119-139.
- Burke, E. K., Cowling, P. I., & Keuthen, R. (2001, April). Effective local and guided variable neighbourhood search methods for the asymmetric travelling salesman problem. In *Workshops on Applications of Evolutionary Computation* (pp. 203-212). Springer, Berlin, Heidelberg.
- Butt, S. E., & Cavalier, T. M. (1994). A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, 21(1), 101-111.
- Butt, S. E., & Ryan, D. M. (1999). An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, 26(4), 427-441.
- Cacchiani, V., Hemmelmayr, V. C., and Tricoire, F. (2014). A set-covering based heuristic algorithm for the periodic vehicle routing problem. *Discrete Applied Mathematics*, 163, 53-64.

-
- Carrabs, F., Cordeau, J. F., & Laporte, G. (2007). Variable neighborhood search for the pickup and delivery traveling salesman problem with LIFO loading. *INFORMS Journal on Computing*, 19(4), 618-632.
- Chao, I. M., Golden, B. L., & Wasil, E. A. (1996). A fast and effective heuristic for the orienteering problem. *European journal of operational research*, 88(3), 475-489.
- Cordeau, J. F., Gendreau, M., & Laporte, G. (1997). A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2), 105-119.
- Crispim, J., & Brandao, J. (2001, July). Reactive tabu search and variable neighborhood descent applied to the vehicle routing problem with backhauls. In *Proceedings of the 4th Metaheuristics International Conference*, Porto (Vol. 1101, pp. 631-636).
- Dell'Amico, M., Maffioli, F., & Sciomachen, A. (1998). A lagrangian heuristic for the prize collecting travelling salesman problem. *Annals of Operations Research*, 81, 289-306.
- Dell'Amico, M., Maffioli, F., & Värbrand, P. (1995). On prize-collecting tours and the asymmetric travelling salesman problem. *International Transactions in Operational Research*, 2(3), 297-308.
- Desrochers, M., & Laporte, G. (1991). Improvements and extensions to the Miller-Tucker-Zemlin subtour elimination constraints. *Operations Research Letters*, 10(1), 27-36.

-
- Feillet, D., Dejax, P., & Gendreau, M. (2005). Traveling salesman problems with profits. *Transportation Science*, 39(2), 188-205.
- Fischetti, M., Salazar-González, J. J., & Toth, P. (2007). The generalized traveling salesman and orienteering problems. In: Gutin, G., & Punnen, A. P. (Eds.). *The Traveling Salesman Problem and Its Variations* (pp. 609-662). *Combinatorial Optimization* Vol. 12. Springer Science+Business Media, LLC.
- Fischetti, M., & Toth, P. (1988). An additive approach for the optimal solution of the prize collecting traveling salesman problem. *Vehicle routing: Methods and studies*, 231, 319-343.
- Fleszar, K., Osman, I. H., & Hindi, K. S. (2009). A variable neighbourhood search algorithm for the open vehicle routing problem. *European Journal of Operational Research*, 195(3), 803-809.
- García-López, F., Melián-Batista, B., Moreno-Pérez, J. A., & Moreno-Vega, J. M. (2002). The parallel variable neighborhood search for the p-median problem. *Journal of Heuristics*, 8(3), 375-388.
- Garey, M. R., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness* (Series of Books in the Mathematical Sciences, Vol. 29). New York: W. H. Freeman.
- Gavish, B., & Graves, S. C. (1978). The travelling salesman problem and related problems.

-
- Geiger, M. J., Wenger, W., & Habenicht, W. (2007, April). Interactive utility maximization in multi-objective vehicle routing problems: a "decision maker in the loop"-approach. In *Computational Intelligence in Multicriteria Decision Making, IEEE Symposium on* (pp. 178-184). IEEE.
- Gendreau, M., Laporte, G., & Semet, F. (1998). A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, 32(4), 263-273.
- Glover, F. (1989). Tabu search—part I. *ORSA Journal on computing*, 1(3), 190-206.
- Glover, F. (1990). Tabu search—part II. *ORSA Journal on computing*, 2(1), 4-32.
- Glover, F., & Laguna, M. (1998). Tabu search. In *Handbook of combinatorial optimization* (pp. 2093-2229). Springer, Boston, MA.
- Golden, B. L., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, 34(3), 307-318.
- Golden, B. L., Wang, Q., & Liu, L. (1988). A multifaceted heuristic for the orienteering problem. *Naval Research Logistics*, 35(3), 359-366.
- Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2), 315-332.

-
- Gurobi Documentation (May 19, 2018). Version 7.5. Gurobi Optimization, LLC. Houston, USA. < <http://www.gurobi.com/documentation/>> (accessed May 2018).
- Gutin, G., & Punnen, A. P. (Eds.). (2007). The Traveling Salesman Problem and Its Variations. Combinatorial Optimization Vol. 12. *Springer Science+Business Media, LLC*.
- Halvorsen-Weare, E. E., and Fagerholt, K. (2013). Routing and scheduling in a liquefied natural gas shipping problem with inventory and berth constraints. *Annals of Operations Research*, 1-20.
- Hansen, P., & Mladenović, N. (1999). An introduction to variable neighborhood search. In *Meta-heuristics* (pp. 433-458). Springer, Boston, MA.
- Hansen, P., & Mladenović, N. (2001). Variable neighborhood search: Principles and applications. *European journal of operational research*, 130(3), 449-467.
- Hansen, P., Mladenović, N., & Pérez, J. A. M. (2010). Variable neighbourhood search: methods and applications. *Annals of Operations Research*, 175(1), 367-407.
- Hayes, M., & Norman, J. M. (1984). Dynamic programming in orienteering: route choice and the siting of controls. *Journal of the Operational Research Society*, 35(9), 791-796.

-
- Hemmelmayr, V. C., Doerner, K. F., & Hartl, R. F. (2009). A variable neighborhood search heuristic for periodic routing problems. *European Journal of Operational Research*, 195(3), 791-802.
- Holland, J. H. (1975). *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*.
- Hu, B., Leitner, M., & Raidl, G. R. (2008). Combining variable neighborhood search with integer linear programming for the generalized minimum spanning tree problem. *Journal of Heuristics*, 14(5), 473-499.
- Kataoka, S., & Morito, S. (1988). An algorithm for single constraint maximum collection problem. *Journal of the Operations Research Society of Japan*, 31(4), 515-531.
- Ke, L., Archetti, C., & Feng, Z. (2008). Ants can solve the team orienteering problem. *Computers & Industrial Engineering*, 54(3), 648-665.
- Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *science*, 220(4598), 671-680.
- Kirkpatrick, S., & Toulouse, G. (1985). Configuration space analysis of travelling salesman problems. *Journal de Physique*, 46(8), 1277-1292.
- Kytöjoki, J., Nuortio, T., Bräysy, O., & Gendreau, M. (2007). An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. *Computers & operations research*, 34(9), 2743-2757.

-
- Lahyani, R., Khemakhem, M., and Semet, F. (2017). A unified matheuristic for solving multi-constrained traveling salesman problems with profits. *EURO Journal on Computational Optimization*, 5(3), 393-422.
- Lahyani, R., Khemakhem, M., and Semet, F. (2015). Rich vehicle routing problems: From a taxonomy to a definition. *European Journal of Operational Research*, 241(1), 1-14.
- Laporte, G., & Martello, S. (1990). The selective travelling salesman problem. *Discrete Applied Mathematics*, 26(2-3), 193-207.
- Liu, S. C., & Chung, C. H. (2009). A heuristic method for the vehicle routing problem with backhauls and inventory. *Journal of Intelligent Manufacturing*, 20(1), 29.
- Mansini, R., Pelizzari, M., & Wolfer, R. (2006). A granular variable neighbourhood search heuristic for the tour orienteering problem with time windows. *Technical Report R.T 2006-02-52*, University of Brescia, Italy.
- Mansini, R., & Tocchella, B. (2009). The traveling purchaser problem with budget constraint. *Computers & Operations Research*, 36(7), 2263-2274.
- Melechovský, J., Prins, C., & Calvo, R. W. (2005). A metaheuristic to solve a location-routing problem with non-linear costs. *Journal of Heuristics*, 11(5-6), 375-391.

-
- Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM)*, 7(4), 326-329.
- Mladenović, N., & Hansen, P. (1997). Variable neighborhood search. *Computers & operations research*, 24(11), 1097-1100.
- Öncan, T., Altinel, İ. K., & Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, 36(3), 637-654.
- Polacek, M., Doerner, K. F., Hartl, R. F., Kiechle, G., & Reimann, M. (2007). Scheduling periodic customer visits for a traveling salesperson. *European Journal of Operational Research*, 179(3), 823-837.
- Polacek, M., Doerner, K. F., Hartl, R. F., & Maniezzo, V. (2008). A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. *Journal of Heuristics*, 14(5), 405-423.
- Polacek, M., Hartl, R. F., Doerner, K., & Reimann, M. (2004). A variable neighborhood search for the multi depot vehicle routing problem with time windows. *Journal of heuristics*, 10(6), 613-627.
- Polat, O., Kalayci, C. B., Kulak, O., & Günther, H. O. (2015). A perturbation based variable neighborhood search heuristic for solving the vehicle routing problem with simultaneous pickup and delivery with time limit. *European Journal of Operational Research*, 242(2), 369-382.

-
- Prins, C., Prodhon, C., Ruiz, A., Soriano, P., and Wolfler Calvo, R. (2007). Solving the capacitated location-routing problem by a cooperative Lagrangean relaxation-granular tabu search heuristic. *Transportation Science*, 41(4), 470-483.
- Qiu, Y., Wang, L., Xu, X., Fang, X., & Pardalos, P. M. (2018). A variable neighborhood search heuristic algorithm for production routing problems. *Applied Soft Computing*, 66, 311-318.
- Repoussis, P. P., Paraskevopoulos, D. C., Tarantilis, C. D., & Ioannou, G. (2006, October). A reactive greedy randomized variable neighborhood tabu search for the vehicle routing problem with time windows. In *International Workshop on Hybrid Metaheuristics* (pp. 124-138). Springer, Berlin, Heidelberg.
- Rousseau, L. M., Gendreau, M., & Pesant, G. (2002). Using constraint-based operators to solve the vehicle routing problem with time windows. *Journal of heuristics*, 8(1), 43-58.
- Salhi, S (2017). *Heuristic Search: The Emerging Science of Problem Solving*. Cham, Switzerland, Springer.
- Sarasola, B., Doerner, K. F., Schmid, V., & Alba, E. (2016). Variable neighborhood search for the stochastic and dynamic vehicle routing problem. *Annals of Operations Research*, 236(2), 425-461.

-
- Shahmanzari, M., Aksen, D., & Salhi, S. (2018). A formulation and matheuristic for the Roaming Salesman Problem: Application to election logistics. *Manuscript submitted for publication*.
- Shahmanzari, M., Aksen, D., & Salhi, S. (2018). Managing election campaigns with the power of analytical modeling. *Manuscript submitted for publication*.
- Tang, H., & Miller-Hooks, E. (2005). A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6), 1379-1407.
- Thomadsen, T., Stidsen, T. (2003). The quadratic selective travelling salesman problem. *Informatics and Mathematical Modelling Technical Report 2003-17*, Technical University of Denmark.
- Todosijević, R., Urošević, D., Mladenović, N., & Hanafi, S. (2017). A general variable neighborhood search for solving the uncapacitated r-allocation p-hub median problem. *Optimization Letters*, 11(6), 1109-1121.
- Toth, P., & Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Inform Journal on computing*, 15(4), 333-346.
- Tricoire, F., Romauch, M., Doerner, K. F., & Hartl, R. F. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. *Computers & Operations Research*, 37(2), 351-367.

-
- Tsiligirides, T. (1984). Heuristic methods applied to orienteering. *Journal of the Operational Research Society*, 35(9), 797-809.
- Vansteenwegen, P., Souffriau, W., & Van Oudheusden, D. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, 209(1), 1-10.
- Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Van Oudheusden, D. (2009). A guided local search metaheuristic for the team orienteering problem. *European journal of operational research*, 196(1), 118-127.
- Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Van Oudheusden, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12), 3281-3290.

APPENDIX A

Table A.1 Characteristics of the dominant party instances

Cities	Criticality factor (CF_i)	Population in 2015 (Pop_i)	Reward (δ_i)	Meeting Duration (σ_i)
Adana	5	2183167	920	120
Adiyaman	4	602774	420	90
Afyonkarahisar	4	709015	580	90
Ağrı	2	547210	270	60
Aksaray	2	386514	250	60
Amasya	2	322167	240	60
Ankara	5	5270575	1505	120
Antalya	5	2288456	935	120
Ardahan	2	99265	210	60
Artvin	2	168370	220	60
Aydın	3	1053506	495	90
Balıkesir	5	1186688	875	120
Bartın	4	190708	440	90
Batman	2	566633	270	60
Bayburt	2	78550	210	60
Bilecik	4	212361	460	90
Bingöl	2	267184	230	60
Bitlis	4	340449	480	90
Bolu	4	291095	480	90
Burdur	4	258339	460	90
Bursa	5	2842547	1040	120
Çanakkale	2	513341	270	60
Çankırı	2	180945	220	60
Çorum	4	525180	540	90
Denizli	3	993442	495	90

Cities	Criticality factor (CF_i)	Population in 2015 (Pop_i)	Reward (δ_i)	Meeting Duration (σ_i)
Diyarbakır	4	1654196	820	90
Düzce	5	360388	625	120
Edirne	2	402537	250	60
Elazığ	3	574304	405	90
Erzincan	3	222918	345	90
Erzurum	3	762321	450	90
Eskişehir	3	826716	465	90
Gaziantep	5	1931836	1125	120
Giresun	4	426686	500	90
Gümüşhane	2	151449	220	60
Hakkari	2	278775	240	60
Hatay	5	1533507	1000	120
Iğdır	2	192435	220	60
Isparta	4	421766	500	90
İstanbul	5	14657434	2370	120
İzmir	5	4168415	1295	120
Kahramanmaraş	3	1096610	510	90
Karabük	2	236978	230	60
Karaman	2	242196	230	60
Kars	3	292660	360	90
Kastamonu	3	372633	375	90
Kayseri	5	1341056	925	120
Kilis	2	130655	220	60
Kırıkkale	3	270271	345	90
Kırklareli	2	346973	240	60
Kırşehir	3	225562	345	90
Kocaeli	5	1780055	1075	120
Konya	5	2130544	905	120
Kütahya	3	571463	405	90
Malatya	5	772904	750	120

Cities	Criticality factor (CF_i)	Population in 2015 (Pop_i)	Reward (δ_i)	Meeting Duration (σ_i)
Manisa	3	1380366	570	90
Mardin	3	796591	450	90
Mersin	3	1745221	630	90
Muğla	3	908877	480	90
Muş	2	408728	250	60
Nevşehir	3	286767	360	90
Niğde	4	346114	480	90
Ordu	3	728949	435	90
Osmaniye	5	512873	675	120
Rize	3	328979	360	90
Sakarya	5	953181	800	120
Samsun	5	1279884	900	120
Şanlıurfa	3	1892320	660	90
Siirt	2	320351	240	60
Sinop	4	204133	460	90
Şırnak	2	490184	260	60
Sivas	4	618617	560	90
Tekirdağ	3	937910	480	90
Tokat	3	593990	420	90
Trabzon	3	768417	450	90
Tunceli	2	86076	210	60
Uşak	2	353048	240	60
Van	4	1096397	680	90
Yalova	4	233009	460	90
Yozgat	3	419440	375	90
Zonguldak	5	595907	700	120

Table A.2 Characteristics of the main opposition party instances

Cities	Criticality factor (CF_i)	Population in 2015 (Pop_i)	Reward (δ_i)	Meeting Duration (σ_i)
Adana	5	2183167	920	120
Adıyaman	2	602774	280	60
Afyonkarahisar	3	709015	435	90
Ağrı	2	547210	270	60
Aksaray	2	386514	250	60
Amasya	3	322167	360	90
Ankara	5	5270575	1505	120
Antalya	5	2288456	935	120
Ardahan	2	99265	210	60
Artvin	2	168370	220	60
Aydın	4	1053506	660	90
Balıkesir	5	1186688	875	120
Bartın	3	190708	330	90
Batman	2	566633	270	60
Bayburt	2	78550	210	60
Bilecik	2	212361	230	60
Bingöl	2	267184	230	60
Bitlis	2	340449	240	60
Bolu	3	291095	360	90
Burdur	2	258339	230	60
Bursa	5	2842547	1040	120
Çanakkale	3	513341	405	90
Çankırı	2	180945	220	60
Çorum	2	525180	270	60
Denizli	4	993442	660	90
Diyarbakır	2	1654196	410	60
Düzce	2	360388	250	60
Edirne	4	402537	500	90
Elazığ	2	574304	270	60

Cities	Criticality factor (CF_i)	Population in 2015 (Pop_i)	Reward (δ_i)	Meeting Duration (σ_i)
Erzincan	4	222918	460	90
Erzurum	2	762321	300	60
Eskişehir	3	826716	465	90
Gaziantep	3	1931836	675	90
Giresun	3	426686	375	90
Gümüşhane	2	151449	220	60
Hakkari	2	278775	240	60
Hatay	5	1533507	1000	120
Iğdır	2	192435	220	60
Isparta	3	421766	375	90
İstanbul	5	14657434	2370	120
İzmir	5	4168415	1295	120
Kahramanmaraş	4	1096610	680	90
Karabük	2	236978	230	60
Karaman	2	242196	230	60
Kars	4	292660	480	90
Kastamonu	4	372633	500	90
Kayseri	2	1341056	370	60
Kilis	2	130655	220	60
Kırıkkale	2	270271	230	60
Kırklareli	4	346973	480	90
Kırşehir	2	225562	230	60
Kocaeli	3	1780055	645	90
Konya	2	2130544	362	60
Kütahya	2	571463	270	60
Malatya	2	772904	300	60
Manisa	3	1380366	570	90
Mardin	2	796591	300	60
Mersin	3	1745221	630	90
Muğla	5	908877	800	120

Cities	Criticality factor (CF_i)	Population in 2015 (Pop_i)	Reward (δ_i)	Meeting Duration (σ_i)
Muş	2	408728	250	60
Nevşehir	2	286767	240	60
Niğde	3	346114	360	90
Ordu	4	728949	580	90
Osmaniye	2	512873	270	60
Rize	2	328979	240	60
Sakarya	3	953181	480	90
Samsun	3	1279884	540	90
Şanlıurfa	2	1892320	440	60
Siirt	2	320351	240	60
Sinop	3	204133	345	90
Şırnak	2	490184	260	60
Sivas	3	618617	420	90
Tekirdağ	4	937910	640	90
Tokat	2	593990	280	60
Trabzon	2	768417	300	60
Tunceli	3	86076	315	90
Uşak	2	353048	240	60
Van	2	1096397	340	60
Yalova	3	233009	345	90
Yozgat	2	419440	250	60
Zonguldak	4	595907	560	90

APPENDIX B

The shaded areas in Figure B.1 denote those cities of Turkey which are included in the 10-day-long campaign period. *Full-MILP* results on **40C-10D** are provided in Figure B.2, where the dots and numbers on a city represent the number of meetings realized and day of meetings, respectively. Table B.1 reveals the daily tours where (M) indicates a meeting. The best gap obtained for this particular instance was 12.6% in 24 hours of CPU time.



Figure B.1 Geographical distribution of 40 cities (shaded areas)

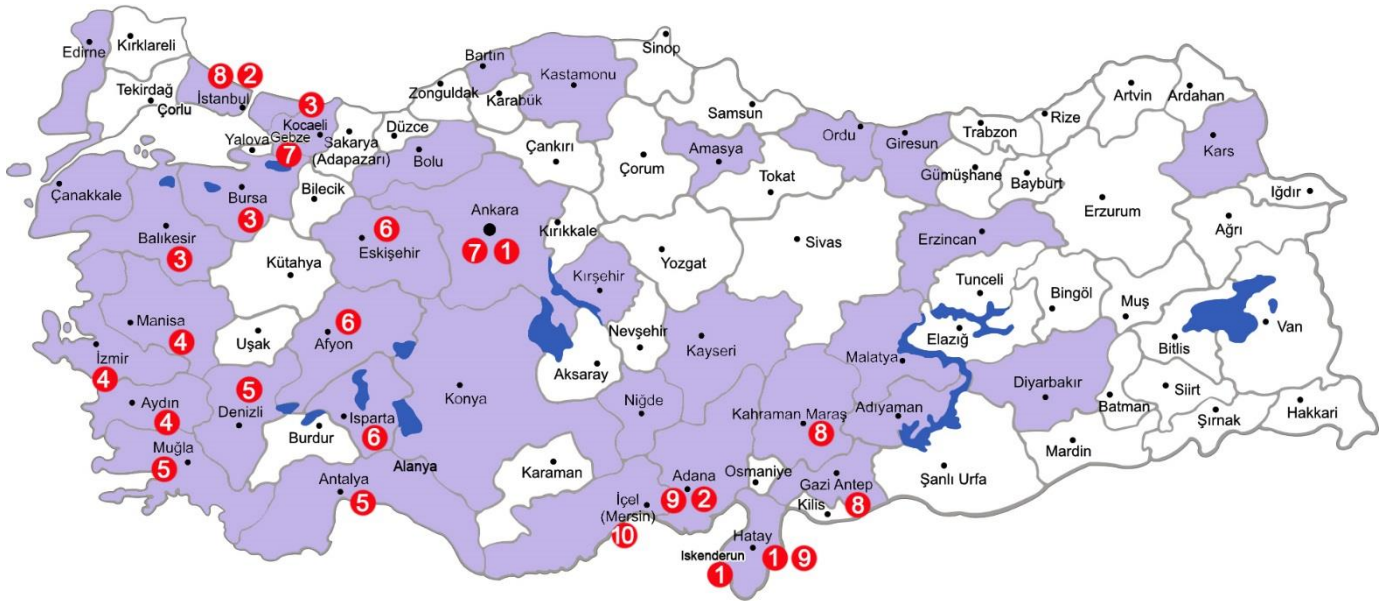


Figure B.2 Cities with meetings

Table B.1 Daily tours of the instance 40C-10D

Days	Route
1	Ankara (M) → Hatay (M) → İskenderun (M)
2	İskenderun → Adana (M) → Istanbul (M)
3	Istanbul → Kocaeli (M) → Bursa (M) → Balıkesir (M)
4	Balıkesir → Manisa (M) → İzmir (M) → Aydın (M) → Muğla
5	Muğla (M) → Denizli (M) → Antalya (M) → Isparta
6	Isparta (M) → Afyonkarahisar (M) → Eskişehir (M) → Ankara
7	Ankara (M) → Gebze (M) → Istanbul
8	Istanbul (M) → Gaziantep (M) → Kahramanmaraş (M)
9	Kahramanmaraş → Hatay (M) → Adana (M) → Mersin
10	Mersin (M)