Formulation and a Two-Phase Matheuristic for the Roaming Salesman Problem: Application to Election Logistics

Masoud Shahmanzari

Graduate School of Business, Istanbul Şehir University, İstanbul, Turkey

Deniz Aksen*

College of Administrative Sciences and Economics, Koç University, İstanbul, Turkey

Saïd Salhi

Kent Business School, Centre for Logistics and Heuristic Optimisation, University of Kent, Canterbury, United Kingdom

Revised on 12 July 2019

ABSTRACT

In this paper we investigate a novel logistical problem. The goal is to determine daily tours for a traveling salesperson who collects rewards from activities in cities during a fixed campaign period. We refer to this problem as the *Roaming Salesman Problem* (RSP) motivated by 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 arc costs and time-dependent node rewards. 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 city selection while the second phase focuses on route generation. The latter capitalizes on an integer program to construct an optimal route among selected cities on a given day. The proposed matheuristic decomposes the RSP into as many subproblems as the number of campaign days. Computational results show that our approach provides near-optimal solutions in significantly shorter times compared to commercial solvers.

Keywords: *Routing, Roaming salesman problem, Election logistics, Matheuristic, Campaign planning.*

1. Introduction

In this paper, we study a logistical problem arising in promotion and marketing campaigns where the campaigner and his/her team needs to plan an efficient schedule throughout the campaign to maximize the total reward by visiting appropriate cities. This problem has a wider range of applications including election logistics, touristic trip 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). It involves a salesperson who collects rewards from activities performed in selected cities during a fixed campaign period. The goal in the RSP is to find an optimal or the 'best' schedule of daily tours for a

* Corresponding author. Tel.: +90 (1) 338 1684, Fax.: +90 (1) 338 1653, Email: daksen@ku.edu.tr (D. Aksen)

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 salesperson. 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. each day the salesperson 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 salesperson 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 originate where yesterday's tour terminated. Hence, the salesperson 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 exists a subset of nodes which may host more than one activity during the campaign, hence can be visited more than once.

One of the main differences between RSP and similar routing problems in the literature lies in time-dependent rewards. This characteristic makes the problem applicable to various situations. For instance, in the planning of touristic trips the scores of visited sites or tour stops can be modeled as time-dependent rewards. This feature can be incorporated within a daily or weekly framework depending on the problem. Some places are more appealing to visit during the day time while others have better sights in the evening or at night. Also, some destinations may become more crowded, thus less attractive as we get closer to the end of the week. RSP is able to encapsulate this kind of dynamic rewards. It can capture the possibility of repeated visits to certain attractions during an extended tour as well.

In this paper, we address a novel adaptation of RSP to election logistics which revolves around a politician holding meetings in various cities during a given campaign period. The problem generalizes the traveling salesman problem (TSP) by extending the planning horizon to τ days; hence, it corresponds to a multi-period problem. The RSP can be defined as follows. Consider a set of nodes $\mathbf{N} = \mathbf{V} \cup \{0\}$ including a fictitious city (indexed as 0) where $\mathbf{V} = \{1,...,n\}$ indicates the set of cities inclusive of a starting city (indexed as 1) and a set of days $\mathbf{T} = \{1,...,\tau\}$. Each city is associated with a nonnegative reward of π_i referred to as the *base reward*. In 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 collecting reward. The base reward of a city 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.

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 $1 \le s \le t-1$.

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 traveling costs and traveling times satisfy the triangular inequality. The time spent by the salesperson (also referred to as the *campaigner* in the sequel) for an activity in city $i \in \mathbf{V}$ is shown by σ_i. The maximum 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 starts in the base 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 salesperson cannot be away from the campaign base for more than κ consecutive days.

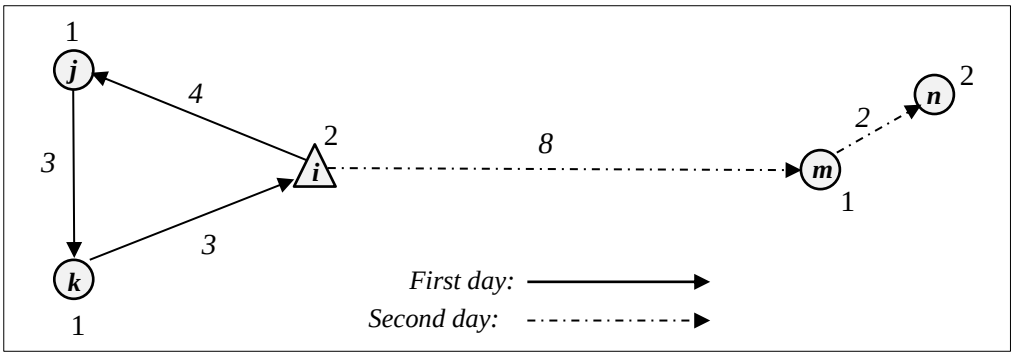

Figure 1. An instance with both closed and open tours.

A distinctive feature of the RSP is that there are three possible types of daily tours during the campaign. Type 1: Open Tour. Type 2: Closed Tour. Type 3: No 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. The travel times and activity times

are written on the arcs and next to the nodes, respectively, both in hours. As shown in Figure 1, 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 holding 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.

By ignoring the activity times σ_i, taking the campaign duration as $\tau = 1$ day, setting π_i and $T_{\max}$ to sufficiently large values, e.g. by setting $\pi_i = \max_{j \in \mathbf{N}}\{d_{ij}\} + \max_{j \in \mathbf{N}}\{d_{ji}\}$ and $T_{\max} = |\mathbf{V}| \max_{i,j \in \mathbf{N}}\{d_{ij}\}$, a given generic TSP instance can be reduced to the associated RSP instance in polynomial time. TSP is a well-known $\mathcal{NP}$-hard combinatorial optimization problem (Garey and Johnson, 1979). RSP is a generalization of the TSP and is therefore also $\mathcal{NP}$-hard. Furthermore, RSP is more complex than TSP since the selection of the terminal node of each day, different tour types and time-dependent rewards are included in the optimization problem as well. This leads to the conclusion that the RSP is also $\mathcal{NP}$-hard, and thus cannot be solved in polynomial time to optimality. Motivated by this challenge, we propose a simple but efficient two-phase matheuristic method which we call *Finding Daily Optimal Routes* (FDOR). For each day 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 which is 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 matheuristic can provide for medium- and large-size instances a promising bundle of accommodation and activity schedules that are complemented by daily routing plans. Actually, FDOR achieves this in remarkably short solution times. Thereby, it can help campaign planners in their decision-making.

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

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.
3. The development of a two-phase matheuristic to solve large-size instances of it.
4. A real-life application of the problem to election logistics covering 81 provinces and 12 highly populated towns of Turkey.

The outline of the paper is as follows. In Section 2 we review the related literature. In Section 3 we present the mathematical formulation of RSP. In Section 4 we present the proposed two-phase matheuristic approach FDOR. We discuss our computational results in Section 5 in the framework of a case study involving a great deal of cities and towns from Turkey. Finally, Section 6 summarizes our results and recommends future research directions.

2. Literature review

The RSP is derived from the well-known *traveling salesman problem* (TSP) which is one of the most famous $\mathcal{NP}$-hard combinatorial optimization problem in the literature. A widely accepted and often cited classification of the TSP and its variants has been presented in Gutin and Punnen (2007). The first TSP variant that is closely related to the RSP is the *periodic traveling salesman problem* (PTSP). Many variations of the 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 salesperson available every day. The goal is to generate a tour for each of the *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 the PTSP can be found in Cordeau et al. (1997).

The other TSP variants resembling the 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 the 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.

Variant 1: PCTSP

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 salesperson who travels between nodes i and j at cost c_{ij}, earns a prize p_k from every visited node k and pays a penalty γ_h for each unvisited node 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 $\pi_{\min}$. A feasible circuit either in the PCTSP or the other TSPP variants visits each node 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 the PCTSP is the *quota TSP* (QTSP) which was first studied in Awerbuch et al. (1998).

Variant 2: PTP

PTP 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 nodes, the sum of incurred penalties is also deducted from the total amount of collected prizes to yield the net profit. The PTP was initially introduced 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).

Variant 3: OP

OP is evidently the most extensively studied variant of the TSPP class. The OP seeks to find a circuit or a path on a graph with n nodes 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 the OP can be viewed in this regard as a combination between the *knapsack problem* (KP) and the TSP. Feillet et al. (2005) point to the equivalence between the path-seeking and circuit-seeking versions of the problem. Pioneering studies of the 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 the TSP and to the Hamiltonian circuit problem, respectively. Applications in the literature of this selective routing problem span a wide range of areas. Labadie et al. (2012) solve single- and multi-tour versions of the OP with time window constraints which dictate that the service at each node start within a predefined time window. An early arrival to a given node leads to waiting times, while a late arrival causes infeasibility. The authors devise a matheuristic which consists of a linear programming (LP)-based granular variable neighborhood search. With this method they manage to obtain the best known solutions for 25 benchmark instances in the literature. Very recently, Archetti et al. (2018) introduce the Set Orienteering Problem which is a generalization of the OP where customers are grouped in clusters and a profit is associated with each cluster.

Within the generic class of TSPPs, the variant that seems most relevant and similar to our problem 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 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 the 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 problem in that each day of the planning horizon is associated with a separate tour. However, our problem 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.

Recent progress in CPU technologies and commercial solvers enables us to solve different MILP models to optimality or near to optimality in short solution times. This leads to the design of a matheuristic, a heuristic that incorporate stages where mathematical programming models are used. In brief, a matheuristic is a heuristic or metaheuristic algorithm which solves at least one of its steps using an exact method such as mathematical programming or dynamic programming. See Salhi (2017) for more details.

In the literature, there are a couple of articles that use matheuristic methods in order to solve routing problems. Prins et al. (2007) propose a matheuristic approach to solve the capacitated location-routing problem. The original problem is decomposed into two phases; location decisions and routing. The location decision problem is solved as a facility location problem using an exact method whereas a tabu search is adopted builds the routes based on given facility set. Halvorsen-Weare and Fagerholt (2013) investigate a routing and scheduling problem emerging in naval logistics. They employ a matheuristic method which separates the scheduling decisions from the routing decisions. The routing problem is solved through a local search heuristic while the scheduling problem is tackled through the exact solution of a MILP formulation. Hemmelmayr et al. (2013) investigate a two-phase matheuristic approach for the problem of determining the size of waste bins 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 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. It includes exact procedures for the examination of loading neighborhoods. A review of different heuristic methods including matheuristics can be found in Salhi (2017).

Before concluding this section we would like to make a remark in regard to time-dependent routing problems in the literature. Although there exist a huge number of papers where the travel duration or cost of an arc depends on the actual time of travel, studies involving other time-dependent parameters are not in abundance. We are aware of a recent paper by Taş et al. (2016) which investigates a variant of the TSP with time-dependent service times. In the proposed setting the required service duration at a customer node is not fixed, but determined as a function of the time at which service starts for that customer. Angelelli et al. (2017) introduce the Traveling Purchaser Problem with time-dependent quantities. The authors assume that

the obtainable quantities of all products that can be purchased from the available markets decrease linearly over time. To the best of our knowledge, none of the previous research has looked into time-dependent rewards collected from customers in a multi-period and multi-visit framework. Our present study makes a novel contribution to the time-dependent routing literature in this respect.

3. Notation and formulation

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:

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

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

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

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 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 $1 \le s < t$, we have $R_{its}=0$ for $t \le s \le \tau$.

R_{its}

U_{it} A continuous nonnegative variable used in the Modified Miller-Tucker-Zemlin subtour elimination constraints. 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:

RSP

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

Subject to:

$$\sum_{j\in\mathbf{N}} X_{ijt} \le 1 \qquad i\in\mathbf{N},\ t\in\mathbf{T} \tag{2}$$

$$\sum_{j\in\mathbf{N}} X_{jit} \le 1 \qquad i\in\mathbf{N},\ t\in\mathbf{T} \tag{3}$$

$$\sum_{i\in\mathbf{V}} Z_{it} \le \alpha \qquad t\in\mathbf{T} \tag{4}$$

$$\sum_{i\in\mathbf{V}} Z_{it} \ge 1 \qquad t\in\mathbf{T} \tag{5}$$

$$\sum_{i\in\mathbf{V}} \sigma_i Z_{it} + \sum_{i\in\mathbf{N}}\sum_{j\in\mathbf{N}} d_{ij}X_{ijt} \le T_{\max} \qquad t\in\mathbf{T} \tag{6}$$

$$FM_{i1} = Z_{i1} \qquad i\in\mathbf{V} \tag{7}$$

$$FM_{it} \le Z_{it} \qquad i\in\mathbf{V},\ t\in\mathbf{T}\setminus\{1\} \tag{8}$$

$$FM_{it} \le 1 - Z_{iu} \qquad i\in\mathbf{V},\ t\in\mathbf{T}\setminus\{1\},\ 1\le u<t \tag{9}$$

$$\sum_{j\in\mathbf{N}} X_{ijt} - \sum_{j\in\mathbf{N}} X_{jit} = L_{it} - E_{it} \qquad i\in\mathbf{N},\ t\in\mathbf{T} \qquad (10)$$

$$L_{it} + E_{it} \le 1 \qquad i\in\mathbf{N},\ t\in\mathbf{T} \qquad (11)$$

$$\sum_{i\in\mathbf{N}} (L_{it} + E_{it}) \le 2 \qquad t\in\mathbf{T} \qquad (12)$$

$$S_{i(t-1)} \le S_{it} + \sum_{j\in\mathbf{N}} \frac{L_{jt} + E_{jt}}{2} \qquad i\in\mathbf{N},\ t\in\mathbf{T}\setminus\{1\} \qquad (13)$$

$$\sum_{j\in\mathbf{N}} \frac{L_{jt} + E_{jt}}{2} + S_{i(t-1)} \ge S_{it} \qquad i\in\mathbf{N},\ t\in\mathbf{T}\setminus\{1\} \qquad (14)$$

$$S_{i(t-1)} \le L_{it} + S_{it} \qquad i\in\mathbf{V},\ t\in\mathbf{T}\setminus\{1\} \qquad (15)$$

$$S_{0t} = 0 \qquad t\in\mathbf{T} \qquad (16)$$

$$S_{it} \ge X_{i0t} \qquad i\in\mathbf{V},\ t\in\mathbf{T} \qquad (17)$$

$$S_{i(t-1)} \ge X_{i0t} \qquad i\in\mathbf{V},\ t\in\mathbf{T}\setminus\{1\} \qquad (18)$$

$$X_{i0t} = X_{0it} \qquad i\in\mathbf{V},\ t\in\mathbf{T} \qquad (19)$$

$$E_{it} \le S_{it} \qquad i\in\mathbf{V},\ t\in\mathbf{T} \qquad (20)$$

$$S_{it} \le \sum_{j\in\mathbf{N}} X_{ij(t+1)} \qquad i\in\mathbf{V},\ 1\le t<\tau \qquad (21)$$

$$\sum_{i\in\mathbf{V}} S_{it} = 1 \qquad t\in\mathbf{T} \qquad (22)$$

$$\sum_{k=t}^{t+\kappa} S_{1k} \ge 1 \qquad 1\le t\le \tau-\kappa \qquad (23)$$

$$Z_{it} \le \sum_{j\in\mathbf{N}} X_{ijt} + E_{it} \qquad i\in\mathbf{V},\ t\in\mathbf{T} \qquad (24)$$

$$Z_{it} \le \sum_{j\in\mathbf{N}} X_{jit} + L_{it} \qquad i\in\mathbf{V},\ t\in\mathbf{T} \qquad (25)$$

$$(\alpha+1)S_{j(t-1)} + (\alpha+1)(1-X_{ijt}) + U_{jt} \ge U_{it} + 1 \qquad i,j\ (i\ne j)\in\mathbf{N},\ t\in\mathbf{T}\setminus\{1\} \qquad (26)$$

$$U_{it} \le \alpha + 1 \qquad i\in\mathbf{N},\ t\in\mathbf{T} \qquad (27)$$

$$U_{it} \le \sum_{j\in\mathbf{N}} \sum_{k\in\mathbf{N}} X_{jkt} + 1 \qquad i\in\mathbf{N},\ t\in\mathbf{T} \qquad (28)$$

$$U_{it} \ge S_{i(t-1)} \qquad i\in\mathbf{N},\ t\in\mathbf{T}\setminus\{1\} \qquad (29)$$

$$(\alpha+1)(1-S_{i(t-1)}) + 1 \ge U_{it} \qquad i\in\mathbf{N},\ t\in\mathbf{T}\setminus\{1\} \qquad (30)$$

$$U_{it} \geq \sum_{j \in \mathbf{N}} X_{ijt} \qquad i \in \mathbf{N},\ t \in \mathbf{T} \tag{31}$$

$$U_{it} \geq S_{it} + \sum_{j \in \mathbf{N}} X_{ijt} \qquad i \in \mathbf{N},\ t \in \mathbf{T} \tag{32}$$

$$U_{it} \leq (\alpha + 1) \sum_{j \in \mathbf{N}} X_{ijt} + (\alpha + 1) \sum_{j \in \mathbf{N}} X_{jit} \qquad i \in \mathbf{N},\ t \in \mathbf{T} \tag{33}$$

$$R_{its} \leq Z_{it} \qquad i \in \mathbf{N},\ 2 \leq t \leq \tau,\ 1 \leq s < t \tag{34}$$

$$R_{its} \leq Z_{i(t-s)} \qquad i \in \mathbf{N},\ 2 \leq t \leq \tau,\ 1 \leq s < t \tag{35}$$

$$\sum_{k=t-s+1}^{t-1} Z_{ik} \leq s(1 - R_{its}) \qquad i \in \mathbf{N},\ 3 \leq t < \tau,\ 2 \leq s < t \tag{36}$$

$$R_{its} = 0 \qquad i \in \mathbf{N},\ t \in \mathbf{T},\ t \leq s \leq \tau \tag{37}$$

$$R_{ius} \leq 1 - FM_{it} \qquad i \in \mathbf{V},\ t \in \mathbf{T} \setminus \{1\},\ t < u < \tau,\ u - t < s < u \tag{38}$$

$$R_{its} \geq Z_{i(t-s)} + Z_{it} - \sum_{k=t-s+1}^{t-1} Z_k - 1 \qquad i \in \mathbf{V},\ 3 \leq t \leq \tau,\ 2 \leq s < t \tag{39}$$

$$X_{ijt},\ L_{it},\ E_{it},\ S_{it},\ Z_{it},\ FM_{it},\ R_{its} \in \{0,1\} \text{ and } U_{it} \geq 0 \tag{40}$$

The MILP model in (1)-(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 $\frac{1}{6}n\tau^3 + 2n\tau^2 + \frac{3}{2}n^2\tau + \frac{62}{3}n\tau + \frac{45}{2}\tau + \frac{1}{2}\tau^2 - n^2 - 3n - \kappa - 4$ constraints. Note that the activity indicator variables Z_{it}, FM_{it} and R_{its} are defined for $i \in \mathbf{V}$ since the fictitious city cannot host an activity. The objective function (1) seeks to maximize the difference between the collected rewards and the incurred routing costs. Note that rewards are depreciated linearly in time as we get closer to the end of the campaign rather than the other way around. We consulted with the political party for which we proposed an application of the RSP; their suggestion was to adopt a depreciation scheme in which earlier meetings of the party leader earn higher rewards than belated meetings towards the end of the campaign period.

The set of constraints (2)-(6) and (40) are adopted from the TSP literature (Öncan et al., 2009). The set of inequalities (2) and (3) are typical selective TSP equations limiting the numbers of incoming and outgoing arcs to one for each node in $\mathbf{N}$. Constraints (4) impose an explicit upper bound α on the total number of daily activities $(\alpha \leq n)$. Constraints (5) force the campaigner to perform at least one activity in each day t while constraints (6) ensure the maximum daily tour duration is not violated. An alternative formulation for constraints (6) is provided in the next subsection. Binary integrality and nonnegativity constraints on the respective decision variables are defined in (40).

Equality constraints (7) ensure that the first activity indicator variable and the activity indicator variable for day 1 must be equal. Constraints (8) set an upper bound for FM_{it}, thereby establish the coupling between FM and Z. Due to the maximization of 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 (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 (10) couple the binary decision variables X, L and E. Constraints (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 (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 (13) and (14) force the campaigner to stay overnight in the source i on day t if there is a closed tour that day. Constraints (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 (16) set the variables s_{0t} to zero since the campaigner can never stay overnight in the fictitious city '0'. Constraints (17)-(18) are added to prevent the inclusion of the fictitious city in Type 1 and Type 3 tours. Along with constraints (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 $(X_{i0t} = X_{0it} = 1)$.

The set of constraints (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. Constraints (21) guarantee that if the campaigner sleeps in city i on day t, he must depart from there the next day. Equalities (22) ensure that the campaigner stays overnight in one city only. Constraints (23) prevent the campaigner from being away from the campaign base (city '1') for more than κ consecutive days. The set of inequalities (24) and (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 city i, there is no activity in city i either.

Constraints (26)-(33) are *Modified Miller-Tucker-Zemlin* inequalities (M-MTZ) for subtour elimination adapted to RSP. The disaggregated constraints (34)-(35) provide the logical coupling between the binary variables R_{its} and Z_{it}. Inequalities (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 (37) signify the domain restriction on the definition of the variables R_{its}. Constraints (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 (39) may seem unnecessary since their coefficients in the

objective function to be maximized are all strictly positive. However, (39) serve as valid inequalities and contribute affirmatively to the solution speed of the model. Other compact formulations for the subtour elimination constraints can be found in Maffioli and Sciomachen (1997) and Bianchessi et al. (2018).

3.3 An alternative formulation for satisfying the maximum tour duration

An alternative way of satisfying the maximum tour duration is to introduce the continuous decision variable A_{it}. Such a formulation is especially useful for problems with time windows. It can also be beneficial if the schedule of coaches or domestic flights is incorporated into the model, or if the time slots of the day are considered in the reward function. However, our empirical testing of both formulations found that constraints (6) provide more favorable results than constraints (41)-(46). See Section 5.1.

$$A_{it} \le T_{\max}(1 - S_{i(t-1)}) \qquad i \in \mathbf{V},\ t \in \mathbf{T} \setminus \{1\} \qquad (41)$$

$$A_{jt} \ge A_{it} + \sigma_i Z_{it} + d_{ij} - T_{\max}(1 - X_{ijt} + S_{j(t-1)}) \qquad i, j \in \mathbf{V},\ t \in \mathbf{T} \setminus \{1\} \qquad (42)$$

$$A_{jt} \le A_{it} + \sigma_i Z_{it} + d_{ij} + T_{\max}(1 - X_{ijt} + S_{j(t-1)}) \qquad i, j \in \mathbf{N},\ t \in \mathbf{T} \setminus \{1\} \qquad (43)$$

$$0 \le A_{it} \le T_{\max} - \sigma_i Z_{it} \qquad i \in \mathbf{N},\ t \in \mathbf{T} \qquad (44)$$

$$A_{it} + \sigma_i Z_{it} + d_{ij} \le T_{\max} + M(2 - S_{j(t-1)} - S_{jt}) \qquad i \in \mathbf{V},\ t \in \mathbf{T} \setminus \{1\} \qquad (45)$$

$$A_{it} \le T_{\max}\Big(\sum_{j \in \mathbf{V}} X_{jit} + \sum_{j \in \mathbf{V}} X_{ijt}\Big) \qquad i \in \mathbf{V},\ t \in \mathbf{T} \qquad (46)$$

The set of constraints (41) ensure that the arrival time for city i on day t will be zero if the salesperson 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 (42) and (43). Inequalities (44) impose the lower and upper bounds of A_{it}. Constraints (45) represent the general maximum tour duration definition. These are binding for open tours. The set of constraints (46) are also binding for closed tours.

3.4 Valid inequalities for tightening up the formulation

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

$$\sum_{t \in \mathbf{T}} FM_{it} \le 1 \qquad i \in \mathbf{V} \qquad (47)$$

$$L_{it} \le 2 - S_{i(t-1)} - S_{it} \qquad i \in \mathbf{N},\ t \in \mathbf{T} \qquad (48)$$

$$E_{it} \le 2 - S_{i(t-1)} - S_{it} \qquad i \in \mathbf{N},\ t \in \mathbf{T} \qquad (49)$$

$$X_{ijt} + X_{jit} \le 1 + S_{it} + S_{jt} \qquad i, j\ (i < j) \in \mathbf{N},\ t \in \mathbf{T} \qquad (50)$$

Valid inequalities (47) ensure that the first activity for each city can occur at most once during the campaign. Valid inequalities (48) and (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 (50) guarantee that if cities i and j are not terminal cities on day t, then there should not be a cyclic tour between them. These constraints are empirically demonstrated to be effective. We provide the computational evidence in Section 5.

3.5 Operational assumptions about the meetings during an election campaign

In this section we introduce several operational assumptions pertinent to the meetings held by the politician. We remark that the words *activity* (of a campaigner) and *meeting* (of a politician) are used interchangeably. We propose the associated constraint equations (51)-(53) below, which have not been included in the original RSP model in (1)-(40).

$$\sum_{i \in \mathbf{B}} Z_{it} \leq 1 \qquad t \in \mathbf{T} \tag{51}$$

$$R_{it1} = 0 \qquad i \in \mathbf{V},\ t \in \mathbf{T} \setminus \{1\} \tag{52}$$

$$\sum_{t \in \mathbf{T}} Z_{it} \leq \beta_i \qquad i \in \mathbf{V} \tag{53}$$

The first supplementary assumption formulated in (51) is that a daily tour cannot involve more than one big city. The set of big cities is indicated by **B**. The second assumption gives rise to constraints (52) 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. This maximum number is denoted by the parameter β_i in (53).

4. The proposed FDOR matheuristic

In this section, a two-phase matheuristic approach is described. We first present the underlying motivation of this approach by introducing a partial variable fixing mechanism. Next, we explain the main steps of the developed solution method.

4.1 Variable Fixing

The idea of using a matheuristic approach to tackle large-size instances is motivated by observing the results of a partial variable fixing. In those instances which we are able to solve to proven optimality using the MILP solver, we convert the binary decision variables S_{it}, L_{it} and E_{it} to input parameters. Their values are set equal to the optimal values of the respective variables. In the remaining instances which we are unable to solve to proven optimality, we perform the same conversion by replacing S_{it}, L_{it} and E_{it} with their best feasible values obtained by the MILP solver. This way the formulation has a substantially smaller number of decision variables and constraints. The best objective values of the original RSP model and the partial variable fixing approach are contrasted in Table 3 in Section 0. We observe that the model with some variables made fixed can be solved to optimality in shorter solution times (CPU times). We deduce

that the difficulty of the RSP is much more attributed to the scheduling and accommodation part rather than to the routing part. Therefore, we decide to design a two-phase method where the scheduling and accommodation part of the problem is segregated from the comprehensive formulation in (1)-(40). In such an approach, the mathematical formulation will take care of the routing part only.

4.2 Finding Daily Optimal Routes method (FDOR)

Motivated by the above observation, we propose a two-phase matheuristic to deal with large-size instances, which we call Finding Daily Optimal Routes method (FDOR). It basically consists of two phases; city selection and route generation. The route generation phase utilizes an integer program to build the optimal route among selected cities. FDOR is an integer programming based heuristic which decomposes the original MILP formulation into as many subproblems as the number of days, where the subproblem associated with a given day depends on how frequently the campaign base is to be visited throughout the campaign duration. For those days on which the campaigner needs to visit the campaign base, FDOR model 1 (FDORM1) is solved; for the other days we solve FDOR model 2 (FDORM2). Both models are solved with respect to the particular subset of cities selected in the first phase of the matheuristic. The high-level description of FDOR is provided in Algorithm 1.

Algorithm 1 The high level description of FDOR

Do the following for each day t 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 with the highest rewards.
- Pseudo-Random City Selection (PCS): Select λ cities pseudo-randomly according to a roulette wheel mechanism.

Phase 2:

(a) Solve a TSPP for the selected cities of Phase 1 using either:

- Model FDORM1: If the campaigner should stay overnight in the campaign base on day t.
- Model FDORM2: Otherwise.

(b) Update the rewards.

Once the candidate cities are selected for a given day t, our matheuristic FDOR optimally solves a Prize-Collecting Traveling Salesman Problem (PCTSP) using either the model FDORM1 or FDORM2. The detailed pseudo code of FDOR is presented in Algorithm 2. The new notation used in Algorithm 2 is defined 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 activities held in city $i \in \mathbf{N}$ during the campaign.

s_i: Number of days since the last activity 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

Input: An RSP instance.

Output: A good feasible solution comprised of τ daily tours.

0: *Initialization:*

1: $S^* = \varnothing,\ B(S^*) = 0,\ \omega_i = 0.$

2: **For** $t = 1:\tau$

3: *Reward calculation:*

4: **If** $t = 1$ **Then**

5: $\Pi_t(i) \leftarrow \pi_i$ *// Every city gets its own original base reward.*

6: $\eta_t = \gamma$ *// Campaign starts from campaign base.*

7: **Else**

8: **If** $\omega_i = 0$ **Then** *// This is going to be the first-time activity in city i.*

9: $\pi_i = \pi_i \dfrac{\tau - t + 1}{\tau}$

10: **Else** *// This is a repeated activity in city i.*

11: $\pi_i = \pi_i \dfrac{\tau - t + 1}{\tau} \times \dfrac{s_i}{K\tau}$

12: $\Pi_t(i) \leftarrow \pi_i$ *// Update the rewards.*

13: **End If**

14: $\eta_t = \varphi_{t-1}$ *// Depot of day t is equal to terminal node of day $t-1$.*

15: **End If**

16: *Phase 1:*

17: $\mathbf{C}_t \leftarrow City\_Selection\_Approach(\lambda, \Pi_t)$ *// Select* λ *cities from* **N**.

18: *Phase 2:*

19: **If** $\gamma \notin \{\varphi_{t-1}, \varphi_{t-2}, ..., \varphi_{t-\kappa}\}$ **Then** *// Force the campaigner to visit* γ *as a terminal node.*

20: Solve FDORM1$(\eta_t, \mathbf{C}_t, \Pi_t, \gamma) \rightarrow B(S_t)$, S_t, φ_t, ω_i.

21: **Else**

22: Solve FDORM2$(\eta_t, \mathbf{C}_t, \Pi_t) \rightarrow B(S_t)$, S_t, φ_t, ω_i.

23: **End If**

24: Update all ω_i values according to S_t.

25: $S^* \leftarrow S^* \cup S_t$ *// Update the best solution.*

26: $B(S^*) \leftarrow B(S^*) + B(S_t)$ *// Update the best total benefit.*

27: **End For**

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

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

As discussed earlier, in FDOR we develop two mathematical formulations which are called iteratively to solve daily STSPs. The first model (FDORM1) is called when the campaigner needs to return to the campaign base. The second one (FDORM2) is called when the campaigner is free to start and finish the daily tour in any node. FDORM1 is developed to build a daily route which may start in any city including the campaign base (the city to be visited at least once every κ days), but must terminate in the campaign base at the end of that day. FDORM2, on the other hand, is developed for those days when the campaigner is not required to return to the campaign base. The feasibility of the 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 activities.

4.3 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 an activity.

U_i: A continuous nonnegative variable used in the Miller-Tucker-Zemlin Subtour Elimination Constraints (referred to as MTZ inequalities) determining the order of visit for city i.

FDORM1($\eta_t, \mathbf{C}_t, \Pi_t, \gamma$):

$$\max. \text{ Daily NET BENEFIT} = \sum_{i \in \mathbf{N}} \Pi_{it} Z_i - \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} c_{ij} X_{ij} \tag{54}$$

Subject to:

$$\sum_{i \in \mathbf{N}} \sigma_i Z_i + \sum_{i \in \mathbf{N}} \sum_{j \in \mathbf{N}} d_{ij} X_{ij} \le T_{\max} \tag{55}$$

$$\sum_{j \in \mathbf{N}} X_{ij} = \sum_{k \in \mathbf{N}} X_{ki} \qquad i \in \mathbf{N},\ i \ne \eta_t,\ \gamma \tag{56}$$

$$\sum_{j \in \mathbf{N}} X_{j,i} \le 1 \qquad i \in \mathbf{N} \tag{57}$$

$$\sum_{j \in \mathbf{N}} X_{i,j} \le 1 \qquad i \in \mathbf{N} \tag{58}$$

$$U_j \ge U_i + X_{ij} - (\alpha + 1)(1 - X_{ij}) \qquad i, j \in \mathbf{N}, j \ne \eta_t \tag{59}$$

$$U_i \le \sum_{j \in \mathbf{N}} \sum_{k \in \mathbf{N}} X_{jk} + 1 \qquad i \in \mathbf{N} \tag{60}$$

$$0 \le U_i \le (\alpha + 1) Z_i \qquad i \in \mathbf{N} \tag{61}$$

$$U_{\eta_t} = 1 \tag{62}$$

$$\sum_{j \in \mathbf{N}} X_{j\eta_t} + \sum_{j \in \mathbf{N}} X_{\gamma j} = 0 \tag{63}$$

$$\sum_{j \in \mathbf{N}} X_{\eta_t j} + \sum_{j \in \mathbf{N}} X_{j\gamma} = 2 \tag{64}$$

$$Z_i \le \sum_{j \in \mathbf{N}} X_{ji} \qquad i \in \mathbf{N},\ i \ne \eta_t \tag{65}$$

$$\sum_{i \in \mathbf{N}} Z_i \le \alpha \tag{66}$$

$$X_{ij} \in \{0,1\} \qquad i, j \in \mathbf{N} \tag{67}$$

$$Z_i \in \{0,1\} \qquad i \in \mathbf{N} \tag{68}$$

$$U_i \ge 0 \qquad i \in \mathbf{N} \tag{69}$$

In the above formulation, the objective function (54) maximizes the net benefit of a tour while deducting travel costs from collected rewards. Constraint (55) ensures that the total travel time of the tour does not exceed the maximum tour duration. The set of constraints (56) guarantee that if the campaigner enters any city, except the depot and the campaign base, he/she should leave there. Constraints (57) and (58) are typical selective TSP inequalities which impose the incoming and outgoing degree of each node. The set of constraints (59) and (60) are node-based MTZ sub-tour elimination constraints (Miller et al., 1960). The lower and upper bounds of the continuous variable U_i are specified in constraints (61) and (62). Equalities

(63) and (64) force the campaigner to leave the depot and to stay overnight in the campaign base. The inequalities (65) couple the binary decision variables $\boldsymbol{Z}$ and $\boldsymbol{X}$, and ensure that there will be no activity in non-visited cities. Such a definition results in holding an activity in every city that the campaigner enters except the depot. Constraint (66) ensures that there will be no more than α activities. Finally, binary and nonnegativity constraints on the respective decision variables are defined in (67)-(69).

Contrasted to the original formulation (1)-(40), FDORM1 is a much easier model to solve. First of all, it represents a single-period problem. As we solve FDORM1 for a particular day t, we already know the activity schedule of the previous days. Therefore, there is no need to include the binary variables FM and R in FDORM1 for the purpose of capturing first-time or repeated activities. Also the starting node of the current day t is known due to the fact that the terminal node of day $t-1$ is known. Thus, there is no need to keep the binary variables L, E and S of the RSP model to track the terminal node of each previous day. The exclusion of these variables leads to a simple yet effective model. The formulation of FDORM2 is similar to FDORM1 except that constraints (63) and (64) are replaced by:

$$\sum_{j\in\mathbf{N}} X_{\eta_t j} = 1 \tag{70}$$

$$\sum_{j\in\mathbf{N}} X_{ij} \le \sum_{k\in\mathbf{N}} X_{ki} \qquad i\in\mathbf{N},\ i\neq\eta_t \tag{71}$$

$$\sum_{j\in\mathbf{N}} X_{ij} + \sum_{j\in\mathbf{N}} X_{ji} - 1 \le Z_i \qquad i\in\mathbf{N} \tag{72}$$

Constraint (70) ensures that the campaigner leaves the depot. Constraints (71) allow the model to generate either an open tour or a closed tour. Finally, constraints (72) couple the binary variables $\boldsymbol{Z}$ and $\boldsymbol{X}$. 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. Compared to the original model in (1)-(40), both FDORM1 and FDORM2 comprise substantially fewer variables and constraints. This massive reduction in size is achieved by decomposing the original problem into as many subproblems as the number of days in the campaign duration.

5. Computational Results

Our computational tests were performed on a Dell Precision T7810 model work station 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 second phase of the FDOR, we employed the commercial MILP solver GUROBI 8.0.1 which is called from inside Python. Since no RSP test instance is available in the literature, we generated three sets of instances:

1: 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 cities have been selected according to their base rewards.

2: Presidential Elections II (**PE.II**): it includes 20 instances where the cities have been selected according to their distances from each other.

3: Local Election (**LE**): it consists of 3 instances with 39 districts of Istanbul.

All 45 instances were generated with real-world distances and travel times queried from Google Maps Turkey. We assume symmetric travel costs and 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 publicly available online at http://shahmanzar.ir/RSP.html.

5.1 Comparison of the original and alternative MTD formulations

We compared the maximum tour duration (MTD) constraints (6) with the alternative MTD constraints in (41)-(46) on a pilot test bed of 14 small size instances. The results are displayed in Table 1. The column header 'Opt.Gap (%)' stands for the relative optimality gap of GUROBI. It is calculated as $\text{Opt.Gap (\%)} = 100 \times \frac{UB-LB}{LB}$ where *UB* and *LB* stand for the best upper and lower bounds, respectively, that are attained by GUROBI on a given RSP instance. The results in Table 1 suggest that the MTD constraints (6) yield better CPU times and can also attain proven optimality in all 14 instances. The weakness of the alternative MTD constraints can be attributed to the continuous variables A_{it} which store arrival times for all cities. In all cases, MTD constraints (6) reduce the solution times without compromising the solution quality. This observation led us not to pursue the alternative MTD constraints any further.

Table 1. Comparison of two MTD formulations

Small Instances	MTD constraints (41)-(46) using A_{it}		MTD constraints (6)	
	Opt.Gap (%)	CPU (s)[a]	Opt.Gap (%)	CPU (s)[a]
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

[a] Measured on a notebook with Intel Core i5-4310U processor.

5.2 Effect of the added valid inequalities on solution quality

We tested the effect of the added valid inequalities (VIs) implied by constraints (47)-(50) on a test bed of 12 instances from the set **PE.I**. Table 2 below displays the GUROBI solutions and corresponding CPU times (in seconds) for the models without valid inequalities, with all VIs but (50), and with all VIs. The optimal solutions are shown in **bold**; the remaining figures show the objective value of the best feasible solutions (BFSs). The better average objective values and gaps are printed in *italic* in the bottom row.

Table 2. Comparison of the models with and without valid inequalities.

	All VIs OFF			*All VIs ON except* (50)			*All VIs ON*		
PE.I Instances	BFS	Opt.Gap (%)	CPU (s)	BFS	Opt.Gap (%)	CPU (s)	BFS	Opt.Gap (%)	CPU (s)
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 2, the original RSP formulation is more compact when all valid inequalities are incorporated. The average objective value improves approximately by 2% from 18759 to 18961; the average optimality gap reduces from 2.9% to a 1.0%. The largest gap is below 7%; this is a massive drop from the previous value of 18.8%. Convinced by these test results, we opted to include in our experiments all VIs proposed in Section 3.4.

5.3 The results of partial variable fixing

Table 3 represents the comparison of the original formulation and the partial variable fixing approach. **Boldface** figures point to proven optimality attained by the commercial solver GUROBI in either case. The first column in each table section hosts the best feasible solution reported by GUROBI.

Table 3. Comparison of the results of original formulation and variable fixing.

	RSP			RSP with partial variable fixing		
Instance	BFS	Opt.Gap (%)	CPU (s)	BFS	Opt.Gap (%)	CPU (s)
15C7D	**17240**	0.0	551.3	**17240**	0.0	0.05
15C10D	**18759**	0.0	30458.5	**18759**	0.0	0.08
21C7D	**19138**	0.0	6705.3	**19138**	0.0	0.80
21C10D	21904	6.9	86400.0	21904	0.0	1.13
30C7D	**29427**	0.0	20670.3	**29427**	0.0	5.35
30C10D	35013	6.0	86400.0	35013	0.0	3.54
40C7D	30086	4.1	86400.0	30195	0.0	25.02
40C10D	36409	12.6	86400.0	36409	0.0	211.32
51C7D	41087	9.9	86400.0	41182	0.0	95.86
51C10D	45667	22.4	86400.0	45810	0.0	424.67

5.4 Comparison of FDOR-DCS, FDOR-GCS and FDOR-PCS

Table 4 below presents the comparison of different city selection approaches for 16 of 22 **PE.I** instances. We did not test the smallest six instances due to their excessively small solution times. The objective values of the best feasible GUROBI solutions and the FDOR solutions (BFS and Obj.Val. respectively) are provided along with the corresponding CPU times for all three approaches. FDOR-DCS outperforms the other two 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 in shorter times. In general, FDOR-DCS spends more time in Phase 2 by considering all n cities for selection. On the other hand, FDOR-GCS and FDOR-PCS work on a subset of λ cities which are selected in a greedy or pseudo-random way to ensure solution diversity.

FDOR-DCS approach dominates with an average total net benefit value of 37162 as can be seen in Table 4. It can find the optimal solution or the BFS in 11 of the 16 instances. FDOR-GCS can do so only in five instances, while FDOR-PCS cannot find any. Compared to the commercial solver GUROBI which returns a solution in the first 12 instances, the average gap between FDOR-DCS and GUROBI solutions is about 4.40%. This relative deviation of FDOR-GCS and FDOR-PCS solutions is 4.98% and 5.20%, respectively. Both FDOR-GCS and FDOR-PCS city selection approaches have been tested thoroughly using five different values of λ ranging from 10 to 21 cities and performing 5, 10, 15 and 20 replications in FDOR-PCS. Yet, FDOR-DCS happens to outperform the other two approaches in solution quality. The detailed results are not provided here, but can be collected from the authors. Given the more promising performance of FDOR-DCS, we focus just on this particular approach in the rest of our experiments. Figure 2 recapitulates the information shown in Table 4 to depict the solution quality comparison of the three city selection approaches in FDOR.

Table 4. Comparison of FDOR-DCS, FDOR-GCS, and FDOR-PCS

	GUROBI		FDOR-DCS		FDOR-GCS		FDOR-PCS	
Instances	BFS	Opt.Gap (%)	Obj.Val.	CPU (s)	Obj.Val.	CPU (s)	Obj.Val.	CPU(s)
12C5D	**14575**	0.0	12906	0.30	12906	0.39	12906	1.47
15C7D	**17240**	0.0	16132	0.52	16132	0.44	16103	1.72
15C10D	**18759**	0.0	17356	0.70	17356	0.42	17234	2.34
21C7D	**19138**	0.0	17325	0.99	17324	0.38	17324	3.86
21C10D	21904	6.8	20673	1.22	20673	0.49	20673	5.07
30C7D	**29427**	0.0	27474	1.72	27963	0.38	27963	12.24
30C10D	35013	5.9	32213	2.26	32427	0.49	32533	23.51
40C7D	30086	4.0	28821	3.74	28114	1.08	27927	14.18
40C10D	36409	12.6	34672	4.97	34278	1.49	33233	14.67
51C7D	41087	9.9	36942	8.40	36446	0.95	36218	22.47
51C10D	45667	22.3	43212	11.38	42406	1.89	42165	15.59
51C30D	47279	186.7	59890	14.56	58587	3.49	59745	105.71
70C15D	–	–	46818	16.64	45752	2.36	43116	54.85
70C40D	–	–	58408	22.26	54235	7.60	54809	103.38
93C30D	–	–	68174	26.61	63085	8.03	58493	108.73
93C40D	–	–	73574	27.12	68307	9.94	62090	69.21
		Average	***37162***	**8.96**	35999	*2.49*	35158	34.94

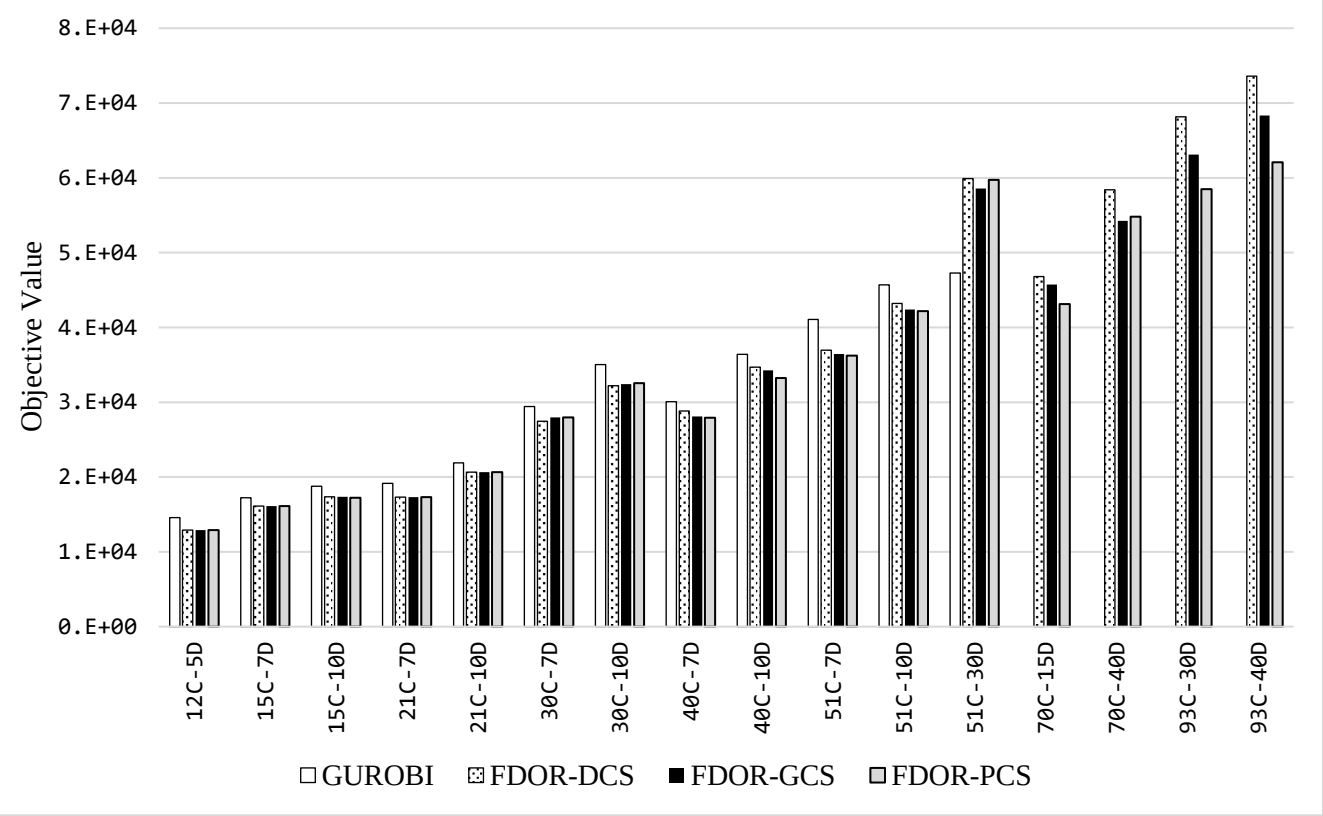

Figure 2. Comparison of FDOR-DCS, FDOR-GCS, and FDOR-PCS

Table 5 presents for a particular test instance, namely `30C7D`, the daily routes generated by FDOR-DCS and the routes of the optimal solution. We specify "Holding a Meeting" by (M).

Table 5. 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) → Aydin (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) → Aydin (M)	(Sleep in Aydin)
	Day 4: Wakeup in Aydin → 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 FDOR-DCS, we compared in Figure 3 the daily net benefit (collected daily rewards minus daily travel costs) of the optimal solution with that of the FDOR-DCS solution in the same instance `30C7D`. For day 1, FDOR-DCS generates the same route as the optimal GUROBI solution and for days 2 and 7, it was able to obtain higher daily net benefits.

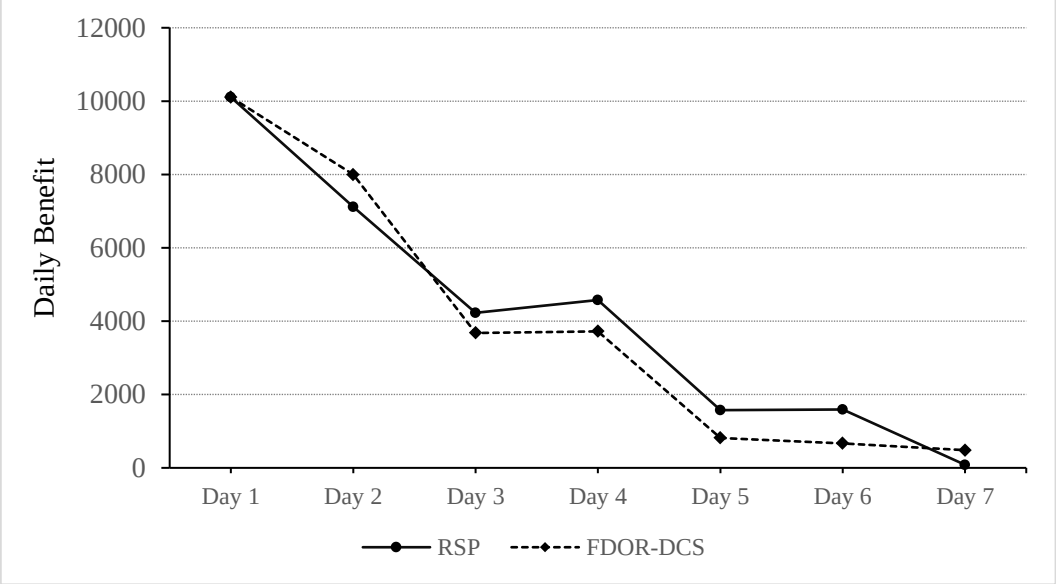

Figure 3. The comparison of the daily net benefit of FDOR and original formulation

5.5 Comparison of GUROBI and FDOR-DCS for all three sets of instances

Since FDOR-DCS was found to return better solutions than FDOR-GCS and FDOR-PCS, we decided to benchmark it against the commerical solver GUROBI in all 45 instances. Table 6 below displays the objective values of all instances for both GUROBI and FDOR-DCS along with CPU times in seconds. The column header t^{BFS} indicates the time elapsed before GUROBI has attained the tightest lower bound on the true optimal objective value. That lower bound corresponds to the BFS of the problem. Finally, the column header 'FDOR.Gap (%)' represents the gap of the FDOR-DCS solution with respect to the tightest lower bound, i.e. the BFS found by GUROBI. It is calculated as $\text{FDOR.Gap (\%)} = 100 \times \frac{\text{BFS} - \text{Obj.Val.}}{\text{BFS}}$. In all computational results, the **boldface** figures signify proven optimality achieved either by GUROBI or by FDOR-DCS.

The results of Table 6 suggest that FDOR-DCS is able to generate good feasible solutions in remarkably shorter times than GUROBI. The average gap between the optimal solution or the BFS of GRUOBI and the net benefit found by FDOR-DCS is 0.83% still in favor of GUROBI. However, the average CPU time decreases from 58357.0 seconds to 12.4 seconds. In other words, FDOR-DCS takes a tiny fraction, approximately 0.021% of the commercial solver's CPU time. Moreover, when the number of cities *n* rises from 51 to 70 in `PE.I` instances, i.e. the last four instances `70C15D`, `70C40D`, `93C30D` and `93C40D`, GUROBI fails to return a feasible solution even in 24 hours, whereas FDOR-DCS finds a feasible solution in 23.1 seconds on average. The same situation is observed in three `PE.II` instances, namely `70C30D`, `80C30D` and `80C40D`.

According to Table 6 for those instances where GUROBI is able to return a proven optimal solution, FDOR-DCS finds quick solutions with an average optimality gap of 4.46%. For the remaining instances where the commercial solver reports a BFS, but cannot reach proven optimality in 24 hours, the average FDOR-DCS objective value is about 0.45% higher, thus better than the average BFS of the commercial solver.

The promising FDOR-DCS solutions are generated very fast; they could be utilized as an initial feasible solution (IFS) for GUROBI to tighten the final optimality gap of the associated MILP model. We elaborate on this idea in Section 5.6. Another observation from Table 6 is that when the problem size (either n or τ) increases, so does the time expenditure of FDOR-DCS at a rapid pace. To amend this, we investigated the other city selection approaches for FDOR, namely FDOR-GCS and FDOR-PCS, which were proposed in Section 4.2.

Table 6. Comparison of GUROBI with FDOR-DCS for all instances

	GUROBI				FDOR-DCS		
PE.I Instances	BFS	Opt.Gap (%)	t^{BFS} (s)	CPU (s)	Obj.Val.	CPU (s)	FDOR.Gap (%)
6C2D	**7110**	0.0	0	0.1	**7110**	0.1	0.0
6C3D	**8181**	0.0	0	0.1	**8181**	0.1	0.0
7C2D	**9629**	0.0	0	0.2	**9629**	0.1	0.0
7C4D	**11597**	0.0	0	0.4	11457	0.2	1.2
9C3D	**10939**	0.0	0	0.5	10788	0.1	1.4
9C4D	**11668**	0.0	1	1.3	11268	0.1	3.4
12C5D	**14575**	0.0	6	6.0	12906	0.3	11.5
15C7D	**17240**	0.0	462	551.3	16132	0.5	6.4
15C10D	**18759**	0.0	19972	30458.5	17356	0.7	7.5
21C7D	**19138**	0.0	2026	6705.3	17325	0.9	9.5
21C10D	21904	6.8	11582	86400.0	20673	1.2	5.6
30C7D	**29427**	0.0	20665	20670.3	27474	1.7	6.6
30C10D	35013	5.9	30040	86400.0	32213	2.2	8.0
40C7D	30086	4.0	59757	86400.0	28821	3.7	4.2
40C10D	36409	12.6	62342	86400.0	34672	4.9	4.8
51C7D	41087	9.9	85597	86400.0	36942	8.4	10.1
51C10D	45667	22.3	77316	86400.0	43212	11.3	5.4
51C30D	47279	186.7	61189	86400.0	59890	14.5	−26.7
70C15D	–	–	–	86400.0	46818	16.6	–
70C40D	–	–	–	86400.0	58408	22.2	–
93C30D	–	–	–	86400.0	68174	26.6	–
93C40D	–	–	–	86400.0	73574	27.1	–
Average [a]	23095		23941.9	36844.1	22558	2.8	3.27

	GUROBI				FDOR-DCS		
PE.II Instances	BFS	Opt.Gap (%)	t^{BFS} (s)	CPU (s)	Obj.Val.	CPU (s)	FDOR.Gap (%)
20C5D	**25118**	0.0	44	239.2	24196	0.6	3.7
20C7D	**27523**	0.0	454	1995.9	25419	0.6	7.6
30C5D	**16635**	0.0	161	709.9	16052	1.5	3.5
30C7D	**18855**	0.0	13163	28216.8	17997	1.8	4.6
30C10D	21251	5.9	17722	86400.0	19577	2.0	7.9
40C7D	32811	20.1	76679	86400.0	31748	3.0	3.2
40C10D	37851	3.8	51297	86400.0	34267	3.6	9.5
50C7D	32829	1.6	4945	86400.0	33101	6.9	−0.8
50C10D	38098	11.8	45006	86400.0	37389	8.5	1.9
50C15D	44098	35.6	70662	86400.0	41687	11.0	5.5
60C7D	40480	2.5	36955	86400.0	38105	13.8	5.9
60C10D	48270	7.0	82709	86400.0	45446	18.8	5.8
60C20D	50559	80.1	64244	86400.0	62869	22.8	−24.3
70C10D	42474	13.9	82434	86400.0	40201	26.1	5.4
70C20D	43705	112.3	83589	86400.0	51055	34.9	−16.8
70C30D	–	–	–	86400.0	57065	36.2	–
80C10D	40808	22.2	52003	86400.0	38423	38.6	5.8
80C20D	50777	75.1	74448	86400.0	53270	41.9	−4.9
80C30D	–	–	–	86400.0	57285	50.0	–
80C40D	–	–	–	86400.0	62576	48.7	–
Average [a]	36008		44500.9	67903.6	35930	13.9	1.39
LE Instances							
39C7D	22361	4.7	85841	86400.0	22164	11.7	0.9
39C10D	26774	18.5	63714	86400.0	27191	13.5	−1.6
39C14D	30214	57.5	76094	86400.0	31757	15.9	−5.1
Average	26450		75216.3	86400.0	27037	13.7	−1.9

[a] The average objective values, CPU times and gaps have been calculated for the pool of only those instances in which GRUOBI is able to find a BFS.

5.6 Speeding up GUROBI using the results of FDOR

The commercial solvers such as GUROBI and CPLEX assume the initial values of all decision variables to be 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 preprocessing step. When we tackle a MILP problem, it is possible to assist the commercial solver by setting the decision variables to certain initial values, which is known as a warm start. A warm start may be performed by starting the solver at a nontrivial feasible solution.

In our case, we can generate a high-quality feasible solution through FDOR which executes in a small amount of CPU time. So, we can feed it as an initial solution into the commercial solver for the purpose of a warm start. This high-quality initial solution may help the solver, though not always, find the optimal solution faster than before. Likewise, it could contribute in large size instances to the detection of a feasible solution which the solver fails to find even in 24 hours. To this end, GUROBI is called after the initial values of the binary decision variables X, Z, R, FM, S, L and E in the RSP model have been set to the respective values retrieved from the FDOR solution of the same problem. The new GUROBI results are presented in Table 7. The solution quality increased in 38 of 45 instances. It is worth noting that GUROBI was previously unable to reach proven optimality for `PE.I 40C7D` and `PE.II 50C7D`; but by feeding the FDOR solution into GUROBI as an initial solution it was possible to do so in less than 24 hours. Moreover, in `PE.I` instances `70C15D`, `70C40D`, `93C30D` and `93C40D` as well as in `PE.II` instances `70C30D`, `80C30D` and `80C40D`, GUROBI was previously unable to return a feasible solution at all. Now after a warm start with the FDOR solution, it can at least arrive at a feasible solution, albeit with an average Opt.Gap still above 100%. For the remaining 38 instances, the average BFS improves (i.e. increases) by approximately 3.79%. Finally, the average Opt.Gap again in those 38 instances drops from 18.49% to 14.64%.

5.7 Comparison of the FDOR method with Party's actual meeting plan

In order to testify the solution quality of the proposed FDOR method further, we scrutinize the real-life instance `70C40D`. We were able to obtain the political party's realized meetings prior to the general election of June 2015 in Turkey. The party started its campaign 40 days before the election day; the party leader held meetings in a total of 70 cities and towns with repeated meetings in several of them. In the light of these meetings, we created our large size instance `70C40D`. We compare the total benefit value implied by the party's actual meeting plan to the BFS found by FDOR. 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. (i) 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. (ii) The second one was forcing the politician to end the campaign at the campaign base. We relaxed this constraint since the actual campaign of the party back in June 2015 had not been completed in Ankara. (iii) The last one was forcing the politician to return to the campaign base frequently. We lifted this constraint as well.

Table 7. The results of setting initial values of RSP to optimal values of FDOR

	RSP (original)			RSP with FDOR solution as the initial solution		
PE.I Instances	BFS	Opt.Gap (%)	CPU (s)	BFS	Opt.Gap (%)	CPU (s)
6C2D	**7110**	0.0	0.1	**7110**	0.0	0.3
6C3D	**8181**	0.0	0.1	**8181**	0.0	0.5
7C2D	**9629**	0.0	0.2	**9629**	0.0	0.5
7C4D	**11597**	0.0	0.4	**11597**	0.0	0.6
9C3D	**10939**	0.0	0.5	**10939**	0.0	1.3
9C4D	**11668**	0.0	1.3	**11668**	0.0	1.5
12C5D	**14575**	0.0	6.0	**14575**	0.0	6.8
15C7D	**17240**	0.0	551.3	**17240**	0.0	572.5
15C10D	**18759**	0.0	30458.5	**18759**	0.0	20630.7
21C7D	**19138**	0.0	6705.3	**19138**	0.0	5311.0
21C10D	21904	6.8	86400.0	21684	6.7	86400.0
30C7D	**29427**	0.0	20670.3	**29427**	0.0	14071.7
30C10D	35013	5.9	86400.0	35197	5.7	86400.0
40C7D	30086	4.0	86400.0	**30122**	0.0	77160.8
40C10D	36409	12.6	86400.0	34763	17.8	86400.0
51C7D	41087	9.9	86400.0	41442	7.8	86400.0
51C10D	45667	22.3	86400.0	46971	19.5	86400.0
51C30D	47279	186.7	86400.0	59895	100.0	86400.0
70C15D	–	–	86400.0	46818	66.1	86400.0
70C40D	–	–	86400.0	58408	170.1	86400.0
93C30D	–	–	86400.0	68174	96.4	86400.0
93C40D	–	–	86400.0	73595	108.1	86400.0
Average [a]	23094	13.8		23797	8.8	

	RSP (original)			RSP with FDOR solution as the initial solution		
PE.II Instances	BFS	Opt.Gap (%)	CPU (s)	BFS	Opt.Gap (%)	CPU (s)
20C5D	**25118**	0.0	239.2	**25118**	0.0	255.0
20C7D	**27523**	0.0	1995.9	**27523**	0.0	2118.4
30C5D	**16635**	0.0	709.9	**16635**	0.0	885.1
30C7D	**18855**	0.0	28216.8	**18855**	0.0	27959.7
30C10D	21251	5.9	86400.0	22464	3.9	86400.0
40C7D	32811	2.0	86400.0	32788	2.2	86400.0
40C10D	37851	3.8	86400.0	37641	4.4	86400.0
50C7D	32829	1.6	86400.0	**33352**	0.0	82875.1
50C10D	38098	11.8	86400.0	42202	10.5	86400.0
50C15D	44098	35.6	86400.0	41922	48.3	86400.0
60C7D	40480	2.5	86400.0	39793	4.4	86400.0
60C10D	48270	7.0	86400.0	48354	6.5	86400.0
60C20D	50559	80.1	86400.0	62869	44.3	86400.0
70C10D	42474	13.9	86400.0	40240	20.2	86400.0
70C20D	43705	112.3	86400.0	51364	80.8	86400.0
70C30D	–	–	–	57092	167.6	86400.0
80C10D	40808	22.2	86400.0	41773	19.4	86400.0
80C20D	50777	75.1	86400.0	55206	88.6	86400.0
80C30D	–	–	–	58008	153.4	86400.0
80C40D	–	–	–	62576	196.1	86400.0
Average [a]	36008	22.0		37535	19.6	
LE Instances						
39C7D	22361	4.7	86400.0	22782	2.7	86400.0
39C10D	26774	18.5	86400.0	27481	15.3	86400.0
39C14D	30214	57.5	86400.0	32496	47.5	86400.0
Average	26450	26.9		27586	21.8	

[a] The average objective values and gaps have been calculated for the pool of only those instances in which GRUOBI is able to find a BFS.

Table 8 below shows the details of FDOR and the original model solutions alongside the actual plan. Accordingly, GUROBI is not able to find an optimal solution even at the end of three days. However, the best feasible solution returned 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 İstanbul, Ankara, and Mersin each, two meetings in İzmir, and one meeting in each of the remaining cities. However, the best feasible solution prescribes three meetings in each of İstanbul, Ankara, İzmir, and Mersin, two meetings in the majority of midsize cities such as Adana, Balıkesir, Bursa, Hatay, Konya, etc., and one meeting in the rest. The results highlight a massive advantage of solving the RSP for the maximization of the net benefit obtained from an election campaign that spans an extended period.

Table 8. FDOR method vs Party's actual meeting plan in the real-life instance `70C40D`

		Solution Value	Opt.Gap (%)	# of Meetings	CPU time
	RSP	LB = 46,640 UB = 117,427	60.3	75	259,200 s (3 days)
	FDOR	58,408	–	96	22.26 s
	Party's Plan	24,534	–	77	n/a
Reward-Only	RSP	LB = 68,399 UB = 106,802	56.1	65	259,200 s (3 days)
Reward-Only	FDOR	94,044	–	102	34.15 s
Reward-Only	Party's Plan	64,124	–	77	n/a

In the *Reward-Only* scenario of the problem under study, both the original formulation and FDOR outperform the party's actual plan. Considering the number of meetings as a performance measure, the solution obtained by FDOR holds 19 more meetings than the party's plan. This difference is more pronounced in the *Reward-Only* scenario where the number of meetings held in the FDOR solution surpasses that number in the party's plan by 25. Note that in both cases the objective value which we are able to obtain in less than 35 seconds using FDOR is far superior to the objective value implied by the party's plan. These performance merits hint the success of the proposed matheuristic method FDOR in scheduling and routing an election campaign.

6. Conclusions and future work

In this paper we introduce a novel logistical problem which we call the Roaming Salesman Problem (RSP). It can be classified as a multi-period version of the prize-collecting traveling salesman problem with dynamic profits, repeated visits to certain customer nodes, varying depot nodes, and three types of time restricted tours. The salesperson in the problem whom we designate as the *campaigner* can stay overnight in any arbitrary city to resume his/her daily tour there the next morning. This extraordinary feature adds another level of complexity to the model of the problem. We propose an innovative MILP formulation followed by an efficient two-phase matheuristic approach consisting of two primary components: a city selection phase and a route generation phase. The proposed matheuristic, coined as *Finding Daily Optimal Routes* (FDOR), decomposes the original MILP formulation into as many subproblems as the number of days in the planning horizon. Each subproblem depends on how frequently the campaign base is to be visited throughout the campaign duration. This decomposition strategy generates the next period's route without the need to track the route of each day, which in turn reduces the computational complexity of the problem greatly. We rigorously tested three city selection approaches coupled with the associated parameter calibration experiments. Computational results suggest that FDOR provides promising solutions in remarkably short computing times.

Our work on this new problem can be extended in many directions. The decomposition scheme in our proposed FDOR method can be adapted to other hard combinatorial problems that are rather difficult to tackle otherwise. A relevant topic is the incorporation of the meetings of a rival party into the calculation of the rewards. The accommodation cost and the weekday of the meetings can also be considered in the calculation of rewards. Time windows constraints can be introduced to keep track of the time of the day and the departure schedules of coaches and planes. Moreover, alternative formulations can be investigated so as to improve the solution quality of the MILP model. For instance, connectivity constraints can be introduced and separation of violated inequalities can be implemented in a branch-and-cut algorithm. Finally, hybrid metaheuristics can be developed which would capitalize on FDOR to start at a high-quality initial solution.

Acknowledgments

The authors would like to thank the two anonymous reviewers for their useful comments and valuable suggestions which enriched the content as well as improved the exposition of the paper.

References

Angelelli, E., Gendreau, M., Mansini, R., & Vindigni, M. (2017). The traveling purchaser problem with time-dependent quantities. *Computers & Operations Research*, *82*, 15-26.

Archetti, C., Carrabs, F., & Cerulli, R. (2018). The set orienteering problem. *European Journal of Operational Research*, *267*(1), 264-272.

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.

Bianchessi, N., Mansini, R., & Speranza, M. G. (2018). A branch-and-cut algorithm for the Team Orienteering Problem. *International Transactions in Operational Research*, *25*(2), 627-635.

Butt, S. E., & Cavalier, T. M. (1994). A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, *21*(1), 101-111.

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.

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.

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.

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.

Gendreau, M., Laporte, G., & Semet, F. (1998). A branch-and-cut algorithm for the undirected selective traveling salesman problem. *Networks*, *32*(4), 263-273.

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.

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*, *203*(1), 167-186.

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., & Vigo, D. (2013). Models and algorithms for the integrated planning of bin allocation and vehicle routing in solid waste management. *Transportation Science*, *48*(1), 103-120.

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.

Labadie, N., Mansini, R., Melechovský, J., & Calvo, R. W. (2012). The team orienteering problem with time windows: An LP-based granular variable neighborhood search. *European Journal of Operational Research*, *220*(1), 15-27.

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.

Maffioli, F., & Sciomachen, A. (1997). A mixed-integer model for solving ordering problems with side constraints. *Annals of Operations Research*, *69*, 277-297.

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.

Öncan, T., Altınel, İ. K., & Laporte, G. (2009). A comparative analysis of several asymmetric traveling salesman problem formulations. *Computers & Operations Research*, *36*(3), 637-654.

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.

Salhi, S (2017). Heuristic Search: The Emerging Science of Problem Solving. *Cham, Switzerland, Springer.*

Thomadsen, T., Stidsen, T. (2003). The quadratic selective travelling salesman problem. Informatics and Mathematical Modelling Technical Report 2003-17, Technical University of Denmark.

Taş, D., Gendreau, M., Jabali, O., & Laporte, G. (2016). The traveling salesman problem with time-dependent service times. *European Journal of Operational Research*, *248*(2), 372-383.

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.