The stochastic and time-dependent traveling salesperson problem

Edison Avraham and Tal Raviv

Department of Industrial Engineering
Tel Aviv University, Ramat Aviv, Tel Aviv 6997801, Israel
Email: edisonav@post.tau.ac.il, talraviv@eng.tau.ac.il

May 2018

Abstract

In this paper, we study a single-vehicle routing problem with stochastic service times, stochastic time-dependent travel times, and soft time windows, where the travel times or the service times may be inter-dependent. The objective is to minimize the expected length of the working day plus penalties for late arrivals. The stochasticity is modeled using a set of scenarios based on historical data. This approach enables the spatial and temporal inter-dependencies in the road network to be captured. We introduce a specialized branch-and-bound algorithm and a successful tabu search heuristic for the problem. In an extensive numerical experiment based on real historical travel time data, we demonstrate the applicability of both methods to problem instances of up to 18 customers and 40 scenarios. These dimensions are a safe upper bound for instances originating from the domain of field service operation. The resulting routes are tested on realistic scenarios that were not included in the input of the problem (the training set) to demonstrate the merits of using historical data. In a comparison with solutions that ignore the time dependency and/or stochasticity of the parameters, our solutions are consistently superior. Moreover, we show that these optimal solutions are significantly better than solutions that overlook the inter-dependencies of the travel and service times.

Keywords: Travelling salesman, Transportation, Vehicle Routing, Field Service Operations

1 Introduction

Field service personnel spend most of their working day on the road or at their customers’ locations. In practice, the scheduling process for such personnel assumes deterministic service times as well as deterministic and time-independent travel times. These assumptions simplify the process and allow the schedule to be constructed using estimators of the abovementioned times. However, such a solution may be suboptimal when implemented in real-life situations. Such situations include cases where the service times of some customers are considerably longer than planned and cases where travel times are longer due to unforeseen events, such as car accidents or extreme weather conditions. Additionally, the mean travel times tend to vary as to reflect the different congestion levels during the working day.
Recent advancements in mobile computing technology have enabled the collection of real data that provide a better understanding of the stochastic and time-dependent nature of travel times. This understanding can be exploited in more accurate optimization models.

Single-vehicle routing and scheduling problems have been largely studied in the context of traveling salesperson (TSP)-type problems; That is, a single vehicle departs from the depot no earlier than a predefined time and is required to visit and serve all customers during a single working day. The travel times between all locations and the service times at the customers are assumed to be known. Each customer has a time window. Two types of time windows, soft, as well as hard, have been studied. Soft time windows allow late arrivals at customers and each late arrival incurs a penalty. The objective function, in this case, is minimizing a weighted sum of the traveling and penalty costs associated with late arrivals at customers. The existence of hard time windows may cause a problem instance to have no feasible solution.

In the above context, most studies have focused on the deterministic and time-independent version of the problem. Other studies have considered either stochasticity or time dependency but not both simultaneously. Moreover, the few studies that address these two features simultaneously generally assume independency between the various travel times. While this assumption makes the analytical calculation of arrival times computationally tractable, it may not hold in real life. In practice, congestion patterns in different parts of a road network are similar; thus, the resulted travel times are dependent.

Our approach to consider dependent travel times while maintaining the computational tractability of the arrival time calculation is to model the stochastic and time-dependent travel times using a set of predefined scenarios. In this paper, we consider the stochastic and time-dependent TSP with soft time windows (S-TD-TSP-STW). Due to the rich nature of the problem, commercial solvers cannot effectively solve even small-sized instances of the problem. Therefore, we developed a specialized branch-and-bound (B&B) algorithm to solve real-life instances to optimality. Then, we devised a Tabu search (TS) algorithm to find high-quality solutions for the problem in a short time. Such a solution approach may be useful when solving the S-TD-TSP-STW as a subroutine in an algorithm that solves a multivehicle version of the problem. Next, we conducted numerical experiments using real travel time data. These experiments demonstrated the added value of considering stochasticity and time dependency when solving the TSP.

The rest of this paper is organized as follows: Section 2 discusses representative papers that highlight important aspects of the state-of-the-art literature related to stochastic and time-dependent vehicle routing. Section 3 defines our notation and formally states the S-TD-TSP-STW problem. The solution algorithms are presented in Section 4, and the numerical experiments and results are described in Section 5. Conclusions and future work are discussed in Section 6.

2 Literature review

Studies in the domain of vehicle routing often address either stochastic or time-dependent travel times; however, few studies consider both aspects simultaneously. In this section, we review the relevant literature. Subsection 2.1 presents literature concerning time-dependent routing problems. Subsection 2.2 presents literature related to vehicle routing with stochastic travel and service times. Subsection 2.3 surveys recent literature that simultaneously addresses the time-dependent and stochastic features of routing problems.
2.1 Time-dependent vehicle routing

In time-dependent vehicle routing problems, the travel time from a given location \( i \) to a given location \( j \) depends on the time at which the vehicle departs from location \( i \). Malandraki (1989), Malandraki and Daskin (1992) and Hill and Benton (1992) present mathematical models for the time-dependent traveling salesperson problem (TD-TSP) and the time-dependent vehicle routing problem (TD-VRP), where time windows and capacity limits exist. The travel times (or travel speed) are given as step functions over time. The authors are aware that this representation may result in a violation of the FIFO property, i.e., vehicle A may depart from location \( i \) later than vehicle B but arrive at location \( j \) earlier than B. Ichoua et al. (2003) calculate the travel time based on a step function that represents the traveling speed over time and consider changes in the traveling speed as time periods are crossed during a journey between two locations. This more realistic approach satisfies the FIFO property. The resulting travel times are piecewise linear continuous functions.

Haghani and Jung (2005) address a capacitated pick-up or delivery VRP with soft time windows. The customer requests arrive as the working day progresses. The travel times are known as time-dependent functions. The routes are modified in reaction to the arrival of new data. Two solution methods are devised: a genetic algorithm and a solution of a mathematical program using a commercial solver. Jabali et al. (2009) consider a capacitated VRP with time-dependent travel times and stochastic service times, in which the duration of each route is limited. The problem is solved by a TS algorithm.

Ehmke and Mattfeld (2012) implement data mining techniques to process large amounts of floating car data to estimate time-dependent travel times in the area of Stuttgart, Germany. They incorporate these time-dependent travel times into vehicle routing models. Travel times are represented as piecewise linear functions that satisfy the FIFO property.

Verbeeck et al. (2014) study the time-dependent orienteering problem. Recall that the orienteering problem is defined by a set of customers and a depot. Each customer is associated with a reward. A solution is a route that starts and ends at the depot and visits some of the customers while satisfying a total tour length (time) constraint. The objective is to maximize the total reward of the visited nodes. An ant colony algorithm is devised to solve the time-dependent variant of the problem. Numerical experiments show that the algorithm rapidly yields good solutions.

2.2. Vehicle routing with stochastic travel and service times

Two approaches are used to address the stochasticity in the characteristics of a VRP, such as travel times and service times. Either the solution is planned off-line or an on-line policy is sought.

To the best of our knowledge, the first study on off-line models for the stochastic routing problem is Laporte et al. (1992). They present three models for the uncapacitated vehicle routing problem with stochastic service and travel times. They present a model that limits the probability of a vehicle to exceed its deadline as well as models that consider penalty costs for violating the allowed duration. Kenyon and Morton (2003) study a similar problem and present two three-indexed formulations for the problem. The first model aims to minimize the expected completion time of all routes. The second model maximizes the probability of completing all the routes at a given time. Capacity and soft time windows are considered by Li et al. (2010). They present a model that limits the probability of violating the time
windows and the probability of exceeding a given route duration. An initial solution is generated by a TS algorithm. Lei et al. (2012) solve a capacitated VRP with stochastic service times, where the duration of the route is constrained. The objective function is minimizing the sum of the travel, service, and expected penalty costs. The authors present a closed-form expression for the expected cost of a single route. Tas et al. (2012) propose a three-stage TS procedure for the VRP with stochastic travel times and soft time windows. Both the operational cost and customer inconvenience are optimized. Tas et al. (2013) present a set-partitioning formulation and a solution method based on column generation and a branch-and-price procedure for the same problem. Souyris et al. (2013) present a robust optimization method for the field service routing problem with stochastic service times and soft upper bounds on the beginning time of the service at each customer. Errico et al. (2013) are the first to present a VRP with stochastic service times and hard time windows. The authors suggest a framework that limits the probability of not violating any time window. They introduce a model based on the set-partitioning formulation for the VRP.

Studies that propose on-line models, where decisions are made as the route is executed, may solve the problem whenever stochastic parameters are realized. Another possible approach is to find an optimal policy for each possible state of the system related to the values of the realized parameters and to apply that policy as the route is executed. Delage (2010) solves a multidepot VRPTW with stochastic service times and two types of requests: repairs that have a time window and maintenance that may be performed at any time. The planning horizon is one day. The author presents two solution methods for the problem. The first method starts by finding routes that serve only repairs. Later, the maintenance requests are inserted to the routes. The solution is updated dynamically with a TS algorithm whenever the actual service time is revealed. The second method finds an initial route for each vehicle and uses a dynamic programming approach to construct a policy by which each vehicle is directed to skip customers if necessary. Binart et al. (2016) address a similar problem with stochastic travel times and devise a two-stage solution method for the problem. The planning stage begins with building routes that serve mandatory customers only. Later, they insert the optional customers between mandatory customers with the intent of skipping some optional customers if necessary. Once the route of each vehicle is planned, a dynamic program is used to plan a skipping threshold policy. Errico et al. (2016) solve the VRP with hard upper bounds on the starting time of the service. They assume that the service times follow a discrete distribution. Two recourse strategies to cope with violation of the next customer's time window are studied. The first is skipping the current customer, and the second is skipping the next customer.

2.3. Stochastic and time-dependent vehicle routing and scheduling

Few studies related to the VRP simultaneously address time dependency and stochasticity. These studies focus on the static setting; that is, the planned routes are executed without modifications. Their solution methods are often adapted from studies addressing the stochastic and time-dependent shortest path problem (TD-S-SP).

Nahum and Hadas (2009) study the stochastic time-dependent VRP (TD-S-VRP). They present a mathematical model for the problem that limits the probability of exceeding some given bound on the travel time. The objective function is minimizing the average total travel time. The authors develop a
solution heuristic executed in polynomial time that is a variant of the saving heuristic (Clarke and Wright (1964)).

Lecluyse et al. (2009) consider a stochastic time-dependent vehicle routing problem that includes the variance and expected value of the travel time in the objective function. The authors assume that the travel times between locations follow a lognormal distribution and approximate the total travel time of a route using that distribution. The problem is solved by applying a TS algorithm.

Duan et al. (2015) solve the stochastic time-dependent routing problem with hard time windows by assuming that the travel times are equal to some upper bound on the travel time between each pair of customers. That is, a feasible solution is one that satisfies all the time windows under the longest possible times. They consider two objective functions: one based on these upper bounds and the other based on the average travel time. Moreover, an ant colony optimization algorithm is presented.

Tas et al. (2014) address a variant of the TD-S-VRP with soft time windows. The objective function is minimizing a weighted sum of the expected transportation and service (lateness and earliness) costs. The authors show that under the assumption that the travel times are independent and follow a gamma distribution, the exact distributions of the arrival times can be derived when no service times are considered. Approximate distributions of the arrival times can be derived when service times are included. The authors devise two solution methods for the problem: a TS algorithm and an adaptive large neighborhood search (ALNS) algorithm.

Verbeeck et al. (2016) study a stochastic version of the time-dependent orienteering problem with hard time windows. Travel times are assumed to be independent of each other and to follow a normal distribution while service times are deterministic. The hard time windows require the authors to devise an estimation algorithm for the calculation of the arrival times. The problem is solved using an ant colony algorithm. Numerical experiments demonstrate the merit of considering time dependency and stochasticity compared to a deterministic time-independent case.

All the studies that model stochastic travel times assume (explicitly or implicitly) that these times follow independent distributions. We believe that this simplifying assumption largely misrepresents the reality of travel times in congested areas, where inter-dependence between traffic conditions in close geographical locations is substantial. Moreover, since there is a positive correlation between the travel times, the independence assumption may lead to too optimistic plans and result in many delays in the service.

In this study, the stochasticity of the travel and service times is modeled by a set of $K$ scenarios that relate to a single working day rather than by closed-form density distribution functions. While this approach may result in sacrificing some accuracy, it has two important advantages: 1. It is relatively easy to create the input for the problem based on historical data; 2. Scenarios readily capture the inherent and complicated dependency between the travel times of journeys that are spatially or temporally close.

The main contribution of this study is in introducing a model and an exact solution method for routing and scheduling a single vehicle under time-dependent stochastic travel times and stochastic service times, where these times can be inter-dependent. In addition, we present a TS heuristic for the problem and demonstrate the effectiveness of both the exact and heuristic methods using real travel time
data. Finally, we demonstrate the importance of considering the above properties rather than following the traditional approach of solving the problem based on the average times.

3 Problem definition

The S-TD-TSP-STW is stated as follows: A set of customers has to be served in a single working day. Each customer has a time window for the beginning of the service. The upper bounds of the time windows are soft while the lower bounds are hard. That is, arriving at a customer after the end of the customer's time window incurs a penalty, which is increasing in the extent of the lateness. When the vehicle arrives at a customer before the beginning of its time window, it waits until the window is opened. A single vehicle is available to serve the customers during the working day. The vehicle departs from the depot not before a given time and returns to the depot after the service of the last customer has been completed. The planning horizon is discretized into short periods, and a travel time matrix is given for each period and each scenario. The service time is given for each customer in each scenario. A solution is the sequence in which the customers should be visited to minimize the sum of the expected working day duration and a penalty function that represents the violations of the time windows for all the scenarios. Clearly, minimizing the sum over all the scenarios is equivalent to minimizing the average.

To present the problem as a mathematical program, we present the following notation.

\[
\begin{align*}
\{1, \ldots, n\} & \quad \text{Set of customers, 0 represents the depot.} \\
\{1, \ldots, K\} & \quad \text{Set of scenarios.} \\
[a_i, b_i] & \quad \text{Time window for customer } i. \\
s_{ik} & \quad \text{Service time at customer } i \text{ in the } k^{th} \text{ scenario.} \\
t_{ij,t',k} & \quad \text{Travel time between customer } i \text{ and customer } j \text{ at the period of the departure } t' \text{ for the } k^{th} \text{ scenario. We assume that the travel times in each scenario follow the FIFO property. Moreover, we assume that the triangular inequality holds in the time-dependent setting. That is, } t_{ij,t',k} \leq t_{il,t,t',k} + t_{lj,t,t',k} - t_{ilj,k} \forall i, j, l = 0, \ldots, n \quad \forall k = 1, \ldots, K \quad \forall t'. \\
G_i(x) & \quad \text{Penalty function for late arrival at customer } i. \text{ The exact shape of the penalty function is an input of this model and should be determined by the service level agreement between the provider and his customers. We assume that } G_i(x) \text{ is an increasing positive function. It is also natural to assume that } G_i(x) \text{ is convex to reflect the increasing marginal penalty for each additional time unit of delay.} \\
\end{align*}
\]

The nonlinear mixed integer programming (NL-MIP) formulation of the problem is presented below.

**Decision Variables**

\[
\begin{align*}
x_{ij} & \quad \text{Binary variable that equals } “1” \text{ if customer } j \text{ is visited immediately after customer } i \\
u_{ik} & \quad \text{Arrival time at customer } i \text{ in realization } k \\
o_{ik} & \quad \text{Lateness at customer } i \text{ in realization } k \\
T_k & \quad \text{Working day duration in realization } k \\
\end{align*}
\]
minimize \sum_{k=1}^{K} T_k + \sum_{i=1}^{n} \sum_{k=1}^{K} G_i(o_{ik}) \quad (1)

subject to

\sum_{j=0}^{n} x_{ij} = \sum_{j=0}^{n} x_{ji} \quad \forall i = 0, \ldots, n \quad (2)

\sum_{j=0}^{n} x_{ij} = 1 \quad \forall i = 0, \ldots, n \quad (3)

u_{jk} \geq (u_{ik} + s_{ik} + t_{i,j,0,0,0} + s_{ik} \cdot k) x_{ij} \quad \forall i = 0, \ldots, n; j = 1, \ldots, n, k = 1, \ldots, K \quad (4)

a_i \leq u_{ik} \quad \forall i = 1, \ldots, n, k = 1, \ldots, K \quad (5)

b_i + o_{ik} \geq u_{ik} \quad \forall i = 1, \ldots, n, k = 1, \ldots, K \quad (6)

T_k \geq (u_{ik} + s_{ik} + t_{i,0,0,0} + s_{ik} \cdot k) x_{i0} \quad \forall i = 1, \ldots, n, k = 1, \ldots, K \quad (7)

x_{ij} \in \{0,1\} \quad \forall i, j = 0, \ldots, n \quad (8)

u_{ik} \geq 0, \text{integer} \quad \forall i = 0, \ldots, n; k = 1, \ldots, K \quad (9)

o_{ik} \geq 0 \quad \forall i = 0, \ldots, n; k = 1, \ldots, K \quad (10)

The model aims to minimize the sum of the working day duration and penalty costs (1). Constraint (2) maintains vehicle flow conservation while constraint (3) ensures that all customers are visited. Constraint (4) relates the arrival times with the routing variables. Note that the decision variable $u_{ik}$ is used as an index in the time-dependent distance matrix; therefore, it must be an integer. Constraint (5) enforces hard lower bounds on the arrival times at customers that the vehicle visits while constraint (6) relates the lateness variables to the arrival times. Constraint (7) relates the working day duration with the arrival time variables. Constraints (8)-(10) define the domains of the decision variables. Note that (1)-(10) constitute a nonlinear and nonconvex mixed integer mathematical model that it is difficult to linearize, especially since we use the arrival times ($u_{ik}$) as an index in constraint (4).

4 Methodology

We began our efforts by solving the mathematical model of the problem using the constraint programming (CP) module in IBM ILOG Cplex 12.6.3. Preliminary numerical experiments with small-sized instances of nine customers showed that this method could not be used to obtain provably optimal solutions since the CP's search algorithm failed to terminate in a reasonable amount of time. Clearly, benchmarking the quality of the solutions that the CP solver obtained was also not possible, which raised the need to develop another exact solution method. Therefore, we devised a specialized B&B algorithm for the problem.

We also developed a TS heuristic solution approach that is less computationally intense than the B&B framework. We found that a TS framework was suitable to cope with the rich settings of the stated problem and yielded good solutions. Applying this heuristic solution method may be useful in one of the following cases:

1. When time limitations do not enable the application of an exact solution method and a near-optimal heuristic solution may be sufficient for the relevant requirements.
2. When the solution algorithm is used as a subroutine in a more general solution process, for example, a solution algorithm for the VRP.

In Section 4.1, we present our B&B algorithm, and in Section 4.2, we present the TS.

4.1 B&B algorithm for the S-TD-TSP-STW

The two main subcomponents of the B&B algorithm are the calculation of the upper bounds, i.e., the values of the feasible solutions, and the calculation of the lower bounds. Pseudocode of the algorithm is presented in Figure 1, and the two subcomponents are discussed in Sections 4.1.1 and 4.1.2. In Section 4.1.3 we present enhancements that accelerate the running time of the algorithm considerably.

| Decided = empty sequence |
| Undecided = the set of customers |
| Incumbent = the sequence of Undecided sorted by EDD |
| GlobalUB = CalcUB (Decided) |

Create a list $L$ with an entry $(\text{Decided, Undecided, CalcLB (Decided), GlobalUB})$

While $L \neq \emptyset$

Remove a node from $L$ and store as $(\text{Decided, Undecided, LB, UB})$

If $\text{LB} < \text{GlobalUB}$

For $i \in \text{Undecided}$

$LBI = \text{CalcLB}((\text{Decided}, i))$

$UBI = \text{CalcUB}((\text{Decided}, i))$

If $LBI < \text{GlobalUB}$

Insert to $L$ $((\text{Decided}, i), \text{Undecided} \setminus \{i\}, LBI, UBI)$

If $UBI < \text{GlobalUB}$

$\text{GlobalUB} = UBI$

$\text{Remaining} = \text{the sequence of customers Undecided} \setminus \{i\} \text{ sorted by EDD}$

$\text{Incumbent} = (\text{Decided}, i, \text{Remaining})$

**Figure 1:** Pseudocode of the specialized B&B algorithm

The list $L$ is the B&B tree that stores information about the nodes scanned while the algorithm runs. Each entry of $L$ has four components. The first component contains a sequence of customers for which the route was already decided, and the second component is a set that contains the rest of the customers. Clearly, any concatenation of the first component and some permutation of the customers of the second component is a feasible solution to the stated problem. The third and fourth components store the lower and upper bounds for the node.

We initialized $L$ with an empty $\text{Decided}$ sequence. The $\text{Undecided}$ set consists of all the customers. The lower bound and the upper bound are calculated using the functions $\text{CalcUB}$ and $\text{CalcLB}$, as described below. The initial solution of the algorithm is obtained as a sequence of $\text{Undecided}$ sorted in an increasing order of $b_i$. This solution is referred to as the earliest due date first (EDD). The variable
Incumbent stores the best solution found so far, while GlobalUB represents the value of the incumbent solution.

In each iteration of the process, one node is removed from L and stored as (Decided, Undecided, LB, UB). New potential entries are constructed by removing a single customer from the Undecided set and adding it to the end of the Decided sequence. The lower bound and the upper bound for this sequence are calculated. If the lower bound of the current sequence is smaller than the global upper bound, the entry is inserted back into L. If the upper bound (value of the feasible solution) of the current entry is smaller than the global upper bound, the incumbent solution and the global upper bound are updated. The process ends when the list is empty. However, if one is interested in an approximate solution, other stopping criteria may apply.

4.1.1 Calculating upper bounds

An upper bound for the value of the sequence (Decided, i) is obtained from the value of the objective function when the vehicle follows that sequence and then visits all the rest of the customers Undecided \ {i} according to the EDD rule and returns to the depot. The entire path constitutes a valid route since the vehicle visits all the customers. Recall that to calculate the objective function, we need to evaluate all the scenarios considering the time windows and the time-dependent travel times.

4.1.2 Calculating lower bounds

A lower bound for the sequence (Decided, i) consists of the sum of a lower bound for the working day duration and a lower bound for the sum of penalties.

The lower bound for the working day duration is obtained as the maximum of the following two lower bounds.

First, the accumulated duration of the working day up to customer i over all scenarios is $\sum_{k=1}^{K} (u_{ik} + s_{ik})$. To cope with the remaining customers, recall that the working day consists of traveling and service times. For each unvisited customer, $j \in \text{Undecided} \cup \{0\} \setminus \{i\}$, denote $\tilde{e}_{jk} = \min\{t_{l,j,t'},k| l \neq j, l \in \text{Undecided}, t'\}$, that is, the minimum travel time to j in scenario k over all departure times and relevant origins. Clearly, $\sum_{k=1}^{K} (\sum_{j \in \text{Undecided} \cup \{0\} \setminus \{i\}} \tilde{e}_{jk})$ is a lower bound for the remaining travel time over all scenarios. The remaining service time over all scenarios is $\sum_{k=1}^{K} \sum_{j \in \text{Undecided} \setminus \{i\}} s_{jk}$. Thus, $\sum_{k=1}^{K} (\sum_{j \in \text{Undecided} \cup \{0\} \setminus \{i\}} \tilde{e}_{jk}) + \sum_{k=1}^{K} \sum_{j \in \text{Undecided} \setminus \{i\}} s_{jk}$ is a lower bound for the remaining working day duration. The latter expression added to $\sum_{k=1}^{K} (u_{ik} + s_{ik})$ constitutes a valid lower bound for the sum of working day durations.

Second, following the triangular inequality, note that for each unvisited node, $j \in \text{Undecided} \setminus \{i\}$, $u_{jk}$ is minimal if j is visited immediately after i. In this case, $u_{jk} \geq \max(a_j, u_{ik} + s_{ik} + t_{i,j,u_{ik}+s_{ik}k})$. Consequently, $T_k \geq u_{jk} + s_{jk} + t_{j,0,u_{jk}+s_{jk}k}$. When generalized to include all scenarios, $\sum_{k=1}^{K} T_k \geq \sum_{k=1}^{K} u_{jk} + s_{jk} + t_{j,0,u_{jk}+s_{jk}k} \ \forall j \in \text{Undecided} \setminus \{i\}$. A lower bound is thus $\max\left\{\sum_{k=1}^{K} u_{jk} + s_{jk} + t_{j,0,u_{jk}+s_{jk}k} \mid j \in \text{Undecided} \setminus \{i\}\right\}$. 

9
The lower bound for the sum of the penalties is obtained as follows. The sum of penalties accumulated up to customer \(i\) over all scenarios is simply \(\sum_{k=1}^{K} \sum_{l \in \text{Decided} \cup \{i\}} G_{l}(o_{lk})\). Following the triangular inequality, for each unvisited customer \(j \in \text{Undecided} \setminus \{i\}, u_{jk}\) is minimal if \(j\) is visited immediately after \(i\). In this case, the penalty is also the minimal since \(G_{l}(x)\) is increasing in \(x\). Consequently, when \(u_{jk} = \max(a_{j}, u_{lk} + s_{ik} + t_{i,j;u_{lk}+s_{ik}k})\), the term \(\sum_{k=1}^{K} \sum_{l \in \text{Undecided} \setminus \{i\}} G_{l}(o_{lk})\) is a lower bound for the sum of future penalties. Thus, \(\sum_{k=1}^{K} \sum_{l \in \text{Decided} \cup \{i\}} G_{l}(o_{lk}) + \sum_{k=1}^{K} \sum_{j \in \text{Undecided} \setminus \{i\}} G_{j}(o_{jk})\) is a valid lower bound for the total penalties.

4.1.3 Algorithmic enhancements

We improved the performance of the specialized B&B algorithm by applying considerations that stem from the observation of Lemma 1 below.

Recall that \((\text{Decided}, i)\) is a sequence of visited customers that ends with the current customer \(i\). Let \(u_{lk}\) represent the arrival time at customer \(i\) in the \(k\)th scenario and let \(C_{i}\) represent the total penalties, over all scenarios, accumulated up to the arrival at customer \(i\) when following the sequence \((\text{Decided}, i)\). Let \(\text{Decided}\) represent an alternative sequence to \(\text{Decided}\) that contains the same customers but not necessarily in the same order. \(u_{ilk}\) and \(C_{i}\) are defined accordingly.

**Lemma 1:** If \(C_{i} \leq C_{i}\) and \(u_{lk} \leq u_{ilk} \forall k = 1, ..., K\), then the sequence \((\text{Decided}, i)\) weakly dominates the sequence \((\text{Decided}, i)\). That is, there exists an optimal sequence that does not contain \((\text{Decided}, i)\) as a prefix.

**Proof:** We consider two valid solutions for the S-TD-TSP-STW. The first solution is the sequence \((\text{Decided}, i, S)\) where \(S\) represents a permutation of the set \(\text{Undecided} \setminus \{i\}\). The second solution is the sequence \((\text{Decided}, i, S)\). Next, we show that the cost of the sequence \((\text{Decided}, i, S)\) is no greater than the cost of the sequence \((\text{Decided}, i, S)\); thus, the domination is proved.

Let \(j\) and \(l\) denote the first and last customers in \(S\). For customer \(j\), \(u_{jk} = \max(a_{j}, u_{lk} + s_{ik} + t_{i,j;u_{lk}+s_{ik}k})\) and \(u_{lk} = \max(a_{j}, u_{ilk} + s_{ik} + t_{i,j;u_{ilk}+s_{ik}k})\). Since the FIFO property in the time-dependent setting ensures that no later departure from origin \(i\) can result in an earlier arrival at destination \(j\). \(u_{lk} + s_{ik} + t_{i,j;u_{lk}+s_{ik}k} \leq u_{ilk} + s_{ik} + t_{i,j;u_{ilk}+s_{ik}k}\); thus, \(u_{jk} \leq u_{ilk} \forall k = 1, ..., K\). Clearly, a similar analysis can be done for all customers in \(S\). Therefore, for customer \(l\), \(u_{lk} \leq u_{ilk} \forall k = 1, ..., K\).

The total duration of the working day is the time from when the vehicle leaves the depot to the time at which it returns. For the sequence \((\text{Decided}, i, S)\), the duration is simply \(u_{lk} + s_{lk} + t_{l,0;u_{lk}+s_{ik}k}\). For the sequence \((\text{Decided}, i, S)\), the total duration is \(u_{ilk} + s_{lk} + t_{l,0;u_{ilk}+s_{ik}k}\). Since \(u_{lk} + s_{lk} \leq u_{ilk} + s_{lk} \forall k = 1, ..., K\), the FIFO property ensures that \(u_{lk} + s_{lk} + t_{l,0;u_{lk}+s_{ik}k} \leq u_{ilk} + s_{lk} + t_{l,0;u_{ilk}+s_{ik}k}\). That is, the total duration of the working day of the sequence \((\text{Decided}, i, S)\) is no greater than the duration of the sequence \((\text{Decided}, i, S)\).

Regarding the penalties, recall that for each customer \(p \in S\), \(u_{pk} \leq u_{pik} \forall k = 1, ..., K\). Recall also that \(G_{l}(x)\) is increasing in \(x\). Thus, for each customer \(p\), the penalty incurred when following the
sequence \((\text{Decided}, i, S)\) is no greater than the penalty incurred when following the sequence \((\text{Decided}, i, S)\). Since \(C_i \leq \tilde{C}_i\), we conclude that the total penalties when following the sequence \((\text{Decided}, i, S)\) are no greater than the penalties when following the sequence \((\overline{\text{Decided}}, i)\).

Since the total cost is the sum of working day duration and penalties, the claim is proven. Therefore, there exists at least one optimal route that does not begin with \((\overline{\text{Decided}}, i)\). ■

The practical implication of Lemma 1 is that branches in the B&B tree that contain the sequence \((\text{Decided}, i)\) may not be explored since there exists at least one optimal solution outside of these branches. To avoid the exploration of such sequences, we cache the values of the dominating sequences encountered during the search and compare each new sequence to previously encountered ones. The cache is stored in a hash table (denoted by \(A\)) indexed by the unordered set that constitutes each sequence and the identity of the last customer in the sequence, denoted by \([\{\text{Decided}\}, i]\). In the entry \(A[\{\text{Decided}\}, i]\) we store the vector of arrival times at \(i\), denoted by \(A[\{\text{Decided}\}, i], u_k\), and the sum of penalties accumulated up to \(i\) over all scenarios \(A[\{\text{Decided}\}, i].C\).

For every sequence \((\text{Decided}, i)\) encountered during the B&B search whose lower bound is smaller than the global upper bound, we check whether this sequence exists in the cache. If it does not exist, we create an entry \(A[\{\text{Decided}\}, i]\) in the cache. That is, \(A[\{\text{Decided}\}, i] = [u_{i1}, \ldots, u_{ik}], C_i\). If it exists in the cache, we check whether it is not dominated by the existing entry. That is, whether there exists \(k = 1, \ldots, K\) such that \(u_{ik} < A[\{\text{Decided}\}, i], u_k\) or \(C_i < A[\{\text{Decided}\}, i].C\). In such a case, we insert \((\text{Decided}, i)\) to the B&B tree. Next, we check whether \((\text{Decided}, i)\) strictly dominates \(A[\{\text{Decide}, i\}]\). In such a case, \(A[\{\text{Decide}, i\}]\) is updated.

4.2 TS algorithm for the S-TD-TSP-STW

In this section, we present a TS algorithm that is capable of producing good solutions for the problem in a relatively short time and scales better than the B&B framework applied earlier.

First, we define a neighborhood for a given solution. In our algorithm, a neighborhood consists of all possible sequences generated by swapping two served customers in the existing sequence and all sequences generated by moving a single customer along the current sequence.

Next, the tabu mechanism is implemented as follows. A tabu list is initialized, and a maximum size is set. The list stores information about the steps that are conducted as the algorithm runs. Each step constitutes a new entry in the list. For swapping steps, we store the two swapped customers. For moving steps, we store the moved customer as well as the preceding and successive customers related to the original sequence, that is, the sequence prior to the execution of the step. When the tabu list exceeds a predefined length, the insertion of a new entry is followed by the removal of the oldest one.

A swapping step is tabued when its reversal is currently stored in the tabu list. A moving step is tabued if its execution would create a new route in which either the newly moved customer or one of its neighbors (preceding or successive) belongs to one of the moving-step entries currently stored in the list.
The complete TS procedure is executed as follows. First, an initial solution is obtained as the sequence of all the customers calculated by the EDD rule. Next, the search procedure is run iteratively. In each iteration, the search procedure finds the best possible step related to the existing solution that is not tabued. This step is executed, and a new entry is inserted into the tabu list. The search procedure continues until a certain stopping criterion is reached, and the best solution encountered during the process is returned.

5 Numerical experiments

In this section, we present our extensive numerical experiments based on real data presented in Section 5.1. The goals of this numerical study are as follows:

1. Demonstrating the computational tractability of the specialized B&B algorithm as a solution method for instances with up to 18 customers. Note that it is highly unlikely for field service personnel to serve a greater number of customers in a single working day. See Section 5.2.
2. Assessing the performance of our TS heuristic in terms of both optimality gaps and running time. See Section 5.3.
3. Characterizing how the variance of stochastic service times affects the total cost of the optimal solutions as well as its effects on the performances of the exact and heuristic algorithms presented earlier. See Section 5.4.
4. Estimating the benefit of considering time dependency and stochasticity in the optimization process. See Section 5.5.
5. Validating the optimization over a set of scenarios as an effective method in addressing the randomness of real-life situations. See Section 5.6.
6. Demonstrating the merit of considering the inter-dependencies of the travel and service times in the optimization process. See Section 5.7.

To meet our last two goals, we divide the scenarios into two groups: training data and test data. The training data are used as the input for all the problems we solve. The test data are used in a later stage to evaluate the methods when applied for future unknown scenarios.

5.1 Problem instances

We have gathered data of real-life travel times from Google Map between 19 locations in central Israel, that is, one depot and 18 customers, at different times of the day during a period of 60 working days. The time granularity is 1 minute, i.e., a single time unit represents a 1-minute period. These data constitute 60 scenarios of time-dependent travel times.

Service times were randomly generated as follows. For each customer $i$, twenty values of $\mu_i$, the expected service time, are randomly generated using a lognormal distribution with a mean of 20 and standard deviation of 5. These are the 20 base instances. From these instances, 5 groups, each containing 20 instances, where each instance itself contains 60 scenarios, are generated in the following manner.

Let $\alpha_{km}, k \in 1, \ldots, 60, m = 1, \ldots, 5$ denote a factor matrix. $\alpha_{km}$ is drawn from a uniform distribution $U(1−0.05(m−1), 1 + 0.05(m−1))$. Next, the service time of customer $i$ in scenario $k$ in group $m$ is generated as follows:

$$s_{ikm} \sim N\left(\mu_i\alpha_{km}, \left(\frac{\mu_i\alpha_{km}}{4}\right)^2\right)$$
At the end of this process, 5 groups of 20 instances, where each instance contains 60 scenarios of service times and time-dependent travel times, are created. Let \( j \in 1, ..., 20 \) denote the instance number. The process insures that the \( j \)-th instances in each of the five groups are matched. That is, the expectancy of the service time of each customer \( i \) is identical among these instances, while its variance increases with the group number. This setup allows us to examine the effect of the variance on the value of the optimal solution and on the performance of our solution methods.

In addition, note that the service times \( s_{ikm} \) are created such that within the same scenario \( k \), there is a weak (but positive) correlation among the service times. This represents the fact that the service is affected by the state of the technician, e.g., mood and fatigue.

All the time windows were generated within a planning horizon of 540 minutes (9 hours). Two types of time windows were considered. In Type 1, there are three nonoverlapping time windows of three hours each, and the customers are equally divided among them. In Type 2, one-half of the customers can be served at any time during the day, and the rest are assigned to time windows of two hours. That is, 200 problem instances that contain a depot and 18 customers were created. Note that Type 1 time windows represent a typical situation in the field service industry when all the appointments are scheduled in advance by the contact center and the workload is balanced. Type 2 is characteristic of utility companies, where some of the tasks are related to infrastructure and others to indoor facilities. The infrastructure tasks can be conducted at any time during the working day while the indoor tasks require appointments with specific time windows.

Next, we created 400 smaller test instances: 200 with 12 customers and 200 with 15 customers. Each smaller instance contains the depot and a randomly selected subset of the 18 customers. The service times and the travel times are increased for these instances.

Recall that each instance has 60 scenarios. We divided these scenarios into two sets. Forty scenarios constituted the training dataset, i.e., the input in our experiment. An additional 20 scenarios composed the test dataset. These scenarios are used to evaluate the quality of the previously found solutions based on new data. This process simulates a real-life situation in which the planning is based on past scenarios, but the results are applied to future ones.

We chose \( G_i(x) = x \). However, the above solution methods can be applied with any increasing penalty function.

Our test machine is a standard desktop with an Intel i7-7700CPU, 3.6 GHz, and 32 GB RAM running Windows 10. The algorithms were implemented as single-threaded applications in Python 2.7.

Next, we discuss the results of the numerical experiments.

### 5.2 B&B algorithm

Table 1 presents the average, median, minimum, and maximum total running time until optimality is proven (in seconds) of the B&B algorithm for the instances with 12, 15 and 18 customers. These values are calculated based on 100 instances for each case. Table 2 presents the same measures for the time in which the optimal solution was encountered for the first time by the B&B algorithm.
Table 1: Total running time of the B&B algorithm until optimality is proven

<table>
<thead>
<tr>
<th>n</th>
<th>Type 1 time windows</th>
<th>Type 2 time windows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Median</td>
</tr>
<tr>
<td>12</td>
<td>4.29</td>
<td>2.91</td>
</tr>
<tr>
<td>15</td>
<td>34.35</td>
<td>20.02</td>
</tr>
<tr>
<td>18</td>
<td>133.46</td>
<td>97.56</td>
</tr>
</tbody>
</table>

Table 2: Running time until the optimal solution was encountered

<table>
<thead>
<tr>
<th>n</th>
<th>Type 1 time windows</th>
<th>Type 2 time windows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Average</td>
<td>Median</td>
</tr>
<tr>
<td>12</td>
<td>2.6</td>
<td>1.97</td>
</tr>
<tr>
<td>15</td>
<td>18.31</td>
<td>10.72</td>
</tr>
<tr>
<td>18</td>
<td>83.45</td>
<td>73.34</td>
</tr>
</tbody>
</table>

Instances with 12 customers and Type 1 time windows are the easiest. Most are solved to proven optimality in less than 3 seconds, and the average total running time is less than 5 seconds. Instances of the same size with Type 2 time windows are much harder to solve. For them, approximately two minutes are required for the worst case to reach proven optimality. However, the optimal solutions are obtained in up to 36 seconds. Instances with 15 customers and Type 1 time window are solved in a few minutes, while the solution process of the same size instances with Type 2 time windows take up to 7 minutes to reach the optimal solution and up to 15 minutes to prove it. Instances with 18 customers and Type 1 time windows take up to 10 minutes to be solved to proven optimality, while the same size instances with Type 2 time windows are computationally intensive and require a solution time of many hours. In these instances, the time to reach the optimal solution is also very long. In general, Type 2 instances are much harder to solve than Type 1 instances.

The B&B algorithm is shown to maintain computational tractability even for instances with 18 customers. However, the time required for the B&B solution procedure to reach the optimal solution makes this method unsuitable as a subroutine for the solution of multivehicle problems when routes with 15 customers or more are considered.

Next, we examine the performance of the heuristic TS algorithm.

5.3 TS algorithm

Table 3 presents the average optimality gaps of the solutions obtained by the TS algorithm and the exact B&B algorithm after 5 and 10 seconds of running. The optimality gap is calculated as follows: if a solution’s value is $Val$ and the value of the optimal solution is $Opt$, then the optimality gap is $\frac{Val}{Opt} - 1$.

All experiments in this section were conducted using a tabu list of $n$ items. In our preliminary experimentations, a tabu list of this size was found to yield good solutions.
Table 3: Average optimality gaps of the TS and B&B algorithms after 5 and 10 seconds of running

<table>
<thead>
<tr>
<th></th>
<th>Type 1 time windows</th>
<th>Type 2 time windows</th>
<th></th>
<th>Type 1 time windows</th>
<th>Type 2 time windows</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>5 seconds</td>
<td>10 seconds</td>
<td>5 seconds</td>
<td>10 seconds</td>
<td>5 seconds</td>
</tr>
<tr>
<td>n</td>
<td>TS</td>
<td>B&amp;B</td>
<td>TS</td>
<td>B&amp;B</td>
<td>TS</td>
</tr>
<tr>
<td>12</td>
<td>0.09%</td>
<td>0.17%</td>
<td>0.07%</td>
<td>0.02%</td>
<td>0.13%</td>
</tr>
<tr>
<td>15</td>
<td>0.62%</td>
<td>7.70%</td>
<td>0.32%</td>
<td>2.50%</td>
<td>0.59%</td>
</tr>
<tr>
<td>18</td>
<td>2.6%</td>
<td>17.22%</td>
<td>1.01%</td>
<td>12%</td>
<td>1.45%</td>
</tr>
</tbody>
</table>

The TS algorithm performs well for all instances, yielding an average optimality gap of at most 1.01% after 10 seconds. For instances with 12 and 15 customers, the average optimality gap after 5 seconds is at most 0.62%. Note that for instances with 12 customers and Type 1 time windows, the B&B algorithm yields even better optimality gaps than those of the TS after 10 seconds. However, for all other instances, the TS algorithm outperforms the B&B when given a time limit of 10 seconds. This trend is likely to hold and to be further strengthened when the number of customers is increased. Note that Table 3 clearly indicates that the TS algorithm is capable of producing high-quality solutions in a short time, even for computationally intense instances with 18 customers and Type 2 time windows.

Note that the advantage of the TS over the B&B is particularly apparent with respect to the Type 2 time windows. It seems that the less-constrained structure of the Type 2 windows is better suited to the heuristic local search algorithm than to the B&B, which is based on establishing tight lower bounds.

While the TS algorithm performs well given a time limit of 10 seconds, it may also be interesting to explore its performance when given a longer time to run. We followed this concept and ran the algorithm until it reached $n^3$ iterations. Table 4 and Table 5 present the average time spent (in seconds), the number of instances that converged to the optimal solution, the average optimality gap and statistics regarding the iteration and the time at which the best solution was first found. These statistics include the average, median and range of these values related to the 100 instances. Table 4 corresponds to Type 1 instances, and Table 5 corresponds to Type 2 instances.
Table 4: Performance measures of the TS algorithm for Type 1 time windows and \( n^3 \) iterations

<table>
<thead>
<tr>
<th>n</th>
<th>Average time spent in seconds</th>
<th>Number of instances that converged to optimal value (out of 100)</th>
<th>Average optimality gap</th>
<th>Best solution found on iteration #</th>
<th>Best solution found at time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Average</td>
<td>Median</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Average</td>
<td>Median</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Minimum</td>
<td>Maximum</td>
</tr>
<tr>
<td>12</td>
<td>49</td>
<td>98</td>
<td>0.01%</td>
<td>46.49</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>722</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1.44</td>
<td>0.35</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.18</td>
<td>19.8</td>
</tr>
<tr>
<td>15</td>
<td>191</td>
<td>97</td>
<td>0.05%</td>
<td>157.74</td>
<td>49.5</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>3</td>
<td>1,740</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>9.21</td>
<td>3.09</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.44</td>
<td>0.44</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>19.8</td>
<td>95.16</td>
</tr>
<tr>
<td>18</td>
<td>580</td>
<td>96</td>
<td>0.01%</td>
<td>500.58</td>
<td>259</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>15</td>
<td>4,169</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>49.8</td>
<td>25.46</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>2.21</td>
<td>404.14</td>
</tr>
</tbody>
</table>

The optimality gaps of the solutions obtained by the TS are very small. The average gap is in the range of 0.003% to 0.05%, and most of the instances are solved to optimality. The best solutions are found in a very short time compared with that of the B&B. In particular, for the most difficult instances with 18 customers and Type 2 time windows, the advantage of the TS is substantial. For example, the median time for finding the best solution with TS is 53 seconds while the median time for finding the optimal solution with B&B is 4,580 seconds.

In general, it seems that the time and number of iterations for finding the best solution are greater for the Type 2 instances than they are for the Type 1 instances. This finding suggests that obtaining a nearly optimal solution using the TS algorithm is harder in Type 2 instances, which is also the case for the B&B algorithm.

Indeed, the exact B&B algorithm requires many hours to converge when applied to the instances with 18 customers and Type 2 time windows. Thus, one may argue that for these instances, as well as for larger instances, the TS algorithm can be used as an alternative way to obtain a nearly optimal solution. Table 6 presents the average, median, minimum and maximum optimality gaps that the TS algorithm yielded when stopped at \( n^3 \) iterations, as well as these measures for the B&B algorithm running for the same computation time. The results are presented for the instances with 18 customers and Type 2 time windows.
Table 6: Optimality gaps for instances with 18 customers and Type 2 time windows

<table>
<thead>
<tr>
<th>TS (n^3 iterations)</th>
<th>B&amp;B (after the same computation time)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average</td>
<td>Median</td>
</tr>
<tr>
<td>0.04%</td>
<td>0</td>
</tr>
</tbody>
</table>

The TS algorithm outperforms the B&B on every measure and appears to be better able to cope with the computationally intense instances with 18 customers and Type 2 time window when given a time limit of roughly ten minutes. This trend is likely to hold as the number of customers considered increases.

5.4 The effect of the variance

In this section, we discuss the effect of the variance of the service times on the running times of the two algorithms as well as on the values of the optimal solutions.

We begin with an analysis of the total running time of the B&B algorithm until optimality is proven and the running time of the B&B algorithm until the optimal solution is encountered. For each instance, the value of the time in the first variance group is set as the baseline. We calculate the ratio between the values of the matched instances in all other variance groups and this baseline. These ratios are clustered by the value of n (12, 15, 18) and by the type of the time window (1, 2). Figure 2 presents the average values of the ratios of the total running time of the B&B algorithm until optimality is proven for each cluster by variance group. Figure 3 presents similar data for the running time of the B&B algorithm until the optimal solution is first encountered.

Figure 2: The effect of service time variance on the running time of the B&B algorithm until optimality is proven
Figure 3: The effect of service time variance on the running time of the B&B algorithm until the optimal solution is first encountered.

In both Figure 2 and Figure 3, it is apparent that the solution time is strongly affected by the service time variance. Larger variance is associated with harder problem instances for the B&B algorithm. The effect seems stronger for the more difficult instances with more complex time-window structures (Type 2) and with a higher number of customers.

The effect of the variance of the stochastic service times on the performance of the TS algorithm is analyzed by using the optimality gaps obtained after 10 seconds. As before, we cluster these gaps by the values on \( n \) (12, 15, 18) as well as by the type of the time window (1, 2) and present the average gap of each cluster, for all variance groups, in Figure 4.

Figure 4: Average optimality gaps of TS after 10 seconds for all variance groups by clusters.
No consistent effect of the service time variance on the quality of the solutions produced by TS after 10 seconds is observed in Figure 4. Clearly, the TS is more robust than B&B to the variance in stochastic service times.

Next, we explore the effect of the variance of the stochastic service times on the values of the optimal solutions. These values are comprised of the total working day duration and the penalties incurred. The two components are referred to as the performance measures below.

For each instance, the value of the performance measure in the first variance group is set as the baseline. We calculate the ratio between the values of the matched instances in all other variance groups and this baseline. These ratios are clustered by the value of $n$ (12, 15, 18) and by the type of time window (1, 2). Figure 5 presents the average value of this ratio for each cluster, for the total duration, for all variance groups. Figure 6 presents similar data for the sum of the penalties.

![Figure 5: Effect of the service time variance on the total duration](image1)

![Figure 6: Effect of the service time variance on the total penalties](image2)
In Figure 5 and Figure 6, both cost components are mostly increasing with increasing service time variance, as expected. The effect is much stronger on the penalties, but this is clearly related to the parameters of our instances. For instances with Type 2 time windows, the effect of the service time variance is much stronger, possibly because this type gives the planner more flexibility.

To conclude this section, greater service time variance is associated with a longer solution time for the B&B algorithm, but the TS is not affected. The cost of the solution is naturally increasing with the variance.

5.5 The added value of considering time dependency and stochasticity

In this section, we wish to estimate the benefit achieved by considering time dependency and stochasticity in the optimization process rather than following the traditional approach and planning the route based on the nominal (average) travel and service times. Therefore, we solved a stochastic but time-independent version of the problem (denoted by S-TSP-STW), a time-dependent deterministic version (denoted by TD-TSP-STW) and deterministic time-independent version (denoted by TSP-STW). As the input for these three special cases, we used the relevant averages of the time-dependent and stochastic data that were used for the full model. Moreover, we used adapted versions of the B&B algorithm presented earlier as solution methods.

Next, using the original input (with all the scenarios), we calculated the value of the optimal solution of each model with respect to the objective function of the S-TD-TSP-STW. These solutions were compared with the optimal ones of the full model by calculating the relative gaps between the solution values. The average gap of each model represents the loss incurred due to the neglectation of one or two aspects of the S-TD-TSP-STW. That is, the gap of S-TSP-STW is the average loss incurred because time dependency was not considered, the gap of TD-TSP-STW is the average loss incurred from neglecting stochasticity, and the gap of TSP-STW is the average loss incurred from neglecting both stochasticity and time dependency. In addition, we calculated the gaps related to the two components of the objective function of the S-TD-TSP-STW, that is, the total duration of the working day and the sum of the penalties. Note that in this case, the gaps with respect to one of the cost components may be negative.

The average gaps for the three models of the problem are presented in Table 7 for each value of \( n \) and for the two types of time windows. Table 8 displays the average, median, minimum and maximum calculation times for the solution of all models (in seconds).
Table 7: Average gaps of the special cases of the S-TD-TSP-STW

<table>
<thead>
<tr>
<th>n</th>
<th>Type</th>
<th>S-TSP-STW</th>
<th>TD-TSP-STW</th>
<th>TSP-STW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Total cost</td>
<td>Duration</td>
<td>Penalties</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>0.33%</td>
<td>0.12%</td>
<td>2.08%</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>0.79%</td>
<td>0.12%</td>
<td>49.01%</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0.65%</td>
<td>0.58%</td>
<td>6.15%</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>0.34%</td>
<td>0.04%</td>
<td>44.32%</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>0.52%</td>
<td>0.19%</td>
<td>14.32%</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>0.45%</td>
<td>-0.1%</td>
<td>165.54%</td>
</tr>
</tbody>
</table>

Table 8: Total running time for the exact solution of special cases of the S-TD-TSP-STW, in seconds

<table>
<thead>
<tr>
<th>n</th>
<th>Type</th>
<th>S-TSP-STW</th>
<th>TD-TSP-STW</th>
<th>TSP-STW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>average</td>
<td>median</td>
<td>min</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>2.78</td>
<td>1.69</td>
<td>0.58</td>
</tr>
<tr>
<td>12</td>
<td>2</td>
<td>20.27</td>
<td>15.63</td>
<td>4.83</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>18.42</td>
<td>10</td>
<td>4.61</td>
</tr>
<tr>
<td>15</td>
<td>2</td>
<td>149.22</td>
<td>131.11</td>
<td>48.53</td>
</tr>
<tr>
<td>18</td>
<td>1</td>
<td>97.19</td>
<td>78.8</td>
<td>34.59</td>
</tr>
<tr>
<td>18</td>
<td>2</td>
<td>3,426</td>
<td>2,758</td>
<td>949</td>
</tr>
</tbody>
</table>

Table 7 clearly indicates that the main merit of solving the intricate S-TD-TSP-STW is the decrease in the expected penalties. While the total duration of the working day hardly changes between the S-TD-TSP-STW and the simpler models, the penalties can increase by a factor of up to four. The difference is much more substantial for the instances with Type 2 time windows.

The average loss incurred by solving the TSP-STW ranges from 0.31% to 1.41%. Both stochasticity and time dependency are proved to be important for the optimization process. Neglecting the time-dependent aspect of the S-TD-TSP-STW may result in a loss of up to 0.8% of the total cost. Neglecting the stochastic aspect of the problem may result in a loss of up to 0.9%. In addition, time dependency appears to be more important to consider than stochasticity in instances with Type 1 time windows. This result is evident from the fact that the average losses for the S-TSP-STW model are greater than the losses for the TD-TSP-STW model for this type of instances.

The effect of neglecting time dependency and the effect of neglecting stochasticity are not additive. That is, the loss incurred by solving the TSP-STW problem is not equal to the sum of losses incurred by solving the stochastic problem and the time-dependent problem, which suggests an interaction between the effect of neglecting time dependency and the effect of neglecting stochasticity.

The computational effort of solving the S-TSP-STW model is similar to that of solving the S-TD-TSP-STW. However, the effort required to solve the TD-TSP-STW and TSP-STW problems is smaller by a factor of approximately 100. Note that in terms of total cost, all the methods provide solutions that are close to the optimum, with a maximum gap of 1.41%. Thus, a reasonable heuristic approach to the solution of the S-TD-TSP-STW may be to use the solution obtained by the TSP-STW or TD-TSP-STW models. To check the applicability of this approach, we solved each of the 600 instances by applying the B&B for the S-TSP-STW, TD-TSP-STW and TSP-STW models with a 10 second time limit. In addition, we applied the TS for the full model with the same time limit. Next, for each instance, we identified the best solution obtained (out of four) with respect to the objective function of the full model.
Table 9 presents the number of times out of 100 instances in which each model reached that best solution for each value of \( n \) and each type of time window. The largest value in each row is in bold font.

**Table 9**: The number of times each model yielded the best solution after 10 seconds

<table>
<thead>
<tr>
<th>n</th>
<th>Time windows</th>
<th>S-TSP-STW</th>
<th>TD-TSP-STW</th>
<th>TSP-STW</th>
<th>TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Type 1</td>
<td>60</td>
<td>60</td>
<td>54</td>
<td>94</td>
</tr>
<tr>
<td>12</td>
<td>Type 2</td>
<td>26</td>
<td>41</td>
<td>16</td>
<td>90</td>
</tr>
<tr>
<td>15</td>
<td>Type 1</td>
<td>32</td>
<td>61</td>
<td>24</td>
<td>72</td>
</tr>
<tr>
<td>15</td>
<td>Type 2</td>
<td>1</td>
<td>19</td>
<td>30</td>
<td>73</td>
</tr>
<tr>
<td>18</td>
<td>Type 1</td>
<td>0</td>
<td>75</td>
<td>67</td>
<td>32</td>
</tr>
<tr>
<td>18</td>
<td>Type 2</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>97</td>
</tr>
</tbody>
</table>

From Table 9, it is apparent that although none of the solution approaches dominates all the others, applying the TS on the full model for 10 seconds consistently delivers high-quality solutions. In particular, the advantage of the TS is clear for the more difficult Type 2 time window instances. Using the solution obtained from B&B after 10 seconds for the S-TSP-STW and TSP-STW models is not an attractive approach. Indeed, the solution of the S-TSP-STW model requires a similar amount of computational effort to that of the full model while ignoring an important characteristic of the problem. The TSP-STW model is easy to solve but ignores the two complicating aspects of the S-TD-TSP-STW.

**5.6 Validating the use of scenarios**

In this section, we analyze the use of scenarios as a valid means of modeling and optimization in stochastic settings. Recall that the data we gathered (and created) were divided into a training dataset of 40 scenarios that was used as an input for all the optimization algorithms and test dataset of 20 scenarios that was left aside to evaluate the solutions. Next, we validate the expected contribution of solving the S-TD-TSP-STW in terms of the objective function calculated based on the test data rather than the training data. This evaluation simulates the situation in which a planner sets the routes of the field service personnel based on historical travel and service times but the costs are determined by their future realizations. The procedure we execute is as follows:

1. We calculate the values of the total cost, working day duration and penalties of the solutions obtained in the previous sections by:
   a. The solution of S-TD-TSP-STW obtained by the exact B&B algorithm
   b. The solution of S-TSP-STW obtained by the exact B&B algorithm
   c. The solution of TD-TSP-STW obtained by the exact B&B algorithm
   d. The solution of TSP-STW obtained by the exact B&B algorithm
   e. The solution of S-TD-TSP-STW obtained by the TS algorithm after 10 seconds

2. We set the values of the optimal solutions of S-TD-TSP-STW as the baseline and calculate the gap of all other solutions relative to the baseline. Note that in this case, some gaps may be negative.

The generated gaps for the solutions of the simpler models constitute the losses incurred due to the neglect of one or two aspects of the S-TD-TSP-STW (see Subsection 5.5) with respect to the test data. The gap of the solutions of the TS represents the loss incurred from implementing a heuristic
solution method with respect to the test data. Table 10 presents the average gap (in percentage) for each value of $n$ and the two types of time windows.

**Table 10**: Gaps of all the solution methods calculated using the test data

<table>
<thead>
<tr>
<th>$n$</th>
<th>Type 1</th>
<th>Type 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>S-TSP-STW</td>
<td>0.30%</td>
</tr>
<tr>
<td></td>
<td>Total cost</td>
<td>1.77%</td>
</tr>
<tr>
<td></td>
<td>Duration</td>
<td>0.03%</td>
</tr>
<tr>
<td></td>
<td>Penalties</td>
<td>0.48%</td>
</tr>
<tr>
<td></td>
<td>TD-TSP-STW</td>
<td>1.28%</td>
</tr>
<tr>
<td></td>
<td>Total cost</td>
<td>163.54%</td>
</tr>
<tr>
<td></td>
<td>Duration</td>
<td>0.15%</td>
</tr>
<tr>
<td></td>
<td>Penalties</td>
<td>2.93%</td>
</tr>
<tr>
<td></td>
<td>TSP-STW</td>
<td>0.70%</td>
</tr>
<tr>
<td></td>
<td>Total cost</td>
<td>1.77%</td>
</tr>
<tr>
<td></td>
<td>Duration</td>
<td>0.03%</td>
</tr>
<tr>
<td></td>
<td>Penalties</td>
<td>0.48%</td>
</tr>
<tr>
<td></td>
<td>TS after 10 seconds</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Total cost</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Duration</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Penalties</td>
<td>-</td>
</tr>
</tbody>
</table>

In Table 10, we observe that the exact solution of the full model produces better solutions than the partial or heuristic ones. The gaps are similar to those obtained for the training data, as presented in Table 7. We conclude that the use of historical scenarios is beneficial when optimizing the routes under travel and service time uncertainty.

5.7 The added value of considering the inter-dependency of travel and service times

We end the numerical experiments section with a demonstration of the merit of considering possible inter-dependencies among travel and service times in the optimization process. For this end, we generated 40 scenarios for the same depot and 18 customers with independent service and travel times. Note that although the travel times within a given scenario are mutually independent, they are still time dependent. We implemented the following procedure:

1. For each time-dependent travel (or service) time, we constructed its empirical discrete distribution using the corresponding values of the 40 scenarios of the training dataset as input.
2. The value of each time-dependent travel (or service) time in each generated scenario was randomly generated based on its discrete distribution.

Next, we solved all problem instances of the S-TD-TSP-STW with the generated travel and service times as inputs. Finally, we calculated the relative gaps between the solutions obtained in this manner and the solutions obtained in Section 5.2. The gaps were calculated with respect to the 40 original scenarios of the training dataset as well as the 20 scenarios of the test dataset. The average values of the gaps are presented in Table 11.

**Table 11**: Gaps between the solutions that consider inter-dependency and the solutions that do not

<table>
<thead>
<tr>
<th>$n$</th>
<th>Time windows</th>
<th>Avg. gap (training dataset)</th>
<th>Avg. gap (test dataset)</th>
</tr>
</thead>
<tbody>
<tr>
<td>12</td>
<td>Type 1</td>
<td>0.63%</td>
<td>0.62%</td>
</tr>
<tr>
<td>12</td>
<td>Type 2</td>
<td>0.30%</td>
<td>0.30%</td>
</tr>
<tr>
<td>15</td>
<td>Type 1</td>
<td>0.83%</td>
<td>1.00%</td>
</tr>
<tr>
<td>15</td>
<td>Type 2</td>
<td>0.30%</td>
<td>0.33%</td>
</tr>
<tr>
<td>18</td>
<td>Type 1</td>
<td>0.43%</td>
<td>0.44%</td>
</tr>
<tr>
<td>18</td>
<td>Type 2</td>
<td>0.35%</td>
<td>0.28%</td>
</tr>
</tbody>
</table>
In Table 11, we observe that overlooking the inter-dependencies of the travel and service times may result in an increase in the total cost of up to 1%. The average gaps displayed in Table 11 were all significantly greater than zero (p-value < 0.001). Moreover, the gaps for the test dataset are very similar to the gaps calculated using the training dataset. Interestingly, comparison of the results presented in Tables 10 and 11 shows that the consideration of the inter-dependencies is at least as important as considering the stochasticity and time-dependency properties.

6 Conclusions
In this paper, we introduced a model that captures the stochastic and time-dependent nature of the field service routing and scheduling task. In particular, our model considers the intricate inter-dependencies between travel times and service times by optimizing over a large set of scenarios. We devised an exact solution method as well as a successful TS heuristic that delivers high quality solutions in a short time and scales better.

Through an extensive numerical study, we demonstrated that our rich model leads to robust solutions that are, on average, better than the solutions of the simpler models that are common in practice. These simpler models abstract out the stochasticity and/or the time dependency. The advantages of our solutions are both in terms of the total duration and in terms of the service level as measured by our convex penalty function. The benefit in terms of the expected service level is particularly large. We also showed that the merits of our model are preserved when the solutions are applied to future scenarios that are not available in the planning phase. We used 40 historical scenarios for optimization and showed that such a set is sufficiently large to avoid overfitting.

For future research, we note that our model and solution methods can be adapted to other single-vehicle routing problems outside of the context of field service routing and scheduling. Moreover, our solution methods can be used as a subroutine in algorithms that solve multivehicle field service routing and scheduling problems. Such algorithms can be based, for example, on column generation or ALNS.

Acknowledgment
This research was supported by the Israel Science Foundation (ISF) grant no 1367/17

References


