The data-driven time-dependent traveling salesperson problem

Edison Avraham and Tal Raviv

Department of Industrial Engineering

Tel Aviv University, Ramat Aviv, Tel Aviv 6997801, Israel

Email: edisonavr@gmail.com, talraviv@eng.tau.ac.il

June 2019

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 may be interdependent. The objective is to minimize the expected length of the route duration 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 interdependencies in the road network to be captured. We introduce a specialized branch-and-bound algorithm and a successful adaptive large neighborhood search heuristic for the problem. In a numerical experiment based on real historical travel time data, we demonstrate the applicability of both methods to problem instances of up to 36 customers and 40 scenarios. These dimensions are a safe upper bound for instances originating from the field service operation domain. The resulting routes are tested on realistic scenarios that were not included in the problem input (the training set) to demonstrate the merits of using historical data. Compared with solutions that ignore the time dependency and/or stochasticity of the parameters, our solutions are consistently superior.

Keywords: Time-dependent, Traveling 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 estimations of the abovementioned times. However, such a solution may be suboptimal when implemented in real-life situations including 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 distributions of the travel times tend to vary, reflecting the different traffic 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 customer locations are assumed to be known. Each customer has a time window. Two types of time windows, soft and hard, have been studied. Soft time windows allow late arrivals at customer locations, and each late arrival incurs a penalty. The objective function, in this case, minimizes a weighted sum of the traveling and penalty costs associated with late arrivals at customer locations. The existence of hard time windows may cause a problem instance to have no feasible solution, especially when traveling and service times are stochastic.

In the above context, most studies have focused on the deterministic and time-independent version of the problem. More recent studies have mostly considered either stochasticity or time dependency but not both simultaneously. Moreover, the few studies that address these two features simultaneously generally assume independence 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 resulting travel times are dependent.

Our approach to considering 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 data-driven and time-dependent TSP with soft time windows (DD-TD-TSP-STW). We developed a specialized branch-and-bound (B&B) algorithm that is capable of solving real-life instances. Then, we devised an adaptive large neighborhood search (ALNS) algorithm to find high-quality solutions for the problem in a shorter time. Next, we conducted numerical experiments using actual travel time data collected from Google Maps. 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 reviews the state-of-the-art literature on stochastic and time-dependent vehicle routing. Section 3 defines our notation and formally states the DD-TD-TSP-STW problem. The solution methods are presented in Section 4, and the numerical experiments and their 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. Section 2.1 presents literature concerning time-dependent routing problems. Section 2.2 presents literature related to vehicle routing with stochastic travel and service times. Section 2.3 surveys the few recent papers that simultaneously address 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) presented 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) calculated travel time based on a step function that represents the traveling speed over time and considers 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 of the departure time.

Fleischmann et al. (2004) demonstrated that a continuous piecewise linear travel time function satisfies the FIFO property if its slopes are strictly greater than -1. They described the derivation of travel time data from traffic information systems and presented a general framework for the implementation of time-dependent travel times in various vehicle routing models. Some computational tests on real data from the city of Berlin were reported.

Haghani and Jung (2005) addressed 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) considered a capacitated vehicle routing problem (CVRP) with time-dependent travel times and stochastic service times, in which the duration of each route is limited. The problem was solved by a tabu search (TS) algorithm.

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

Verbeeck et al. (2014) studied 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 quickly yields good solutions.

Cordeau et al. (2014) studied the TD-TSP with the objective of minimizing the duration of the route. The authors proved that when all arcs in the road network share a common congestion pattern, the optimal solution for the TD-TSP can be found by solving a simpler time-independent asymmetric TSP. Next, they formulated the TD-TSP as an integer linear program, derived valid inequalities and embedded them in a branch-and-cut algorithm. Arigliano et al. (2018) presented a combinatorial branch-and-bound algorithm for the same problem that outperforms the previous branch-and-cut algorithm for large instances.

Arigliano et al. (2015) studied TD-TSP-TW with the objective function of minimizing the duration of the route. The authors proved that if congestion patterns are the same throughout the road network, the problem can be solved as an asymmetric TSP with time windows where adjusted windows and constant travel times are its input. In addition, they showed that when these conditions are not met, optimal solutions for the asymmetric TSP give lower and upper bounds for the stated problem. An integer program is presented, and the problem is solved by developing a branch-and-cut algorithm.

Montero et al. (2017) studied the TD-TSP-TW problem with the same objective function, which they refer to as the *makespan*. The authors formulated the problem as an integer program and devised an exact branch-and-cut algorithm. Preprocessing rules and valid inequalities that improved the performance of the algorithm were incorporated. Instances with up to 40 customers were solved.

Vu et al. (2018) addressed the TD-TSP-TW. The authors considered two objective functions, minimizing the duration of the route, including and excluding the waiting time at the depot before the departure. First, they formulated the problem as an integer program that is related to a time-space network. Next, they implemented a framework of a dynamic discretization discovery (Boland 2017). In this framework, the problem is solved iteratively. However, not all time points are represented in each iteration of the solution process. Indeed, each partial time-space network is used to solve a relaxation of the problem, and hence, lower and upper bounds are derived. Next, the partial network is modified dynamically until optimality is reached. The algorithm was shown to outperform previous solution methods for the problem.

Arigliano et al. (2019) studied the TD-TSP-TW problem. The objective was to minimize the duration of the route. The problem was solved using a B&B framework with the aid of a novel domination rule that is based on the time windows. Instances with up to 40 customers were solved.

2.2. Vehicle routing with stochastic travel and service times

The stochasticity of the travel and service times in VRP can be addressed either by applying a static (off-line) solution with the goal of optimizing the expectation of the objective function over all possible scenarios or by devising a dynamic (on-line) policy.

To the best of our knowledge, the first study of off-line models for the stochastic routing problem was Laporte et al. (1992). They presented stochastic programming models for the uncapacitated vehicle routing problem with deadlines where the service and travel times are stochastic. They applied two different modeling approaches: either limiting the probability of exceeding the deadline or penalizing for its violation. Kenyon and Morton (2003) studied a similar problem and presented two formulations for the problem: one formulation aims to minimize the makespan, and the other formulation maximizes the probability of completing all the routes by some deadline.

Capacity and soft time windows were considered by Li et al. (2010). They presented a model that limits the probability of violating the time windows and the probability of exceeding a given route duration. An initial solution was generated by a TS algorithm. Lei et al. (2012) solved a CVRP with stochastic service times, where the duration of the route is constrained. The objective function minimizes the sum of the travel, service, and expected penalty costs. The authors presented a closed-form expression for the expected cost of a single route.

Tas et al. (2012) proposed a three-stage TS procedure for the VRP with stochastic travel times and soft time windows. Both the operational cost and customer inconvenience were optimized. Tas et al. (2013) presented 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) presented a robust optimization method for the field service routing problem with stochastic service times and a soft deadline for the starting time of the service for each customer. Errico et al. (2013) presented a VRP with stochastic service times and hard time windows. The authors suggested a framework that limits the probability of violating any time window. They introduced a model based on the set-partitioning formulation for the problem.

Ehmke et al. (2015) solved the VRP with stochastic travel times and hard time windows. Stochasticity was dealt with by limiting the probability of violating the time windows. The travel times were assumed to be normally distributed. As an alternative to an exact and intense calculation of the distributions of arrival times at each customer, approximations were used. The authors applied statistical considerations on the distribution of the maximum of two independent variables and showed that the distribution of the service starting times at each customer is approximately normal. Finally, the authors created a feasibility test and incorporated it into their solution algorithm. They noted that this framework enables a quick solution of fairly large instances.

The exact solution of the dynamic version of the stochastic problem can be based on mapping each possible state of the system (characterized by time of the day, location of the vehicle, available information about the traffic and other parameters) to a routing decision. Since the state space is too large, practical solution approaches are divided into the following types: approaches that approximate the state and action spaces and approaches that repeatedly solve the problem in a rolling horizon manner based on the current state of the system and some approximated information about the future.

Delage (2010) solved a multidepot VRP-TW 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 presented two solution methods for the problem. The first method starts by finding routes that serve only repairs. Later, the maintenance requests are inserted into the routes. The solution is updated repeatedly using a TS algorithm after each time when an 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 skipped customers if necessary.

Binart et al. (2016) addressed a similar problem with stochastic travel times and devised 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) solved the VRP with hard deadlines on the starting time of the service. They assumed that the service times follow a discrete distribution. Two recourse strategies to cope with a violation of the next customer's time window were 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

Gendreau et al. (2015) surveyed the research in the area of time-dependent routing and noted that stochastic and time-dependent routing is in its infancy. However, the few studies that do address time dependency and stochasticity simultaneously focus on static settings; that is, the planned routes are executed without modifications.

Nahum and Hadas (2009) studied the stochastic time-dependent VRP (TD-S-VRP). They presented a mathematical model for the problem that limits the probability of exceeding some given bound on the travel time. The objective function minimizes the expected total travel time. The authors developed a saving heuristic executed in polynomial time and inspired by Clarke and Wright (1964).

Lecluyse et al. (2009) considered a TD-S-VRP that includes the variance and expected value of the travel time in the objective function. The authors assumed that the travel times between locations follow a lognormal distribution and approximate the total travel time of a route using this distribution. The problem was solved by applying a TS algorithm.

Tas et al. (2014) addressed a variant of the TD-S-VRP with soft time windows, i.e., the service can start before or after the time window. The objective function minimizes a weighted sum of the expected transportation costs and penalties for lateness and earliness. The authors showed 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 devised two solution methods for the problem: a TS algorithm and an ALNS algorithm.

Duan et al. (2015) solved a TD-S-VRP with hard time windows. They assumed that the support of the travel time distributions is bounded. The time window constraints were enforced based on the maximal possible travel times, while the objective function was based either on expected travel times or maximal travel times. They presented an ant colony optimization algorithm for the problem.

Verbeeck et al. (2016) studied a stochastic version of the time-dependent orienteering problem with hard time windows. The travel times were assumed to follow a normal distribution, while service times were deterministic. They devised a method to approximate the arrival time at each customer, which is challenging due to the hard time windows. The problem was solved using an ant colony algorithm. Numerical experiments demonstrated the merit of considering time dependency and stochasticity compared to using the nominal deterministic and fixed times.

Çimen et al. (2017) extended the green CVRP to accommodate time-dependent and stochastic travel speeds, which are crucial for green routing. They formulated the problem as a Markovian decision process (MDP). The objective function was to minimize the sum of the route duration and fuel costs over all vehicles. They solved the problem by an approximate dynamic programming (ADP)-based heuristic. Numerical experiments illustrated the applicability of the heuristic in terms of running times and demonstrated the merit of considering the stochasticity of travel times in the model.

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 interdependencies between traffic conditions in close geographical locations are substantial. Moreover, since there is a positive correlation between the travel times, the independence assumption may lead to plans that are too optimistic and result in many service delays.

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 merits. 1. It is relatively easy to create input for the problem based on historical travel and service time data. 2. Scenarios readily capture the inherent and complicated dependency between the travel times of journeys that are spatially or temporally close.

We note that the generation of scenarios based on the estimation of the travel time distributions is neither practical nor desirable in our case since there is insufficient data to approximate the joint distribution of the time-dependent travel times. Note also that it is not sensible to use data from the "far" history (for example, more than several months ago) since the traffic patterns are rapidly changing over time. Therefore, we advocate using travel time observations collected over a period of the last few weeks in the network on similar days. The shortfall of this approach is that it is hard to verify the validity of the result of each single instance of the problem. We partially overcome this issue by repeating our experiments with many different sets of customers and their corresponding travel time scenarios. When tested on the travel times in subsequent days, our method consistently delivers better solutions than those that are based on average (fixed or TD) travel times.

The main contribution of this study is to introduce 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 interdependent. The stochasticity is modeled by a set of scenarios that can easily be collected from GIS systems such as Google Maps. In addition, we present an ALNS heuristic for the problem that is capable of delivering near-optimal solutions in a relatively short time. We demonstrate the effectiveness of both the exact and heuristic methods using real travel time data that are not included in the scenarios used by the algorithm. Finally, we demonstrate the importance of considering time dependency and stochasticity rather than following the traditional approach of solving the problem based on average times.

3 Problem definition

The DD-TD-TSP-STW is stated as follows. A set of customers must be served on 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 location after the end of the customer's time window incurs a penalty, which is increasing (not necessarily linearly) in the extent of the lateness. When the vehicle arrives at a customer location before the beginning of its time window, it waits until the window is opened. This choice of time window represents the practice of field service operations where late arrivals may be unavoidable when the travel and service times are unknown. However, earlier arrivals can be avoided by idling the technician for some time.

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 travel time is a function of the origin, destination, departure time, and scenario. The service time is given for each customer in each scenario. A solution is the sequence by which the customers should be visited to minimize the expected sum of the route duration and penalties for the violations of the time windows over all the scenarios.

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

n	Number of customers; 0 represents the depot.
K	Number of scenarios.
$[a_i, b_i]$	Time window for service at customer's location i.
s_{ik}	Service time at customer's location i in the k^{th} scenario.
$t_{i,j,k}(t')$	Travel time between customer i and customer j at departure time t' for the k^{th} 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, $t_{i,j,k}(t') \leq t_{i,l,k}(t') + t_{l,j,k}\left(t' + t_{i,l,k}(t')\right) \quad \forall i,j,l = 0,\dots,n \quad \forall k = 1,\dots,K \quad \forall t'$.
$G_i(x)$	Penalty function for late arrival at customer's location i. The exact shape of the penalty function is an input of this model and should be determined by the service level agreement between the providers and their customers. We assume that $G_i(x)$ is a nondecreasing positive function.

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

Decision Variables

x_{ij}	Binary variable that equals "1" if customer j is visited immediately after customer i
u_{ik}	Time when the service of customer i begins in realization k
o_{ik}	Lateness at customer's location i in realization k
T_k	Route duration in realization k

$$minimize\ \frac{1}{K}\left(\sum_{k=1}^{K} T_k + \sum_{i=1}^{n}\sum_{k=1}^{K} G_i(o_{ik})\right) \tag{1}$$

$subject\ to$

$$\sum_{j=0}^{n} x_{ij} = \sum_{j=0}^{n} x_{ji} \qquad \forall i = 0,\dots,n \tag{2}$$

$$\sum_{j=0}^{n} x_{ij} = 1 \qquad \forall i = 0,\dots,n \tag{3}$$

$$u_{jk} \geq \left(u_{ik} + s_{ik} + t_{i,j,k}(u_{ik} + s_{ik})\right)x_{ij} \qquad \forall i = 0,\dots,n; j = 1,\dots,n, k = 1,\dots,K \tag{4}$$

$$a_i \leq u_{ik} \qquad \forall i = 1,\dots,n, k = 1,\dots,K \tag{5}$$

$$b_i + o_{ik} \geq u_{ik} \quad \forall i = 1,\dots,n, k = 1,\dots,K \tag{6}$$

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

$$x_{ij} \in \{0,1\} \quad \forall i, j = 0, \dots, n \tag{8}$$

$$u_{ik} \geq 0 \quad \forall i = 0, \dots, n; \quad k = 1, \dots, K \tag{9}$$

$$o_{ik} \geq 0 \quad \forall i = 0, \dots, n; \quad k = 1, \dots, K \tag{10}$$

The model aims to minimize the expected route duration and penalty costs (1). Constraint (2) maintains vehicle flow conservation while constraint (3) ensures that all customers are visited. Constraint (4) relates the starting times of the service with the routing variables. Constraint (5) enforces hard lower bounds on the starting times of the service of customers that the vehicle visits, while constraint (6) relates the lateness variables to these times. Constraint (7) relates the route duration with the start times of service 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 is difficult to linearize.

4 Methodology

In this section, we present solution methods for the DD-TD-TSP-STW problem. An exact specialized B&B algorithm (Section 4.1) and an ALNS heuristic with some SA features (Section 4.2). The two solution methods are tested and compared in Section 5.

We note that in a preliminary experiment, we formulated the problem as a constraint programming (CP) model and tried to solve it using the IBM ILOG CPLEX. However, this approach failed to solve even small instances of the problem.

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

B&B algorithms have been widely used to solve discrete optimization problems in the last 60 years (Land and Doig (1960), Little et al. (1963)). In this section, our specialized algorithm is described. First, an overview of the algorithm is given, and some of its components are discussed. Next, the more involved components of the algorithm are discussed in detail. Section 4.1.1 discusses branching. Section 4.1.2 discusses lower bound calculations. In Section 4.1.3, we present enhancements of the B&B framework that accelerate the running time of the algorithm.

Pseudocode of the algorithm is presented in Figure 1.

```
Decided = empty sequence    /* Sequence of customers already scheduled */
Undecided = the set of customers  /* the rest of the customers */
Incumbent = the sequence of Undecided sorted by EDD  /* best sequence found so far */
GlobalUB = CalcUB (Decided)  /* Calculate the upper bound for the decided sequence
Create a list L with an entry (Decided, Undecided, CalcLB (Decided), GlobalUB)
While L ≠ ∅
   Remove a node from L and store as (Decided, Undecided, LB, UB)
  If LB < GlobalUB
        CandCustomers = {i|i ∈ Undecided and i can be next in an optimal solution}
```

```
For i ∈ CandCustomers
    LBI = CalcLB((Decided, i))  /* calculate lower bound for the concatenated sequence */
    UBI = CalcUB((Decided, i)) /* calculate upper bound for the concatenated sequence */
    If LBI < GlobalUB
        Insert to L ((Decided, i), Undecided \ {i}, LBI, UBI)
        If UBI < GlobalUB
            GlobalUB = UBI
            Remaining = the sequence of customers  Undecided \ {i} sorted by EDD
            Incumbent = (Decided, i, Remaining)   /* store best found solution to return */
```

Figure 1: Pseudocode of the specialized B&B algorithm

In list L, we store all the open nodes of the B&B tree. Each entry of L has four components: a sequence of customers for which the route was already decided, a set that contains the rest of the customers, and the lower and upper bounds for the node. Clearly, any concatenation of the first component and some permutation of the customers of the second component is a feasible solution.

We initialize L with an empty $Decided$ sequence. The $Undecided$ set consists of all customers. The lower bound and the upper bound, i.e., value of a feasible solution, are calculated using the functions $CalcLB$ and $CalcUB$, respectively, as described below. The initial solution of the algorithm is obtained as a sequence of $Undecided$ sorted in a nondecreasing 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 that solution.

In each iteration of the process, one node is removed from L and stored as ($Decided, Undecided, LB, UB$). Next, $CandCustomers$, a list that contains all customers from the $Undecided$ set that can be next in the sequence of an optimal solution, is constructed. The algorithm branches on the items in this list. That is, new potential entries are constructed by removing a single customer from $CandCustomers$ 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 an approximate solution is desired, other branching and stopping criteria may apply.

The search tree is implemented using a priority queue with the lower bound of each entry as its key. Therefore, the next node to be processed at each iteration is the node whose lower bound is the smallest. 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 according to the EDD rule and returns to the depot. The entire path constitutes a valid route since the vehicle visits all 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.

4.1.1 Branching

In each iteration, the algorithm branches on the customers in the list $CandCustomers$. The creation of this list relies on the concept of a *local precedence relation*. This relation is established based on the values of the openings of the customers' service time windows, a_i.

For the sake of simplicity, we first explain this concept by assuming a single scenario problem and denote this scenario by k. Let us consider a node in the B&B tree. The vehicle has just finished serving customer j at time $t' = u_{jk} + s_{jk}$. Furthermore, consider customer m and customer h that have not yet been served. If customer m is visited immediately after j then $u_{mk} = max\{a_m, t' + t_{j,m,k}(t')\}$ is the time when the service starts at m and $u_{mk} + s_{mk}$ is the time when the service of m ends. Next, if $u_{mk} + s_{mk} + t_{m,h,k}(u_{mk} + s_{mk}) \leq a_h$, then every solution where h is visited immediately after j can be improved by inserting m between j and h. Consequently, h cannot be the customer visited after j in an optimal solution.

To see why h cannot be next to j in the sequence in an optimal solution, we show that visiting m immediately after j and then visiting h is dominating any solution where h is visited immediately after j. Recall that the objective function consists of the mean route duration and the sum of the penalties. Indeed, by inserting m before h, we do not postpone the service end time at h but reduce the route duration by saving the need to visit m after h. The penalties for h and all the customers that are visited subsequently are not increased while the penalty at m is minimized (in this branch of the tree).

We emphasize that the precedence relation between m and h is *local* in the sense that it is valid for particular customer j and departure time t'. Furthermore, it is established for a problem with a single scenario. However, in the multiscenario case, if at a particular time, the relation holds for all the scenarios, then there is no need to branch on customer h. The algorithm below returns the latest time $\bar{t}(j, m, h, k)$, where the precedence relation holds for all j, m and h in any scenario k. This algorithm is a preprocessing procedure that runs once before the B&B algorithm is launched.

$forall\ k \in \{1, \dots, K\}$
  $forall\ j \in \{0, \dots, n\}$
    $forall\ m \in \{1, \dots, n\} \backslash \{j\}$
      $forall\ h \in \{1, \dots, n\} \backslash \{j, m\}$
        $t' = a_j + s_{jk}$
        $while\ max\{a_m, t' + t_{j,m,k}(t')\} + s_{mk} + t_{m,h,k}\left(max\{a_m, t' + t_{j,m,k}(t')\}\right) \leq a_h$
          $t' = t' + 1$
        $\bar{t}(j, m, h, k) = t' - 1$

Figure 2: Pseudocode for calculating $\bar{t}(j, m, h, k)$

For each triplet of distinct customers and a scenario (j, m, h, k), we first set t' to be the minimal departure time from j, namely, $a_j + s_{jk}$. Assuming departure from j to m at time t', the service starting time at m is $max\{a_m, t' + t_{j,m,k}(t')\}$, the departure time from m is $max\{a_m, t' + t_{j,m,k}(t')\} + s_{mk}$, and the arrival time at h is $max\{a_m, t' + t_{j,m,k}(t')\} + s_{mk} + t_{m,h,k}(max\{a_m, t' + t_{j,m,k}(t')\})$. We iteratively increase t' as long as the arrival time at h is earlier than the opening of its service window at a_h and set $\bar{t}(j,m,h,k)$ to the largest value of t' for which the condition still holds. If the condition of the while loop never holds, the value of $\bar{t}(j,m,h,k)$ is smaller than the earliest possible departure time from j, $a_j + s_{jk}$, and thus, the precedence relation never holds in an actual node of the B&B tree.

Recall that travel times are time-dependent and therefore gradually increasing t' to find the value where the precedence relation holds cannot be avoided. However, the FIFO property assures us that if the precedence relation does not hold for t', then it does not hold for any $t'' > t'$.

When the algorithm branches on the next customer to be visited after completing service at j, if for each scenario k and some distinct pairs of customers m and h in the *Undecided* set, $u_{jk} + s_{jk} \leq \bar{t}(j,m,h,j)$, then customer h can be eliminated from $CandCustomers$. This procedure is described by the pseudocode in Figure 3.

$CandCustomers = \ Undecided$

$for\ h\ in\ CandCustomers$

  $for\ m\ in\ CandCustomers \setminus \{h\}$

    $if\ u_{jk} + s_{jk} \leq \bar{t}(j,m,h,k)$ for all scenarios k

      $CandCustomers = \ CandCustomers \setminus \{h\}$

      Exit loop

Figure 3: Pseudocode for building $CandCustomers$

4.1.2 Lower bound calculation

In this section, we present two lower bounds that are valid for our problem and can be used in the B&B procedure. The first is based on simple local considerations and is very easy to calculate, while the second is based on a solution of an assignment problem and is more computationally involved. Interestingly, none of these bounds is strictly tighter than the other, and thus, we calculate both bounds at each node of the B&B tree and use the larger bound.

Our first lower bound is obtained as the sum of a lower bound on the route duration and the expected sum of penalties. A lower bound on the route duration at each node can be calculated as follows: first, we define a lower bound on the service starting time at each unvisited customer ($j \in Undecided \setminus \{i\}$) at each scenario k, given the service starting time at the current customer, u_{ik}

$$\tilde{u}_{jk} = \max\{a_j, u_{ik} + s_{ik} + t_{i,j,k}(u_{ik} + s_{ik})\}$$

The above bound is valid due to the triangle inequality and the FIFO property. Consequently, the route duration in scenario k satisfies

$$T_k \geq \tilde{u}_{jk} + s_{jk} + t_{j,0,k}(\tilde{u}_{jk} + s_{jk}),$$

which is the route duration in the relaxed case when only customer j is yet to be served. Thus, a lower bound on the expected route duration is

$$\frac{1}{K}\sum_{k=1}^{K}\left(\tilde{u}_{jk}+s_{jk}+t_{j,0,k}(\tilde{u}_{jk}+s_{jk})\right).$$

Since this lower bound is valid for all $j \in Undecided \setminus \{i\}$), a lower bound is obtained by

$$\max\left\{\frac{1}{K}\sum_{k=1}^{K}\left(\tilde{u}_{jk}+s_{jk}+t_{j,0,k}(\tilde{u}_{jk}+s_{jk})\right): j \in Undecided \setminus \{i\}\right\}.$$

The lower bound for the sum of penalties is obtained as follows. Let us denote the mean, over all scenarios, of the sum of penalties accumulated up to customer i, at the current node, by Π. Then, a lower bound on the mean total penalties in the branch of the current node is given by

$$\Pi+\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in Undecided\setminus\{i\}} G_j\left(\max(0,\tilde{u}_{jk}-b_j)\right).$$

The lower bound for the objective function is the sum of the two lower bounds.

The second lower bound for the value of the sequence $(Decided, i)$ is calculated as the sum of the following: (1) the mean accumulated duration of the route up to customer i over all scenarios $\frac{1}{K}\sum_{k=1}^{K}(u_{ik}+s_{ik})$, (2) the mean accumulated penalty, denoted by Π, (3) the mean remaining service time over all scenarios $\frac{1}{K}\sum_{k=1}^{K}\sum_{j\in Undecided\setminus\{i\}} s_{jk}$, and (4) a lower bound for the mean sum of the remaining travel time and penalty costs. Next, we establish the latter. Thus, we present the following notations:

$\delta_{i,j,t',k}$	The minimal contribution of arriving at customer j, at time t' or later, after serving customer i in scenario k, to the objective function value.
t_{max}	An upper bound on the time of the last departure time from a customer

The values of $\delta_{i,j,t',k}$ are calculated as follows in a preprocessing procedure.

$$\delta_{i,j,t_{max}+1,k}=\infty \qquad i,j\in\{0,\dots,n\}, k\in\{1,\dots,K\}$$

$forall\ k \in \{1,\dots,K\}$

  $forall\ i \in \{0,\dots,n\}$

    $forall\ t'\ in\ (t_{max}, t_{max}-1, \dots a_i+s_{ik})$

      $forall\ j \in \{0,\dots,n\}\setminus\{i\}$

        $\sigma = max\{a_j, t'+t_{i,j,k}(t')\}$

        $\delta_{i,j,t',k} = min\{\sigma - t' + G_j(\sigma-b_j), \delta_{i,j,t'+1,k}\}$

Figure 4: Pseudocode for calculating $\delta_{i,j,t',k}$

First, $\delta_{i,j,t',k}$ are initialized to ∞ for each origin i, destination j and scenario k. Recall that the vehicle cannot start serving customer i prior to a_i and therefore cannot depart from i prior to $a_i + s_{ik}$. Next, for each scenario k and origin i, we iterate over all the relevant departure times, t', in descending order. For each destination $j \neq i$, we calculate the starting time, σ, at j, assuming departure from i at time t'. Based on σ, we calculate the minimal contribution $\delta_{i,j,t',k}$. This is obtained by recursively taking the minimum between the contribution to the objective function assuming departing exactly at t' and the value of the parameter calculated for $t' + 1$.

When processing a node $(Decided, i)$ in the B&B tree, the set of locations that have not yet been visited is $Undecided \cup \{0\} \setminus \{i\}$. This set is referred to as the $Destinations$. The customers in $Destinations$ can be reached from the set $Undecided$, referred to as the $Origins$ set. The cardinality of the two sets is the same. Recall that the starting time at the last customer i, in scenario k, u_{ik}, is also known at the node.

A lower bound for the remaining travel time and penalties at each particular node is equal to the value of a minimal cost assignment of members in the set $Origins$ to members in the set $Destinations$. The assignment costs are calculated based on $\delta_{i,j,t',k}$ as follows:

$$c_{mj} = \begin{cases} \dfrac{1}{K} \displaystyle\sum_{k \in \{1,\dots,K\}} \delta_{i,j,u_{ik}+s_{ik},k} & , m = i \\ \dfrac{1}{K} \displaystyle\sum_{k \in \{1,\dots,K\}} \delta_{m,j,\max(u_{ik}+s_{ik}+t_{i,m,k}(u_{ik}+s_{ik}),a_m)+s_{mk},k} & , m \neq i \end{cases},$$

where m indexes the origins and j indexes the destinations.

Note that c_{mj} is a lower bound on the contribution of the journey from j to m and the penalty cost at m at the current node. If the origin is i (the current customer at the node), then the departure time in scenario k is the starting time plus the service time, $u_{ik} + s_{ik}$. A lower bound on the departure time from other origins, m, is obtained when assuming that m is served immediately after i. The starting time at m cannot be earlier than the opening of its time window, a_m, as well as $u_{ik} + s_{ik} + t_{i,m,k}(u_{ik} + s_{ik})$, which is the departure time from i plus the travel time. The departure time from m occurs s_{mk} units of time after this starting time. Finally, we note that our lower bound can be further tightened by setting $c_{i0} = \infty$ to eliminate returning to the depot directly from the current node if there are still unvisited customers.

The idea of using an assignment problem as a relaxation for TSP has been widely used in the literature (see Balas and Toth, 1983). However, for the DD-TD-TSP-STW problem, the cost of each leg in the relaxation depends on the entire sequence, which is unknown at each node of the B&B tree. Therefore, we use a lower bound of this cost and tighten it at the nodes.

4.1.3 Algorithmic enhancements

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

Recall that $(Decided, i)$ is a sequence of visited customers that ends with the current customer i. Let C_i represent the average penalties over all scenarios accumulated up to the arrival at customer i when following the sequence $(Decided, i)$. Let $Decided'$ represent an alternative sequence to $Decided$ that contains the same customers but not in the same order. u'_{ik} and C'_i are the arrival time at customer i and the accumulated penalty in the sequence $(Decided', i)$, respectively.

Lemma 1: If $C_i \leq C'_i$ and $u_{ik} \leq u'_{ik}$ $\forall k = 1, \ldots, K$, then the sequence $(Decided, i)$ weakly dominates the sequence $(Decided', i)$. That is, there exists an optimal sequence that does not contain $(Decided', i)$ as a prefix.

Proof: We consider two valid solutions for the DD-TD-TSP-STW. The first solution is the sequence *(Decided,* i*, S)*, where S represents a permutation of the set $Undecided \setminus \{i\}$. The second solution is the sequence *(*$Decided'$*,* i*, S)*. We prove the Lemma by showing that the cost of the sequence *(Decided,* i*, S)* is no greater than the cost of the sequence *(*$Decided'$*,* i*, S)*.

Let j and l denote the first and last customers in *S*. For customer j, $u_{jk} = \max(a_j, u_{ik} + s_{ik} + t_{i,j,k}(u_{ik} + s_{ik}))$ and $u'_{jk} = \max(a_j, u'_{ik} + s_{ik} + t_{i,j,k}(u'_{ik} + s_{ik}))$. 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, $u_{ik} + s_{ik} + t_{i,j,k}(u_{ik} + s_{ik}) \leq u'_{ik} + s_{ik} + t_{i,j,k}(u'_{ik} + s_{ik})$; thus, $u_{jk} \leq u'_{jk}$ $\forall k = 1, \ldots, K$. Clearly, a similar analysis can be performed for all customers in *S*. Therefore, for customer l, $u_{lk} \leq u'_{lk}$ $\forall k = 1, \ldots, K$.

The total duration of the route is the time when the vehicle returns to the depot after serving the last customer. For the sequence *(Decided,* i*, S)*, the duration is simply $u_{lk} + s_{lk} + t_{l,0,k}(u_{lk} + s_{lk})$. For the sequence *(*$Decided'$*,* i*, S)*, the total duration is $u'_{lk} + s_{lk} + t_{l,0,k}(u'_{lk} + s_{lk})$. Since $u_{lk} + s_{lk} \leq u'_{lk} + s_{lk}$ $\forall k = 1, \ldots, K$, the FIFO property ensures that $u_{lk} + s_{lk} + t_{l,0,k}(u_{lk} + s_{lk}) \leq u'_{lk} + s_{lk} + t_{l,0,k}(u'_{lk} + s_{lk})$ $\forall k = 1, \ldots, K$. That is, the average duration of the route of the sequence *(Decided,* i*, S)* is no greater than the duration of the sequence *(*$Decided'$*,* i*, S)*.

Regarding the penalties, recall that for each customer $p \in S$, $u_{pk} \leq u'_{pk}$ $\forall k = 1, \ldots, K$ and that $G_i(x)$ is nondecreasing in x. Thus, for each customer p, the penalty incurred when following the sequence *(Decided,* i*, S)* is no greater than the penalty incurred when following the sequence *(*$Decided'$*,* i*, S)*. Since $C_i \leq C'_i$, we conclude that the total penalties when following the sequence *(Decided,* i*, S)* are no greater than the penalties when following the sequence *(*$Decided'$*,* i*, S)*.Since the cost is the sum of the average route duration and the average penalties, the claim is proven. Therefore, there exists an optimal route that does not begin with $(Decided', i)$. ■

The practical implication of Lemma 1 is that branches in the B&B tree that contain the sequence $(Decided', i)$ 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 sequences. 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 $[\{Decided\}, i]$. In the entry $A[\{Decided\}, i]$, we store the vector of arrival times at all the scenarios at i, denoted by $A[\{Decided\}, i].u_k$, and the average penalties accumulated up to i over all scenarios by $A[\{Decided\}, i].\mathrm{C}$.

For every sequence *(*$Decided, i$*)* encountered during the B&B, we check whether an equivalent sequence, $[\{Decided\}, i]$, exists in the cache. If it does exist, and *(*$Decided, i$*)* is not dominated by the existing entry, or if it does not exist, we calculate its lower bound and compare it to the global upper bound. If the lower bound is smaller than the upper bound, we update the cache (and create a new entry if necessary) and a new node in the B&B tree. A new sequence $(Decided, i)$ is said to be *dominated* by an existing equivalent sequence if the arrival time at the last customer, i, for any scenario is not earlier than the arrival time in the existing sequence and its average penalty is not smaller.

By applying this domination rule, the B&B tree is reduced significantly. In some preliminary experiments with 18 customers, we observed an approximately six-fold reduction in the running time. The effect seems to increase with the dimension of the problem. In terms of memory usage, the space needed for the hash table is negligible compared to the reduction in the memory required for the B&B tree.

4.2 ALNS heuristic for the DD-TD-TSP-STW

In this section, we present the ALNS heuristic (See Ropke and Pisinger (2006a), Ropke and Pisinger (2006b), Pisinger and Ropke (2007)) with SA features (See Kirkpatrick et al. (1983)) that can produce high-quality solutions for the problem in a relatively short time and scales better than the B&B framework applied earlier. The heuristic is comprised of two stages: a *construction* stage in which an initial solution is created, and an *improvement* stage.

The initial solution is constructed using an insertion algorithm. Starting with an empty route, the algorithm iteratively selects the best possible insertion until all customers are routed. The best insertion is found by checking all the yet unrouted customers and all the possible positions in the existing routes. The customer whose insertion at its best position minimizes the average contribution over all scenarios to the value of the objective function is inserted.

In the ALNS framework, an initial solution is gradually improved by applying various removal and insertion heuristics iteratively. In each iteration, one removal and one insertion operator are randomly selected based on their current weights. These weights are updated periodically as the search progresses based on the performance of the operators. If a newly created solution maintains an acceptation criterion, it is accepted; otherwise, the current solution remains the same. The algorithm is terminated once a predefined stopping criterion is met.

The iterative improvement stage of our algorithm implements a variant of the ALNS framework and incorporates some elements of SA. Next, we describe our proposed removal and insertion operators as well as a selection rule for their use. In addition, we present a procedure for updating their weights. For $N \in \{N_{min}, \dots, N_{max}\}$, we define the N-Removal operation as follows: randomly remove N customers from the solution. At each iteration, the N-Removal operator is applied with probability p_N for $N \in \{N_{min}, \dots, N_{max}\}$. The insertion of the removed customers is performed sequentially as follows: a random permutation by which the customers are inserted is created. Next, and according to their order, each customer is inserted in the position that minimizes the cost of the insertion.

We now describe the process of updating the probability vector $\mathbf{p}$. This process is carried out every Ω iterations to increase the probability of selecting more successful removals in the next block of Ω iterations. Let C_N denote the number of times where the N-removal operator was applied during the last Ω iterations, and S_N denotes the number of successful removals. A *successful removal* is defined as a removal that leads to moving from the current solution to a solution with lower cost. We calculate the success rate of the N-Removal operator as

$$R_N = \begin{cases} \frac{S_N}{C_N}, & C_N > 0 \\ 0, & C_N = 0 \end{cases}$$

At the end of each block of Ω iterations, the probability vector is updated as follows:

$$p_N \leftarrow \alpha \frac{R_N}{\sum_{i=N_{min}}^{N_{max}} R_i} + (1-\alpha)p_N, \qquad \forall N \in N_{min}, \dots, N_{max}$$

where $\alpha \in (0,1]$ is an exponential smoothing coefficient. However, $\mathbf{p}$ is updated only if $\sum_{i=N_{min}}^{N_{max}} R_i > 0$.

In order to further increase the chance of escaping local optima, the generated solution is evaluated using a simulated annealing acceptance criterion. Let T represent the current temperature of the search process, and e (resp., e') represents the value of the current (resp., candidate) solution. The probability of moving to the candidate solution is $\min\left(1, \exp\left(-\frac{e'-e}{T}\right)\right)$. The initial temperature is obtained as an increasing function, $Z(x)$, of the value of the initial solution, i.e., the solution generated in the construction stage.

We implemented an adaptive cooling procedure. ω represents the number of iterations carried out between two consecutive updates of T. Next, in each update, T is reduced by Δ units only if a new best-known solution has been encountered since the previous update. Δ is selected proportionally to the initial temperature. The proposed cooling procedure deviates from the standard implementation of SA. Its merits are in diversifying the search, even at later stages of the procedure, if the best-known solution cannot be improved during a large number of iterations.

Note that since the SA allows movements to solutions that are inferior to the current solution, some acceptable moves are not a result of successful removal, and some successful removals do not lead to a new best-known solution.

5 Numerical experiments

In this section, we present our 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 an exact solution method for instances with up to 30 customers. Note that in most practical settings, a technician serves a much smaller number of customers in a working day.
2. Evaluating the performance of our ALNS heuristic in terms of both optimality gaps and running time.
3. Demonstrating the merits of considering stochasticity and time dependency in the optimization process in real-life settings.
4. Validating our scenario-based (data-driven) optimization approach as an effective method to address the randomness of real-life situations of field service operations.

To meet goals 3 and 4, we divided the scenarios into training dataset and test dataset. The training dataset is used as the set of scenarios on which the problems are solved while the test set is used at a later stage to evaluate the quality of the solutions when applied for unknown future scenarios.

In Section 5.1, we describe the dataset used in our experiments. In Sections 5.2 and 5.3, we report the performance analysis of the B&B and ALNS algorithms, respectively. In Section 5.4, we compare the outcomes of the stochastic and time-dependent model with those of simpler models. Finally, in Section 5.5, we validate the scenario-based approach.

5.1 Problem instances

Problem instances are constructed based on a set of 57 locations in central Israel where each problem instance involves a subset of these locations. Time-dependent travel times between these 57 locations were gathered from Google Maps for a single working day. We refer to these travel times as the *nominal time-dependent* travel times. We denote the nominal travel time between locations i and j starting at time t' by $\tilde{t}_{ij}(t')$. In addition, for a subset of 19 representative locations in the set, time-dependent travel time data were sampled in real time during 60 working days between March and June 2017, every 90 minutes. These travel times are referred to as *scenario time-dependent* travel times, denoted by $t_{ijk}(t')$. Note that these times represent the actual travel times affected by the traffic conditions during this period.

We estimated the travel time at each particular minute $t' \in [t'_1, t'_2]$ during the day, where t'_1 and t'_2 are two consecutive sampling times, in the representative set of locations by the following interpolation formula:

$$t_{ijk}(t') = \frac{1}{90}\left[t_{ijk}(t'_1)(t'_2 - t') + t_{ijk}(t'_2)(t' - t'_1)\right].$$

These values were rounded to integer minutes and corrected for very few minor violations of the FIFO property.

The scenario time-dependent travel times between the rest of the 57 locations were estimated based on the data gathered from the subset of the 19 locations as well as their nominal time-dependent data. For each pair of locations (i, j) within the 57 locations, on each day (for a total of 60 days) and for each time period (of one minute), we find another pair, (i', j'). i' (j') represents the location in the subset of the 19 locations whose travel time to i (j) is the shortest. Next, we estimate the time-dependent travel time between locations i and j as follows:

$$t_{ijk}(t') = \tilde{t}_{ij}(t') \cdot \frac{t_{i',j',k}(t')}{\tilde{t}_{i',j'}(t')}.$$

That is, we adjust the nominal time-dependent travel time by the ratio that represents the temporal traffic congestion along a similar route.

We note that this procedure constitutes a reasonable method for obtaining the data required for a solution of a stochastic problem in the field service domain. Recall that in field service, each working day involves new customers, and therefore gathering the required travel time data for stochastic modeling may be very challenging. However, service zones usually do not change. Therefore, the option for gathering (each day and continuously) the time-dependent travel times between a **subset** of representative (central) locations in the road network and using this detailed data to approximate the stochasticity in travel times of other (and new) locations seems both practical and tractable.

We created problem instances with 12, 24, 30 and 36 customers. Twenty instances of each size were created for a total of 80 instances. Service times were randomly generated so that the total time spent on service was approximately 6 hours a day for all instances. The time windows considered were three nonoverlapping time slots of three hours each (over a planning horizon of 9 hours), while the customers were equally divided among them. These time windows represent a common practice in the field service industry when all appointments are scheduled in advance by the contact center, and the workload is balanced.

It is natural to assume that the penalty function $G_i(x)$ is strictly convex to reflect the increasing marginal penalty for each additional time unit of delay. Such a function favors several small delays at customers' locations rather than a large delay at a single customer's location and thus balances the service levels for the customers. We use $G_i(x) = x^2$. However, we mention that our solution methods can be applied to any nondecreasing penalty function.

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 by simulation the quality of the previously found solutions based on new data. This process represents a real-life situation in which planning is based on past scenarios, but the results are applied to future scenarios. We note that using longer historical data may render the training set nonrepresentative as the traffic conditions change over time. We demonstrate below that 40 scenarios are enough to capture the stochasticity and create solutions that perform well over the 20 test set scenarios. The advantages of our scenario-based optimization over simpler models are already statistically significant when applied to the 20 scenario test set.

The algorithms were implemented as single-threaded applications in Python 2.7 and tested on an Intel i9-9900K, 3.6 GHz desktop with 64 GB RAM running Ubuntu Linux 18.04.

5.2 B&B algorithm

We solved the instances with 12, 24 and 30 customers using our 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 each instance size. 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 (seconds) of the B&B algorithm until optimality is proven.

n	Average	Median	Minimal	Maximal
12	1	1	1	2
24	594	350	42	3,493
30	77,319	48,640	714	325,439

Table 2: Running time (seconds) until the optimal solution was encountered

n	Average	Median	Minimal	Maximal
12	1	1	1	2
24	477	316	42	3,493
30	61,265	48,640	665	219,000

Instances with 12 customers are solved within 2 seconds. This demonstrates the applicability of the algorithm to be used as a subroutine for solving multivehicle routing problems when the number of served customers in each route is small. Instances with 24 customers are all solved to optimality within one hour of running time. However, optimal or near-optimal solutions are encountered after several minutes. The instances with 30 customers are computationally intensive and require a solution time of many hours. In these instances, the time to reach the optimal solution is also too long for operational settings and were solved to explore the tractability boundaries of the algorithm and to provide solved benchmark instances.

Recall that our Python implementation is single-threaded. It is likely that more cautious coding could result in a significant running time improvement, but this is outside the scope of this study. However, much larger instances are unlikely to be solved with reasonable CPU and memory resources even with a better implementation of this algorithm.

In this context, it is relevant to present the performance of an adapted version of our algorithm designed to address a time-dependent deterministic TSP with soft time windows (see in detail Section 5.4 later in this section). That is, we used the same algorithm but with a single scenario obtained from the average over the 40 training scenarios. Table 3 presents the total running time of such an algorithm for adapted instances with 24, 30 and 36 customers.

Table 3: Total running time for time-dependent problems

n	Average	Median	Minimal	Maximal
24	2	1	1	6
30	25	8	6	229
36	274	83	54	2,692

It is quite clear that our algorithm can efficiently solve time-dependent instances of the TSP with time windows in reasonable time. The improvement here is not only because the algorithm does not need to process multiple scenarios but also because of stronger bounds that can be obtained since they are established based on the "average scenario" rather than on the best case scenario.

Recall that whereas most recent studies assumed hard time windows when time dependency was considered, this study addresses the less constraining but computationally harder case of soft time windows. We believe that dealing with this case has unique merit in field service. In this industry, the booking of a time window at a customer location is often done heuristically. Thus, the situation in which a technician cannot meet the customers' time window on certain days is not rare.

5.3 ALNS algorithm

We carried out ten ALNS replications for each problem instance. Each replication included n^3 iterations. Preliminary numerical experiments showed that the algorithm performs well for $\Omega = \omega = n^2$ and exponential smoothing factor $\alpha = 0.4$. In addition, $Z(x) = 0.1x$ and $\Delta = \frac{Z(x)}{n^3/\omega}$. Moreover, $N_{min} = \left\lfloor \frac{n}{5} \right\rfloor$ and $N_{max} = \left\lfloor \frac{n}{2} \right\rfloor$. Finally, we used initial probabilities $p_N = \frac{1}{N_{max} - N_{min} + 1}$ for $N \in \{N_{min}, \dots, N_{max}\}$.

Table 4 presents the statistics related to the performance of the ALNS algorithm. In the first four columns, we present the average, median, minimal and maximal optimality gaps achieved for the instances with 24 and 30 customers. These were calculated based on all 200 runs of each instance size (20 instances × 10 replications). The optimality gap was calculated as $100 \times \frac{ALNS - OPT}{ALNS}$, where $ALNS$ represents the value of the solution and OPT represents the value of the optimal solution obtained by our B&B algorithm. In the right-most column, we present the average running time of the ALNS algorithm for each instance size.

Table 4: Optimality gaps and average running time of the ALNS algorithm

	Optimality gaps				Average running times (seconds)
n	Average	Median	Minimal	Maximal	
24	0.00%	0.00%	0.00%	0.18%	177
30	0.77%	0.00%	0.00%	28.21%	1,109
36*	0.23%	0.00%	0.00%	1.79%	4,038

* Gaps compared to best known (minimal of 10 ALNS runs)

As expected, the ALNS algorithm scales better than the B&B algorithm and can produce high-quality solutions in a reasonable time. We note that the best solution out of 10 replications of the ALNS is almost always (with two expectations out of 40) identical to the optimal solution. For the instances with 36 customers, we could not compute an optimal solution; thus, we calculated the gaps relative to the best solution obtained in the 10 runs. Since it is easy to run the replications in parallel on a multicore CPU, a heuristic that is based on multiple runs of the ALNS seems very attractive.

5.4 The added value of considering time dependency and stochasticity

In this section, we evaluate the merits achieved by considering time dependency and stochasticity in the optimization process rather than following simpler route planning approaches. As we demonstrated in Table 3, the deterministic time-dependent version of the problem is much easier to solve. Thus, the question of whether the extra effort required to solve a time-dependent multiscenario model pays off needs to be answered.

Therefore, we solved three simpler versions of the problem: (1) a stochastic time-independent version (denoted by DD-TSP-STW). (2) a time-dependent deterministic version (denoted by TD-TSP-STW). (3) deterministic time-independent version (denoted by TSP-STW). As input for these three special cases of the DD-TD-TSP-STW, we used the relevant averages of the time-dependent and scenario data that were used for the full model.

Next, we applied the optimal solution obtained by each of the methods on each of the 40 training and 20 test scenarios. The average gap between the value of these solutions and the solution obtained from the full DD-TD-TSP-STW model were compared for both the training and test sets. Recall that the solutions were obtained without considering the test data because it represents unknown future scenarios, whereas the training set represents historical travel and service time data. Applying the solution to the test scenarios is a simulation of a real-life setting in which decisions are based on past data but applied to the future.

The average gap of each model represents the loss incurred due to neglecting one or two aspects of the DD-TD-TSP-STW. The average gaps for the three models of the problem are presented in Table 5 for the 20 instances with 24 customers.

Table 5: Average gaps of the special cases of the DD-TD-TSP-STW

Dataset	DD-TSP-STW	TD-TSP-STW	TSP-STW
Training	3.8%	3.7%	8.4%
Test	3.3%	3.6%	7.4%

The average loss incurred by solving the TSP-STW is estimated by our test dataset to be 7.4%. Both stochasticity and time dependency are proven to be important for the optimization process. Neglecting the time-dependent aspect of the DD-TD-TSP-STW may result in a loss of up to 3.3% of the total cost. Neglecting the stochastic aspect of the problem may result in a loss of up to 3.6%. The advantage of the full model over the TD-TSP-STW and the TSP-STW is significant in a one-sided paired t-test with p-value<0.01. The advantage of the full model over the DD-TSP-STW is marginally significant with p-value $= 0.06$. Note that solving the DD-TSP-STW requires similar computational effort to solving the full model; therefore, the latter should be preferred if the time limit allows it. However, in cases where the time limit is tight, solving the TD-TSP-STW or using the ALNS heuristic may be considered as an approximate solution to the full model.

5.5 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 were divided into a training dataset of 40 scenarios that was used as input for all the optimization algorithms and a test dataset of 20 scenarios that was left aside and used to evaluate the solutions.

We demonstrated in the results reported in Table 5 that the merits of considering stochasticity and time dependence are similar for both the training and test scenarios, which is a first indication for the applicability of our approach and for our claim that 40 scenarios constitute a sufficiently large training set.

Next, we aim to verify that our data-driven solution method obtains good routing solutions for each scenario in the test dataset. Thus, we optimized the time-dependent deterministic routing problem of each scenario in the test dataset separately and with hindsight. This solution is the best solution that a planner with complete information could achieve and apply for each particular scenario. Therefore, it is a valid lower bound on the value of the solution that can be obtained using any optimization procedure.

For each test scenario, we calculate the values of the solutions of the DD-TD-TSP-STW using the time-dependent data of that scenario. Finally, we calculate the gap between the performance of the best solution with hindsight and the performance of our model. We considered 20 instances with 24 customers and all 20 scenarios in the test data set and thus calculated 400 relative gaps. The average relative gap was 4.15% (with a 95% confidence interval of 3.4%-4.9%). This result implies that the optimal solution of the DD-TD-TSP-STW with 40 scenarios cannot be very far from the best possible solution of the time-dependent version of the problem.

6 Conclusions

In this paper, we introduced a model that captures the stochastic and time-dependent nature of the field service routing and scheduling tasks. In particular, our model considers the intricate interdependencies between travel times and service times by optimizing over a set of scenarios that are based on historical data. We devised an exact solution method as well as a more scalable successful ALNS heuristic.

Through a numerical study, we demonstrated that our model leads to robust solutions that are, on average, better than the solutions of the simpler models that are common in practice and are not considerably inferior to optimal solutions with hindsight. An insight obtained from these numerical experiments is that it is worth exerting the effort required to solve our more involved model instead of models that abstract the stochasticity and/or the time dependency.

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 when the number of customers served by each vehicle is not large.

Acknowledgment

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

The first author is partially supported by a scholarship from the Shlomo-Shmeltzer institute for smart transportation in Tel-Aviv university.

References

1. Arigliano, A., Ghiani, G., Grieco, A., & Guerriero, E. (2015). Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm, Technical Report, *Optimization Online.*
2. Arigliano, A., Calogiurim, T., Ghiani, G., & Guerriero, E. (2018). A branch-and-bound algorithm for the time-dependent travelling salesman problem, *Networks*, 72(3), 382-392.
3. Arigliano, A., Ghiani, G., Grieco, A., Guerriero, E., & Plana, I. (2019). Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm, *Discrete Applied Mathematics,* 261, 28-39.
4. Balas, E., & Toth, P. (1983). Branch and Bound Methods for the Traveling Salesman Problem, Technical Report, Carnegie-MelIon university, Pittsburgh, Pennsylvania, Management sciences research group.

5. Binart, S., Dejax, P., Gendreau, M., & Semet F. (2016). A 2-stage method for a field service routing problem with stochastic travel and service times, *Computers & Operations Research*, 65, 64-75.
6. Boland, N., Hewitt, M., Vu, D.M., & Savelasbergh, M. (2017). Solving the traveling salesman problem with time windows through dynamically generated time-expanded networks, *14th Conference on Integration of Artificial Intelligence and Operations Research Techniques: volume lecture notes in computer science.*
7. Çimen, M., & Soysal, M. (2017). Time-dependent green vehicle routing problem with stochastic vehicle speeds: An approximate dynamic programming algorithm, *Transportation Research Part D*, 54, 82-98.
8. Clarke, G., & Wright J. (1964). Scheduling of vehicles from a central depot to a number of Delivery Points, *Operations Research*, 12(4), 568-581.
9. Cordeau, J.-F., Ghiani, G., & Guerriero, E. (2014). Analysis and Branch-and-Cut Algorithm for the Time-Dependent Travelling Salesman Problem, *Transportation science*, 48(1), 46-58.
10. Delage, E. (2010). Re-optimization of technician tours in dynamic environments with stochastic service time, Master's thesis, Ecole des Mines de Nantes.
11. Duan, Z., Sun, S., Sun, S. & Li W. (2015). Stochastic time-dependent vehicle routing problem: Mathematical models and ant colony algorithm, *Advances in Mechanical Engineering*, 7(11), 1-16.
12. Ehmke, J.F. & Mattfeld, D.C. (2012). Vehicle routing for attended home delivery in city Logistics, *Procedia-Social and Behavioral Sciences*, 39, 622-632.
13. Ehmke, J.F., Campbell, A.M., & Urban, T. L. (2015). Ensuring service levels in routing problems with time windows and stochastic travel times, *European Journal of Operational Research*, 240, 539-550.
14. Errico, F., Desaulniers, G., Gendreau, M., Rei, W. & Rousseau, L.-M. (2013). The vehicle routing problem with hard time windows and stochastic service times, *Les Cahiers du GERAD*, G-2013-45.
15. Errico, F., Desaulniers, G., Gendreau, M., Rei, W. & Rousseau, L.-M. (2016). A priori optimization with recourse for the vehicle routing problem with hard time windows and stochastic service times, *European Journal of Operational Research*, 249, *55-66*.
16. Fleischmann, B., Gietz, M., & Gnutzmann, S. (2004). Time-Varying Travel Times in Vehicle Routing, *Transportation science*, 38(2), 160-173.
17. Gendreau, M., Jabali, O. & Rei, W. (2014). Stochastic Vehicle Routing Problems. In: Toth P. and Vigo D. (Eds.). *Vehicle Routing: Problems, Methods and Applications, Second Edition* (pp. 213-239), MOS-SIAM Series on Optimization, Philadelphia.
18. Gendreau, M., Ghiani, G., & Guerriero. E. (2015). Time-dependent routing problems: A review, *Computers & Operations Research*, 64, 189-197.
19. Haghani, A., & Jung, S., (2005). A dynamic vehicle routing problem with time-dependent travel times, *Computers & Operations Research*, 32(11), 2959-2986.
20. Hill, A.V. & Benton, W.C., (1992). Modeling intra-city time-dependent travel speeds for vehicle scheduling problems, *Journal of the Operations Research Society*, 43(4), 343–351.

21. Ichoua, S., Gendreau, M., & Potvin, J.-Y. (2003). Vehicle dispatching with time-dependent travel times. *European Journal of Operational Research*, 144(2), 379–396.
22. Jabali, O., Van Woensel, T., De Kok, A.G., Lecluyse, C. & Permenas, H. (2009). Time-dependent vehicle routing subject to time delay perturbations, *IIE Transactions*, 41(12), 1049 – 1066.
23. Kenyon, A. S. & Morton, D. P. (2003). Stochastic vehicle routing with random travel times, *Transportation Science*, 37, 69–82.
24. Kirkpatrick, S., Gelatt Jr, C. D., & Vecchi, M. P.(1983). Optimization by Simulated Annealing, *Science*, 220 (4598), 671–680
25. Land, H., & Doig, A. G. (1960). An automatic method of solving discrete programming problems, *Econometrica, 28(3), 497-520.*
26. Laporte, G., Louveaux, F. V. & Mercure, H. (1992). The vehicle routing problem with stochastic travel times, *Transportation Science*, 26(3), 161–170.
27. Lecluyse, C., Van Woensel, T., & Peremans, H. (2009). Vehicle routing with stochastic time-dependent travel times, *4OR*, 7, 363-377.
28. Lei, H., Laporte, G., & Gou, B. (2012). A generalized variable neighborhood search heuristic for the capacitated vehicle routing problem with stochastic service times, *TOP*, 20, 99-118.
29. Li, X., Tian, P. & Leung, S. C. H. (2010). Vehicle routing problems with time windows and stochastic travel and service times: models and algorithm, *International Journal of Production Economics*, 125, 137-145.
30. Little, J. D. C., Murty, K. G., Sweeny, D. W. & Karel, C. (1963). An algorithm for the traveling salesman problem, *Operations Research*, 11(6), 972 – 989.
31. Malandraki, C., (1989). Time dependent vehicle routing problems: formulations, solution algorithms and computations experiments, Ph.D. Dissertation, Northwestern University, Evanston, III.
32. Malandraki, C. & Daskin, M.S., (1992). Time dependent vehicle routing problems: formulations, properties and heuristic algorithms, *Transportation Science*, 26(3), 185–200.
33. Miller, C.E., Tucker, A.W. & Zemlin, R.A. (1960). Integer programming formulations and traveling salesman problems, *Journal of the Association for computing machinery*, 7(4), 326-329.
34. Montero, A., Mendez-Diaz. I., & Miranda-Bront, J.J. (2017). An integer programming approach for the time-dependent traveling salesman problem with time windows, *Computers & Operations Research*, 88, 280-289.
35. Nahum, O. E., & Hadas, Y. (2009). Developing a model for the stochastic time-dependent vehicle-routing problem, *Proc. IEEE. International Conference on Computers & Industrial Engineering*, 118-123.
36. Pisinger, D., & Ropke, S. (2007). A general heuristic for vehicle routing problems, *Computers & Operations Research*, 34(8), 2403-2435.
37. Ropke, S., & Pisinger, D. (2006). A unified heuristic for a large class of vehicle routing problems with backhauls, *European Journal of Operational Research*, 171(3), 750-775.
38. Ropke, S., & Pisinger, D. (2006). An Adaptive Large Neighborhood Search Heuristic for the Pickup and Delivery Problem with Time Windows, *Transportation Science*, 40(4), 455–472.

39. Souyris, S., Cortés, C. E., Ordóñez, F. & Weintraub, A. (2013). A robust optimization approach to dispatching technicians under stochastic service times, *Optimization Letters*, 7(7), 1549-1568.
40. Tas, D., Dellaert, N., Van Woensel, T. & de Kok, T. (2012). Vehicle routing problem with stochastic travel times including soft time windows and service costs, *Computers & Operations Research*, 40, 214-224.
41. Tas, D., Gendreau, M., Dellaert, N., Van Woensel, T. & de Kok, T. (2013). Vehicle routing with soft time windows and stochastic travel times: A column generation and branch-and-price solution approach, *European Journal of Operational Research*, 236(3), 789-799.
42. Tas, D., Dellaert, N., Van Woensel, T. & de Kok, T. (2014). The time-dependent vehicle routing problem with soft time windows and stochastic travel times, *Transportation Research Part C: Emerging Technologies*, 48, 66-83.
43. Verbeeck, C., Sörensen, K., Aghezzaf, E. & Vansteenwegen, P. (2014). A fast solution method for the time-dependent orienteering problem, *European Journal of Operational Research*, 236(2), 419-432.
44. Verbeeck, C., Vansteenwegen, P., & Aghezzaf, E.-H. (2016). Solving the stochastic time-dependent orienteering problem with time windows, *European Journal of Operational Research*, 255(3), 699-718.
45. Vu, D.M., Hewitt, M., Boland, N., & Savelasbergh, M. (2018). Solving Time Dependent Traveling Salesman Problems with Time Windows, Technical Report, *Optimization Online.*