Solving Time-Dependent Traveling Salesman Problem with Time Windows under Generic Time-Dependent Travel Cost

Duc Minh Vu [1][0000−0001−5882−3868], Mike Hewitt[2][0000−0002−9786−677X], and Duc Duy Vu[3][0000−0002−2657−0445]

[1] Phenikaa University & ORLab, Yen Nghia, Ha Dong, Vietnam
Corresponding Author: minh.vuduc@phenikaa-uni.edu.vn
[2] Loyola University Chicago, US
[3] University of Michigan - Flint, US

Abstract. In this paper, we present formulations and an exact method to solve the Time Dependent Traveling Salesman Problem with Time Window (TD-TSPTW) under a generic travel cost function where waiting is allowed. A particular case in which the travel cost is a non-decreasing function has been addressed recently. With that assumption, because of both First-In-First-Out property of the travel time function and the non-decreasing property of the travel cost function, we can ignore the possibility of waiting. However, for generic travel cost functions, waiting after visiting some locations can be part of optimal solutions. To handle the general case, we introduce new lower-bound formulations that allow us to ensure the existence of optimal solutions. We adapt the existing algorithm for TD-TSPTW with non-decreasing travel costs to solve the TD-TSPTW with generic travel costs. In the experiment, we evaluate the strength of the proposed lower bound formulations and algorithm by applying them to solve the TD-TSPTW with the total travel time objective. The results indicate that the proposed algorithm is competitive with and even outperforms the state-of-art solver in various benchmark instances.

Keywords: time-dependent travel time · time-dependent travel cost · traveling salesman problem · dynamic discretization discovery

1 Problem Formulation

The TD-TSPTW is presented mathematically as follows. We let (N, A) denote a directed graph, wherein the node set $N = \{0, 1, 2, ..., n\}$ includes the depot (node 0) as well as the set of locations (node $1, \ldots, n$) that must be visited. Associated with each location $i \in N$ is a time window $[e_i, l_i]$ during which the location must be visited. A tourist must visit the city i within its time window. Note that the tourist may arrive at city $i \in N \setminus \{0\}$ before e_i, in which case he must wait until the time window opens. Because of waiting, he does not need to depart immediately after his visit. The time window associated with the depot

means that the tour departs the depot at the time of at least e_0 and must return to the depot no later than l_0.

We define $A \subseteq N \times N$ as the set of arcs that present travel between locations in N. Associated with each arc $(i, j) \in A$ and time, t, at which travel can begin on the arc, is a travel time $\tau_{ij}(t)$. The FIFO property implies that for each arc $(i, j) \in A$ and times t, t' wherein $t \leq t'$, we must have $t + \tau_{ij}(t) \leq t' + \tau_{ij}(t')$. Thus, formally, the vehicle departs from node 0 at time $t \geq e_0$, arrives at each city $j \in N$ exactly once within its time window $[e_j, l_j]$ by traveling on arcs in A, and then returns to node 0 at time $t' \leq l_0$.

We formulate this tour as an integer program defined on a time-expanded network, $\mathcal{D} = (\mathcal{N}, \mathcal{A})$, with node set $\mathcal{N}$ and arc set $\mathcal{A}$. This formulation is based on the presumption that time may be discretized into a finite set of integer time points. As such, for each node $i \in N, t \in [e_i, l_0]$, $\mathcal{N}$ contains the node (i, t). $\mathcal{A}$ contains travel arcs of the form $((i, t), (j, t'))$ wherein $i \neq j, (i, j) \in A$, $t \geq e_i, t' = \max\{e_j, t + \tau_{ij}(t)\}$ (the vehicle cannot visit early), and $t' \leq l_j$ (the vehicle cannot arrive late). Note that, since waiting is allowed, we can depart from i to j at a time later than the time windows $[e_i, l_i]$ of location i. Finally, $\mathcal{A}$ contains arcs of the form $((i, t), (i, t + 1))$ presenting waiting at location i.

To formulate the integer program, for each arc $a = ((i, t), (j, t')) \in \mathcal{A}$, binary variable x_a represents whether the vehicle travels/waits along that arc. Let $c_a = c_{ij}(t)$ be the non-negative travel cost associated with arc a. If $i = j$ and $t' = t+1$, $c_{ii}(t)$ represents the waiting cost for one unit of time period, from time period t to time period $t+1$. Notations $\delta^+(i, t)$ and $\delta^-(i, t)$ present the set of incoming arcs and the set of outgoing arcs at the node $(i, t) \in \mathcal{N}$. The following formulation solves the TD-TSPTW with a generic travel cost function:

$$z = \text{minimize} \sum_{a \in \mathcal{A}} c_a x_a \tag{1}$$

subject to

$$\sum_{a=((i,t)(j,t'))\in\mathcal{A}|i\neq j} x_a = 1, \quad \forall i \in N, \tag{2}$$

$$\sum_{a \in \delta^+(i,t)} x_a - \sum_{a \in \delta^-(i,t)} x_a = 0, \quad \forall (i, t) \in \mathcal{N}, i \neq 0, \tag{3}$$

$$x_a \in \{0, 1\}, \quad \forall a \in \mathcal{A}. \tag{4}$$

Constraints (2) ensure that the vehicle arrives at each node exactly one time during its time window. Constraints (3) ensure that the vehicle departs every node at which it arrives. Finally, constraints (4) define the decision variables and their domains.

2 Literature Review

Due to the space, we refer [7] as a most recent review on TSP in general. Regarding time-dependent TSPTW literature, [6] propose a branch-and-cut algorithm

while [1] extend the ideas described in [5] by using a branch-and-bound algorithm. [1] show that lower and upper bounds of time-dependent asymmetric TSPTW can be obtained from the optimal solution of a well-defined asymmetric TSPTW. [4] present the first application of the DDD method to solve the static TSPTW problem. Based on that work, [9] propose the first DDD method to solve the time-dependent TSPTW problem under the assumption of a non-decreasing travel cost function. The experiment results showed that the algorithm outperformed the state-of-art method for a particular problem of this class, the make-span problem [6]. [10] extend the DDD approach for solving TD-TSPTW to the Time-Dependent Minimum Tour Duration Problem and the Time-Dependent Delivery Man Problem, which, unlike TSPTW, have a scheduling element. Regarding how to refine the partial networks, [8] study path-based refinement strategy and compare results of various refinement strategies applied to TSPTW when using layer graph expansion. For a general discussion of Dynamic Discretization Discovery, readers can refer to [3] as a base source.

In this paper, we extend the ideas in [9,10] to solve the TD-TSPTW with a generic travel cost function. As far as we know, it is the first research for TD-TSPTW with generic travel costs. We propose lower-bound formulations, and extend the DDD algorithms to find optimal solutions for this generalized problem.

3 Partially Time-expanded Network Formulation and Properties

To solve TD-TSPTW, we rely on the concept of the *partially time-expanded network* [2,9]. A *partially time-expanded network*, $\mathcal{D}_\mathcal{T} = (\mathcal{N}_\mathcal{T}, \mathcal{A}_\mathcal{T})$, is derived from a given subset of the timed nodes, $\mathcal{N}_\mathcal{T} \subseteq \mathcal{N}$. Given $\mathcal{N}_\mathcal{T}$, the arc set $\mathcal{A}_\mathcal{T} \subseteq \mathcal{N}_\mathcal{T} \times \mathcal{N}_\mathcal{T}$ consists of travel arcs and waiting arcs. A travel arc $((i,t),(j,t'))$, wherein $(i,t) \in \mathcal{N}_\mathcal{T}$, $(j,t') \in \mathcal{N}_\mathcal{T}$, $i \neq j$, and $(i,j) \in A$, models travel between locations i and j. We do not allow violations of time windows, so $t' \leq \max\{e_j, t+\tau_{ij}(t)\}$. An arc is *too short* if $t' < \max\{e_j, t+\tau_{ij}(t)\}$. A waiting arc $((i,t)(i,t+1))$ is in $\mathcal{A}_\mathcal{T}$ if both nodes (i,t) and $(i,t+1)$ are in $\mathcal{N}_\mathcal{T}$. For each arc $a = ((i,t),(j,t')) \in \mathcal{A}_\mathcal{T}$, we define $\underline{c}_{ij}(t)$ as the travel cost of arc a. We set these costs, $\underline{c}_{ij}(t)$, in such a manner that they under-estimate how the cost of such travel is presented in $\mathcal{D}$. Specifically, $\underline{c}_{ij}(t)$ is defined as $\underline{c}_{ij}(t) = \min\{\sum_{h=h'}^{h"-1} c_{ii}(h) + c_{ij}(h") | t \leq h' \leq h" \wedge h" + \tau_{ij}(h") \leq l_j\}$. The underestimated cost $\underline{c}_{ii}(t)$ for a waiting arc $((i,t)(i,t+1))$ is 0 since the waiting cost is taken into account while evaluating underestimated travel cost.

Given a partially time-expanded $\mathcal{D}_\mathcal{T}$ that meets Property 1-5, we establish the formulation TD-TSPTW($\mathcal{D}_\mathcal{T}$) defined by the objective function and constraints (5) - (8). We optimize this formulation with respect to the cost $\underline{c}_{ij}(t)$.

$$\text{TD-TSPTW}(\mathcal{D}_\mathcal{T}) : \min \sum_{a \in \mathcal{A}} \underline{c}_a x_a \tag{5}$$

$$\sum_{a=((i,t)(j,t'))\in\mathcal{A}_\mathcal{T}|i\neq j} x_a = 1, \forall i \in N, \tag{6}$$

$$\sum_{a\in\delta^-((i,t))} x_a - \sum_{a\in\delta^+((i,t))} x_a = 0, \forall (i,t) \in \mathcal{N}_\mathcal{T} \tag{7}$$

$$x_a \in \{0,1\}, \forall a \in \mathcal{A}_\mathcal{T}. \tag{8}$$

Property 1 *$\forall i \in N$, both nodes (i, e_i) and (i, l_i) are in $\mathcal{N}_\mathcal{T}$.*

Property 2 *$\forall (i,t) \in \mathcal{N}_\mathcal{T}, e_i \leq t$.*

Property 3 *If $(i,t) \in \mathcal{N}_\mathcal{T}$ and $(i, t+1) \in \mathcal{N}_\mathcal{T}$, the waiting arc $((i,t)(i,t+1))$ is in $\mathcal{A}_\mathcal{T}$.*

Property 4 *Underestimate travel-time arc: $\forall (i,t) \in \mathcal{N}_\mathcal{T}$ and arc $(i,j) \in A$, there is a travel arc of the form $((i,t)(j,t')) \in \mathcal{A}_\mathcal{T}$ if $t+\tau_{ij}(t) \leq l_j$. Furthermore, every travel arc $((i,t),(j,t')) \in \mathcal{A}_\mathcal{T}$ must have either (1) $t + \tau_{ij}(t) < e_j$ and $t' = e_j$, or (2) $e_j \leq t' \leq t + \tau_{ij}(t)$. Finally, there is no $(j,t'') \in \mathcal{N}_\mathcal{T}$ with $t' < t'' \leq t + \tau_{ij}(t)$.*

Property 5 *Underestimate travel cost of arc: $\forall ((i,t)(j,t')) \in \mathcal{A}_\mathcal{T}, i \neq j$, the cost $\underline{c}_{ij}(t) = \min\{\sum_{h=h'}^{h"-1} c_{ii}(h) + c_{ij}(h") | t \leq h' \leq h" \wedge h" + \tau_{ij}(h") \leq l_j\}$. Underestimate waiting cost $\underline{c}_{ii}(t)$ takes value 0 for all i and t.*

Property 1-5 ensure Lemma 1, 2 and 3. Since the non-decreasing property of $\underline{c}_{ij}(t)$, the algorithm proposed in [9] will converge to an optimal solution to the TD-TSPTW($\mathcal{D}$) but with the parameterized cost $\underline{c}_{ij}(t)$. Because $\underline{c}_{ij}(t)$ may be not equal $c_{ij}(t)$, we may not reach the optimal solution to TD-TSPTW($\mathcal{D}$). Also, using the result of Lemma 2, if all optimal solutions to TD-TSPTW($\mathcal{D}$) with the original cost have at least one waiting arc occurring after visiting some cities, then the current lower bound formulation, TD-TSPTW($\mathcal{D}_\mathcal{T}$), is not enough to find optimal solutions of TD-TSPTW($\mathcal{D}$).

Lemma 1. *$\underline{c}_{ij}(t)$ is a non-decreasing function of t.*

Lemma 2. *If TD-TSPTW($\mathcal{D}_\mathcal{T}$) with the parameterized cost $\underline{c}_{ij}(t)$ is feasible, it always has an optimal solution without waiting arcs.*

Lemma 3. *TD-TSPTW($\mathcal{D}_\mathcal{T}$) with cost $\underline{c}_{ij}(t)$ is a lower bound of TD-TSPTW($\mathcal{D}$) with cost $c_{ij}(t)$.*

As the first attempt to address the challenges, we extend the lower bound formulation TD-TSPTW($\mathcal{D}_\mathcal{T}$) (5) - (8) to take into account original travel cost function c. Conditions to determine whether we can evaluate an arc with its correct travel cost are presented. Then we introduce new algorithmic ideas and show how to adapt the algorithmic framework presented in [9] to find optimal solution to TD-TSPTW($\mathcal{D}$) using TD-TSPTW($\mathcal{D}_\mathcal{T}$). Let us state the conditions in which we can evaluate an arc with its correct travel cost.

Property 6 *An arc $a = ((i,t)(j,t')) \in \mathcal{D}_\mathcal{T}$ can be evaluated with correct travel cost if and only if two following conditions are met:*

1. *The node (i,t) can be reached by a sequence of correct travel time arcs and waiting arcs from the depot $(0,e_0)$.*
2. *The waiting arc $((i,t)(i,t+1))$ is in $\mathcal{D}_\mathcal{T}$.*

The first condition of Property 6 states that if an arc $a = ((i,t)(j,t'))$ can be evaluated with the correct travel cost, then it must be reached from the depot by a sequence of correct travel time arcs (including waiting arcs). With this condition, t is the correct arrival time at the location i. The second condition implies the possibility that we travel from location i at time t to another location with the correct travel cost, or we can wait and travel from i at time at least $t+1$ with underestimate travel cost. It is used to maintain the non-decreasing property of TD-TSPTW($\mathcal{D}_\mathcal{T}$) when we update $\mathcal{D}_\mathcal{T}$. Now, we present two new formulations satisfying Property 1-6.

4 New Lower Bound Formulations and Algorithm

4.1 Path-arc-based formulation

We start with a path-arc-based formulation that allows us to evaluate arcs with their correct travel costs by using additional path variables representing paths with correct travel times. Let $p = ((u_0 = 0, t_0 = e_0) - (u_1,t_1) - (u_2,t_2) - ... - (u_m,t_m))$ be a path departing from the depot with correct travel times. We associate with p a binary variable x_p and with cost $c_p = \sum_{i=0}^{m-1} c_{u_i u_{i+1}}(t_i)$. Let denote $\mathcal{P}_\mathcal{T}$ as a set of paths originated from the depot with correct travel times in $\mathcal{D}_\mathcal{T}$. We denote $\mathcal{P}_\mathcal{T}(i) \subseteq \mathcal{P}_\mathcal{T}$ as a set of paths visiting the city i and $\delta_p^+(i,t)$ as a set of paths ending at (i,t). Let $\delta^+(i,t) = \{((j,t')(i,t)) \in \mathcal{A}_\mathcal{T} | j \neq i\}$ denote of the set of travel arcs ending at (i,t). Let $\delta^+(i) = \{((j,t')(i,t)) \in \mathcal{A}_\mathcal{T} | j \neq i\}$ denote of the set of travel arcs ending at city (i). Let $\mathcal{A}_\mathcal{T}^= \subseteq \mathcal{A}_\mathcal{T}$ be the set of arcs with correct travel times, and let $\mathcal{Q}_\mathcal{T} \subseteq \mathcal{P}_\mathcal{T} \times \mathcal{A}_\mathcal{T}^=$ such that if $(p,a) \in \mathcal{Q}_\mathcal{T}$ then $p \oplus a \in \mathcal{P}_\mathcal{T}$. Here $p \oplus a = ((u_0 = 0, t_0 = e_0) - (u_1,t_1) - (u_2,t_2) - ... - (u_m = i, t_m = t), (j,t'))$, the path obtained by expanding the arc a to the end of p where $p \in \delta_p^+(i,t)$ and $a = ((i,t)(j,t')) \in \mathcal{A}_\mathcal{T}^=$. The path-based formulation TD-TSPTW($\mathcal{D}_\mathcal{T}, \mathcal{P}_\mathcal{T}$) is:

$$\text{TD-TSPTW}(\mathcal{D}_\mathcal{T}, \mathcal{P}_\mathcal{T}) : \min \sum_{a \in \mathcal{A}_\mathcal{T}} \underline{c}_a x_a + \sum_{p \in \mathcal{P}_\mathcal{T}} c_p x_p \tag{9}$$

$$x_a + \sum_{p \in \mathcal{P}_\mathcal{T} | a \in p} x_p \leq 1, \quad \forall a \in \mathcal{A}_\mathcal{T}^= \tag{10}$$

$$x_p + x_a \leq 1, \quad \forall (p,a) \in \mathcal{Q}_\mathcal{T} \tag{11}$$

$$\sum_{a \in \delta^+(i)} x_a + \sum_{p \in \mathcal{P}_\mathcal{T}(i)} x_p = 1, \quad \forall i \in N \tag{12}$$

$$\sum_{a\in\delta^+(i,t)} x_a + \sum_{p\in\delta_p^+(i,t)} x_p - \sum_{a\in\delta^-(i,t)} x_a = 0, \quad \forall (i,t) \in \mathcal{N}_{\mathcal{T}} \tag{13}$$

$$\sum_{((i,t)(i,t+1))\in\mathcal{A}_{\mathcal{T}}} x_{((i,t)(i,t+1))} = 0, \tag{14}$$

$$x_a, x_p \in \{0,1\}, \quad \forall a \in \mathcal{A}_{\mathcal{T}}, p \in \mathcal{P}_{\mathcal{T}}. \tag{15}$$

The objective function (9) estimates a lower bound of TD-TSPTW using paths with correct travel times and travel costs plus arcs with under-estimated travel costs. Suppose $p', p \in \mathcal{P}_{\mathcal{T}}$ and $p' = p \oplus a$ with some $a \in \mathcal{A}_{\mathcal{T}}^{=}$ then constraint set (11) forces to use p' instead of p and a to ensure that correct cost is used. However, if $p' \notin \mathcal{P}_{\mathcal{T}}$ and if p and a are selected, we will create p' and add p' to the formulation. Constraint set (12) ensures each city is visited exactly once (e.g. by an arc ending at (i,t) or by a path p passing through node (i,t)). Constraint set (13) ensures the balance of selected arcs at each node, either by arcs or by paths. Mathematically, if $x_a = 1$ for any $a \in \delta^+(i,t)$ in equation (12), then $\sum_{\mathcal{P}_{\mathcal{T}}(i,t)} x_p = 0$ and $\sum_{p\in\delta^+(i,t)} x_p = 0$ in (13). So there is an arc $a' \in \delta^-(i,t)$ with $x_{a'} = 1$, making the node (i,t) balanced. Otherwise, if $x_a = 0$ for all $a \in \delta^+(i,t)$, then $\sum_{\mathcal{P}_{\mathcal{T}}(i,t)} x_p = 1$. If $\sum_{p\in\delta^+(i,t)} x_p = 1$ then again there is an arc $a' \in \delta^-(i,t)$ with $x_{a'} = 1$, implying the node (i,t) balanced. Otherwise, $\sum_{p\in\delta^+(i,t)} x_p = 0$, so there is $p \in \mathcal{P}_{\mathcal{T}} \backslash \delta^+(i,t)$ such that $x_p = 1$. Because $p \notin \delta^+(i,t)$, there are exactly two arcs with correct travel time of form $((u,h)(i,t))$ and $((i,t)(j,t'))$ in p, making (i,t) balance. Constraint (14), which is used to strengthen the formulation by Lemma 3.2, eliminate waiting arcs with underestimated cost from the optimal solutions, in other words, waiting arcs (with correct travel costs) can only appear in paths in $\mathcal{P}_{\mathcal{T}}$. Finally, constraint (15) defines domains of the variables x and p.

We have the following results, which say that TSPTW($\mathcal{D}_{\mathcal{T}}, \mathcal{P}_{\mathcal{T}}$) is always a lower bound of TD-TSPTW($\mathcal{D}$) and explains when we find an optimal solution to TSPTW($\mathcal{D}$).

Lemma 4. *TSPTW($\mathcal{D}_{\mathcal{T}}, \mathcal{P}_{\mathcal{T}}$) is a lower bound of TSPTW($\mathcal{D}$).*

Lemma 5. *If an optimal solution to TSPTW($\mathcal{D}_{\mathcal{T}}, \mathcal{P}_{\mathcal{T}}$) is a single tour prescribed by a path (variable), it is an optimal solution to TSPTW($\mathcal{D}$).*

4.2 Arc-based Formulation

As we can see, there are two major disadvantages of the path-arc-based formulation TD-TSPTW($\mathcal{D}_{\mathcal{T}}, \mathcal{P}_{\mathcal{T}}$) (9)-(15). First, the number of paths in $\mathcal{P}_{\mathcal{T}}$ can be an exponential number in terms of the number of nodes and arcs in $\mathcal{D}_{\mathcal{T}}$, making it impossible to solve the TD-TSPTW($\mathcal{D}_{\mathcal{T}}, \mathcal{P}_{\mathcal{T}}$) by MIP solver. Second, given a sequence of correct travel time arcs in $\mathcal{D}_{\mathcal{T}}$, we cannot evaluate those arcs with correct travel costs unless there is a path including those arcs. We present another modeling approach to solve the two above issues. We associate with each

arc $a \in \mathcal{A}_\mathcal{T}$ a variable z_a indicating whether this arc can be evaluated with correct travel cost or not. Let $\Delta_a = c_a - \underline{c}_a$ for $\forall a \in \mathcal{A}_\mathcal{T}$.

The following defines a new relaxation of TD-TSPTW($\mathcal{D}$) that allows evaluating any sequence of arcs with correct travel time from the depot with their correct travel cost when Property 1-6 are met. Let $\mathcal{N}_\mathcal{T}^W = \{(i,t) \in \mathcal{N}_\mathcal{T} | (i,t+1) \in \mathcal{N}_\mathcal{T}\}$ be the set of time nodes having waiting arcs, and $\mathcal{N}_\mathcal{T}^{NW} = \mathcal{N}_\mathcal{T} \backslash \mathcal{N}_\mathcal{T}^W$ be the set without this property. Let $\lambda^+_{(i,t)}$ be the set of correct travel time/correct travel cost arcs arriving at the timed node (i,t), and λ^+_i be the set of correct travel time/correct travel cost arcs arriving at the node i.

$$\text{TD-TSPTW}(\mathcal{D}_\mathcal{T}, Z_{\mathcal{A}_\mathcal{T}}) : \min \sum_{a \in \mathcal{A}_\mathcal{T}} x_a \underline{c}_a + \sum_{a \in \mathcal{A}_\mathcal{T}} z_a \Delta_a \tag{16}$$

subject to constraints (6), (7) and

$$z_a \leq x_a, \forall a \in \mathcal{A}_\mathcal{T}, \tag{17}$$

$$z_a = x_a, \forall a \in \delta^-_{(0,e_0)} \text{ if } (0, e_0 + 1) \in \mathcal{N}_\mathcal{T}, \tag{18}$$

$$z_a = 0, \forall (i,t) \in \mathcal{N}_\mathcal{T}^{NW}, \forall a \in \delta^-_{(i,t)}, \tag{19}$$

$$z_a \geq x_a + \sum_{a' \in \lambda^+_{(i,t)}} z_{a'} - 1, \forall (i,t) \in \mathcal{N}_\mathcal{T}^W \backslash (0, e_0), \forall a \in \delta^-_{(i,t)}, \tag{20}$$

$$\sum_{a \in \lambda^+_{(i,t)}} z_a \geq \sum_{a \in \delta^-_{(i,t)}} z_a, \forall (i,t) \in \mathcal{N}_\mathcal{T} \backslash (0, e_0), \tag{21}$$

$$\sum_{a \in \lambda^+_{(i,t)}} z_a \geq x_{((i,t)(i,t+1))}, \forall (i,t) \in \mathcal{N}_\mathcal{T}^W \backslash (0, e_0) \tag{22}$$

$$x_a, z_a \in \{0,1\}, \forall a \in \mathcal{A}_\mathcal{T}. \tag{23}$$

The objective function (16) ensures that if x_a and z_a both take value 1 in a solution to TD-TSPTW($\mathcal{D}_\mathcal{T}, Z_{\mathcal{A}_\mathcal{T}}$), the cost associated to this arc in the objective function is exactly c_a. We are going to prove that given a solution $\{\bar{x}, \bar{z}\}$ to TD-TSPTW($\mathcal{D}_\mathcal{T}, Z_{\mathcal{A}_\mathcal{T}}$), for any $a = ((i,t)(j,t')) \in \mathcal{A}_\mathcal{T}$, $\bar{z}_a$ takes the value of 1 if and only if Property 6 is met.

Constraint set (17) ensures that an arc a is evaluated with its correct travel cost only if this arc is selected in a solution to the formulation. Next, constraint (18) implies that arcs representing departure from $(0, e_0)$ should be evaluated with correct travel costs if waiting arc $((0, e_0), (0, e_0+1)) \in \mathcal{A}_\mathcal{T}$. Next, constraint (19) enforces that if there is no waiting possibility at the node (i,t) in the current $\mathcal{D}_\mathcal{T}$, all arcs outgoing from this node cannot be evaluated with correct travel costs. Otherwise, constraint set (20) says that if $a = ((i,t)(j,t')) \in \mathcal{A}_\mathcal{T}$ is selected and if we can reach the node (i,t) by an arc $a' \in \mathcal{A}^=_\mathcal{T}$ which is also evaluated by

its correct travel cost ($\sum_{a' \in \lambda^+_{(i,t)}} z_{a'} = 1$ or $z'_a = 1$ for some a'), this constraint forces a to be evaluated by its correct travel cost, or $z_a = 1$. Constraint (21) says that if we cannot reach (i,t) by a correct travel time and correct travel cost arc, no outgoing arc of this node can be evaluated with the correct travel cost. Precisely, if $\sum_{a' \in \lambda^+_{(i,t)}} z_{a'} = 0$, constraint (21) forces that any outgoing arc $a \in \delta^-_{(i,t)}$ will be evaluated with its underestimate cost since $z_a = 0$ for all $a \in \delta^-_{(i,t)}$). This constraint also forces $z_a = 0$ for any arc a in a sub-tour $((u_0,t_0), (u_1,t_1), ..., (u_m,t_m), (u_0,t_0))$ where $u_i \neq 0$ for all i. W.r.t, we assume $t_0 \leq t_m$. Because the too short incoming arc $((u_m,t_m),(u_0,t_0)) \notin \lambda^+_{(u_0,t_0)}$ of the node (u_0,t_0) is selected, then the left-hand side of constraint (21) takes the value 0, consequently, $z_a = 0$ for all arcs a of the sub-tour $((u_0,t_0), (u_1,t_1), ..., (u_m,t_m), (u_0,t_0))$. Finally, constraint set (23) defines the domain of variables.

In conclusion, the set of constraints (17) - (23) ensures that if an arc is evaluated with correct travel cost, this arc must belong to the path from the depot node $(0,e_0)$ and all arcs in this path also must be evaluated with correct travel costs.

Aggregation formulation of TD-TSPTW($\mathcal{D}_\mathcal{T}, Z_{\mathcal{A}_\mathcal{T}}$) Aggregating constraints (18) - (21) gives us constraints (24) - (27). λ^+_i and δ^-_i present the set of correct-travel-time inbound arcs to city i and set of outbound arcs from city i. The aggregated formulation TD-TSPTW-AGG($\mathcal{D}_\mathcal{T}, \mathcal{A}_\mathcal{T}$) includes the objective function (16), the constraints (6), (7), (17), (23) and the constraints

$$\sum_{a \in \delta^-_{(0,e_0)}} x_a = \sum_{a \in \delta^-_{(0,e_0)}} z_a \text{ if } (0, e_0+1) \in \mathcal{N}_\mathcal{T}, \tag{24}$$

$$\sum_{(i,t) \in \mathcal{N}^{NW}_\mathcal{T}} \sum_{a \in \delta^-_{(i,t)}} z_a = 0, \tag{25}$$

$$\sum_{a \in \delta^-_{(i,t)}} z_a \geq \sum_{a \in \delta^-_{(i,t)}} x_a + \sum_{a \in \lambda^+_{(i,t)}} z_a - 1, \forall (i,t) \in \mathcal{N}^W_\mathcal{T} \backslash \{0, e_0\}, \tag{26}$$

$$\sum_{a \in \lambda^+_i} z_a \geq \sum_{a \in \delta^-_i} z_a, \forall i \in N \backslash 0. \tag{27}$$

Actually, while TD-TSPTW-AGG($\mathcal{D}_\mathcal{T}, Z_{\mathcal{A}_\mathcal{T}}$) is the aggregated version of TD-TSPTW($\mathcal{D}_\mathcal{T}, Z_{\mathcal{A}_\mathcal{T}}$), we can prove that they are equivalent (see online Appendix [11]). Similar to the results of the path-based formulation, we have:

Lemma 6. *If $\mathcal{D}_\mathcal{T}$ satisfies Properties 1 - 5, then TD-TSPTW($\mathcal{D}_\mathcal{T}, Z_{\mathcal{A}_\mathcal{T}}$) (or TD-TSPTW-AGG($\mathcal{D}_\mathcal{T}, Z_{\mathcal{A}_\mathcal{T}}$), respectively) is a relaxation of TD-TSPTW($\mathcal{D}$).*

Lemma 7. *An optimal solution to TD-TSPTW($\mathcal{D}_\mathcal{T}, \mathcal{A}_\mathcal{T}$) (or TD-TSPTW-AGG ($\mathcal{D}_\mathcal{T}, Z_{\mathcal{A}_\mathcal{T}}$), respectively) that has no too short arcs defines an optimal solution to TD-TSPTW($\mathcal{D}$)*

Algorithm 1 DDD-TD-TSPTW

Require: TD-TSPTW instance (N, A), e, l, τ and c, and optimality tolerance ϵ

1: Perform preprocessing, updating A, e and l.
2: Create a partially time-expanded network $\mathcal{D}_\mathcal{T}$.
3: Set $\mathcal{P}_\mathcal{T} \leftarrow \emptyset$, (or $\mathcal{A}_\mathcal{T}^{=} \leftarrow$ the set of correct travel time arcs in $\mathcal{A}_\mathcal{T}$)
4: Set $S \leftarrow \emptyset$
5: **while** not solved **do**
6: Set $\bar{S} \leftarrow \emptyset$
7: Solve primal heuristics, TD-TSPTW($\mathcal{D}_\mathcal{T}^1$) and TD-TSPTW($\mathcal{D}_\mathcal{T}^2$), with underestimate cost $\underline{c}$, harvest integer solutions and add to $\bar{S}$.
8: Solve TD-TSPTW($\mathcal{D}_\mathcal{T}$), harvest integer solutions, $\bar{S}$, and lower bound, z.
9: **for** $s \in \bar{S}$ **do**
10: Let s' be a copy of s without waiting arcs.
11: **if** s' can be converted to a feasible solution to TD-TSPTW($\mathcal{D}$) **then**
12: Solve R-TD-TSPTW($\mathcal{D}, s'$) to find the best tour using travel arcs in s'.
13: Update S with the solution returned by R-TD-TSPTW($\mathcal{D}, s'$).
14: Add a cut to exclude all copies of s' in TD-TSPTW($\mathcal{D}_\mathcal{T}$).
15: **else**
16: Add cuts corresponding to sub-tours and infeasible paths in s' to TD-TSPTW($\mathcal{D}_\mathcal{T}$)
17: **end if**
18: Update $\mathcal{D}_\mathcal{T}$, (and $\mathcal{P}_\mathcal{T}$), and TD-TSPTW($\mathcal{D}_\mathcal{T}$) by lengthening arcs in s.
19: **end for**
20: Compute gap δ between the best solution in S and lower bound, z.
21: **if** $\delta \leq \epsilon$ **then**
22: Stop: best solution in S is $\epsilon-$optimal for TSPTW.
23: **end if**
24: **end while**

4.3 Algorithm to solve TD-TSPTW($\mathcal{D}_\mathcal{T}$)

Algorithm 1 shows how we solve the TD-TSPTW($\mathcal{D}$) using proposed lower bound formulations. Readers can refer to [4,9,10] for additional reference of the components of the algorithm. In Algorithm 1, TD-TSPTW($\mathcal{D}_\mathcal{T}$) refers to one of the lower bound formulations presented in Section 4. While it shares the main steps with the one mentioned in [9,10], there are differences. First, to check the feasibility of a solution s found by lower bound formulations, we check with the solution s' obtained from s excluding all waiting arcs (Line 10-11). It ensures that if s' is infeasible, s is always infeasible. If s' is feasible, we solve R-TD-TSPTW(D, s') to find the best tour using only travel arcs in s' (e.g. by adding waiting arcs to s', Line 12). R-TD-TSPTW(D, s') is a restricted formulation of TD-TSPTW(D) where only travel arcs in s' can be selected. If R-TD-TSPTW(D, s') is solved to optimality, we add a cut to exclude all copies of s' from TD-TSPTW($\mathcal{D}_\mathcal{T}$). If s' is infeasible, we add cuts corresponding to sub-tours and infeasible paths extracted from s' to TD-TSPTW($\mathcal{D}_\mathcal{T}$). The two primal heuristics in [9,10] are used to help find feasible solutions. We exclude all waiting arcs when solving those two primal heuristics (Line 7). Arcs in those primal heuristics are evaluated with

under-estimate cost $\underline{c}_{ij}(t)$ and with basic formulation (5)-(8). We add cuts to exclude all tours corresponding to feasible solutions in S before solving those primal problems to force finding new feasible solutions. When updating $\mathcal{D}_\mathcal{T}$ (Line 18), given an arc $((i,t)(j,t'))$ to lengthen, we also add the node $(i,t+1)$ to $\mathcal{D}_\mathcal{T}$ to introduce waiting opportunities at (i,t) if $(i,t+1) \notin \mathcal{D}_\mathcal{T}$. To maintain Property 1-5, when a new node (i,t) is added to $\mathcal{D}_\mathcal{T}$, we also add arc $((i,t)(i,t+1))$ (or $((i,t-1)(i,t))$) if node $(i,t+1)$ (or $(i,t-1)$) existed in $\mathcal{D}_\mathcal{T}$. When the path-arc-based formulation is used, Algorithm ADD-PATHS (see online Appendix [11]) is employed to update $\mathcal{P}_\mathcal{T}$.

5 Experiments

The algorithm is implemented in C++ using Gurobi 8.0 as the MIP solver. All experiments were run on a workstation with an Intel(R) Xeon (R) CPU E5-4610 v2 2.30GHz processor running Ubuntu 14.04.3. One of the stopping conditions was a provable optimality gap of $\epsilon = 10^{-2}$. Two sets named "Set 1" and "Set w100" from [9] are used. Each set has 960 instances from 15 to 40 nodes with up to 73 travel time profiles. We set the cost $c_{ij}(t)$ be $\tau_{ij}(t)$, the travel time between i and j at time point t. We assess two points:

1. We compare the path-arc-based formulation (Path), the arc-based formulation (Z), and the aggregated arc-based formulation (Z-Agg) and the corresponding algorithms based on the number of instances solved.
2. We compare the strongest algorithm and formulation and the state-of-the-art solver, Gurobi, solving the problem with the full-time-expanded network formulation.

First, Table 5 reports the number of solved instances using the Path, Z, and Z-Agg formulation. In this experiment, we consider a setting in which waiting at site i after visiting i is not allowed, so $c_{ii}(t) = \infty$, or a very high value. Maximum running time for this setting is 1 hour. This can happen for time-dependent scheduling problems where all jobs (cities) are performed without stopping. As we expect, the Z-Agg formulation is the most efficient and competitive formulation, while the Path formulation is the worst one. Using the Z-Agg formulation, we can solve 873 and 871 instances of Set 1 and Set w100 to optimality. It means the proposed algorithm is able to solve this particular variant.

Table 1. Number of instances solved optimally by each formulation.

	Set 1			Set w100		
n	Path	Z	Z-Agg	Path	Z	Z-Agg
15	240	240	240	240	240	240
20	239	239	240	226	228	231
30	214	218	224	192	230	232
40	147	149	169	115	167	188

Second, we consider the setting in which $c_{ii}(t) = 0$. This setting is harder because of the larger solution space, so we let 2 hours of execution. We observe that while Gurobi can efficiently solve instances of Set w100, it struggles to find feasible solutions to instances of Set 1. Given two hours of computation time, it can only find feasible solutions to 596 over 960 instances of Set 1 (Table 5), while it is able to find feasible solutions to all instances of Set w100. Technically, while having the same number of nodes, instances of Set 1 have wider time windows, making the complete networks larger and harder to solve. The proposed algorithm finds feasible solutions for all instances.

Table 2. Feasible solutions: Gurobi versus Z-Agg (Set 1)

n	15	20	30	40
Gurobi	239	228	115	14
Z-Agg	240	240	240	240

Finally, Table 3 compares the number of ϵ-optimal solutions that Gurobi and the proposed algorithm find for instances of Set 1. Gurobi can prove optimality for 396 instances, while the proposed method with Z-Agg formulation can solve 712 instances. The average gap of unsolved instances is 7.37% (Gurobi) and 2.94% (the proposed algorithm). These preliminary results show that the proposed algorithm and formulations are promising for solving time-dependent TSPTW instances.

Table 3. Optimal solutions: Gurobi versus Z-Agg (Set 1)

n	15	20	30	40
Gurobi	218	145	29	4
Z-Agg	228	199	157	128

To conclude, the three lower bound formulations can be used to solve the time-dependent TD-TSPTW in which the aggregated formulation TD-TSPTW-AGG($\mathcal{D}_\mathcal{T}, \mathcal{A}_\mathcal{T}$) is the most effective one. It has fewer constraints than TD-TSPTW($\mathcal{D}_\mathcal{T}, \mathcal{A}_\mathcal{T}$), and therefore, makes it easier to solve with mixed-integer programming solvers. The proposed algorithm with the aggregated formulation performs better than the solver over benchmark instances.

6 Conclusion

In this paper, we study a generalized version of the time-dependent traveling salesman problem where travel cost is modeled as a generic function. We present

three lower-bound formulations based on path and arc variables, and we introduce iterative exact algorithms based on the dynamic discrete discovery approach to solve the problems. The experiment results confirm the advantages of the proposed method over the state-of-art solvers when solving small- and medium-sized instances. Finally, our ongoing research shows that we can apply the proposed method in this paper to solve variants of the TSPTW including those with soft time windows. It confirms the strength and innovation of our proposed method.

Acknowledgement

The work has been carried out partly at the Vietnam Institute for Advanced Study in Mathematics (VIASM). The corresponding author (Duc Minh Vu) would like to thank VIASM for its hospitality and financial support for his visit in 2023.

References

1. Arigliano, A., Ghiani, G., Grieco, A., Guerriero, E., Plana, I.: Time-dependent asymmetric traveling salesman problem with time windows: Properties and an exact algorithm. Dis. App. Mat. (2018)
2. Boland, N., Hewitt, M., Marshall, L., Savelsbergh, M.: The continuous-time service network design problem. Ope. Res. **65**(5), 1303–1321 (2017)
3. Boland, N.L., Savelsbergh, M.W.: Perspectives on integer programming for time-dependent models. TOP pp. 1–27 (2019)
4. Boland, N., Hewitt, M., Vu, D.M., Savelsbergh, M.: Solving the traveling salesman problem with time windows through dynamically generated time-expanded networks. In: 14th Conference on Integration of Artificial Intelligence and Operations Research Techniques in Constraint Programming. vol. LNCS. Springer (2017)
5. Cordeau, J., Ghiani, G., Guerriero, E.: Analysis and branch-and-cut algorithm for the time-dependent travelling salesman problem. Tra. Sci. **48**(1), 46–58 (2014)
6. Montero, A., Méndez-Dıaz, I., Miranda-Bront, J.J.: An integer programming approach for the time-dependent traveling salesman problem with time windows. Com. & Ope. Res. **88**, 280–289 (2017)
7. Pop, P.C., Cosma, O., Sabo, C., Sitar, C.P.: A comprehensive survey on the generalized traveling salesman problem. Eur. Jou. of Ope. Res. (2023)
8. Riedler, M., Ruthmair, M., Raidl, G.R.: Strategies for iteratively refining layered graph models. In: International Workshop on Hybrid Metaheuristics. pp. 46–62. Springer (2019)
9. Vu, D.M., Hewitt, M., Boland, N., Savelsbergh, M.: Dynamic discretization discovery for solving the time-dependent traveling salesman problem with time windows. Tran. Sci. **54**(3), 703–720 (2020)
10. Vu, D.M., Hewitt, M., Vu, D.D.: Solving the time dependent minimum tour duration and delivery man problems with dynamic discretization discovery. Eur. Jou. of Ope. Res. **302**(3), 831–846 (2022)
11. Vu, D.M., Hewitt, M., Vu, D.D.: Solving time-dependent traveling salesman problem with time windows under generic time-dependent travel cost (2023), https://arxiv.org/abs/2311.08111