TOP (2019) 27:147–173
https://doi.org/10.1007/s11750-019-00514-4

INVITED PAPER

# Perspectives on integer programming for time-dependent models

Natashia L. Boland[1] · Martin W. P. Savelsbergh[1]

Published online: 10 May 2019

**Abstract**
Integer programs for solving time-dependent models—models in which decisions have to be made about the times at which activities occur and/or resources are utilized—are pervasive in industry, but are notoriously difficult to solve. In the last few years, interest in the role of discretization in approaches to solve these problems has intensified. One novel paradigm, dynamic discretization discovery, has emerged with the potential to greatly enhance the practical tractability of time-dependent models using integer programming technology. We introduce dynamic discretization discovery, illustrate its use on the traveling salesman problem with time windows, highlight its core principles, and point to opportunities for further research. Relations to other approaches for tackling time-dependent models are also discussed.

**Keywords** Integer programming · Time-expanded network · Dynamic discretization discovery · Traveling salesman problem with time windows

**Mathematics Subject Classification** 90C10 · 90C90

## 1 Introduction

In 2006, Martin Grötschel, then Secretary of the International Mathematical Union, wrote

This invited paper is discussed in the comments available at https://doi.org/10.1007/s11750-019-00510-8, https://doi.org/10.1007/s11750-019-00511-7, https://doi.org/10.1007/s11750-019-00512-6.

✉ Natashia L. Boland
natashia.boland@isye.gatech.edu

Martin W. P. Savelsbergh
martin.savelsbergh@isye.gatech.edu

1 H. Milton Stewart School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, USA

The development of linear programming is, in my opinion, the most important contribution of the mathematics of the 20th century to the solution of practical problems arising in industry and commerce. The subsequent progress in the applicability of integer and combinatorial optimization begins, at present, to even surpass this impact.

Grötschel's conjecture was indeed right. Since 2006, the use of *integer linear programming* (IP) has easily surpassed the use of linear programming in commercial software and now accounts for a substantial majority of the use of all optimization software. A majority of finalists for the Edelman prize (the most prestigious prize for the application of operations research and analytics) use IP as core technology, in industries including health, energy, transportation, and logistics.

However, there are important classes of IPs, whose optimal solution could yield substantial benefits, but which cannot be solved satisfactorily. One such class is the IPs that we address in this paper: IPs that are characterized by being *about time*. In these problems, decisions have to be made about the times at which activities occur and/or resources are utilized. We call the models of these problems *time dependent* (TD). They are pervasive in applications, since they occur whenever a schedule of activities needs to be constructed.

A classic example is airline scheduling, in which activities (flights) and resources (airplanes and crews) must be scheduled to meet the demand of potential passengers. A 1- or 2-min change in the official departure time of a flight can have a huge impact on the legal connections that are possible for both crews and passengers. In the energy industry, the unit commitment problem, for scheduling the generators of power in a day-ahead electricity market, is to decide, at each time, the on–off states of the generators, and so is very much about time. In health care, surgery facilities and expensive equipment need to be scheduled and coordinated to maximize patient care in an environment with scarce resources. Timing and coordination are similarly critical in the design and operation of air- and sea-lift networks for defense forces. Innovative logistics practices driven by more demanding markets, such as same-day and next-hour delivery of online orders, are creating a need for more precise TD models that can be solved efficiently. Fuel efficiency and greenhouse gas emissions from delivery vehicles depend critically on their speed of travel. Traffic congestion typically occurs at specific times of day, so the departure time of a trip can have a strong impact on its fuel efficiency. Thus, optimizing fuel efficiency also requires TD modeling.

Unfortunately, TD models are notoriously difficult to solve. Compact models that use continuous variables to model time have weak linear programming (LP) relaxations. Their solution with current IP solver technology is limited to only small instances. Extended formulations, with binary variables indexed by time, have much stronger relaxations, but (tend to) have a huge number of variables. Such formulations rely on a *discretization* of time, which introduces approximation. Fine discretizations yield optimal, or near-optimal, solutions, but their IP formulations are prohibitively large for current solvers. Thus, in practice, coarse discretizations are used. This can significantly reduce the quality of solutions and may even cause infeasibility.

In the last few years, a novel paradigm, dynamic discretization discovery, has emerged as a way to overcome some of these challenges and effectively and efficiently find optimal or near-optimal solutions to TD models. Dynamic discretization discovery allows the solution of a TD model based on a fine discretization **without ever fully constructing it**. The paradigm has three main components:

- the design of time-indexed IP models, based on a *partial discretization of time* that are efficiently solvable in practice and that yield lower bounds, upper bounds, or exact solutions;
- the design of algorithms that *dynamically discover* partial discretizations, i.e., algorithms that can "refine" a partial time discretization to strengthen the quality of a time-indexed IP model; and
- the design of algorithms that efficiently solve time-indexed IP models.

The goal of this paper is to present perspectives on various aspects of time-dependent models and to introduce the primary concepts underlying dynamic discretization discovery. The aim is to provide food for thought and to offer readers an opportunity to learn what is, in our view, an exciting area of research with a huge need and potential for further contributions.

The remainder of the paper is organized as follows. In Sect. 2, we introduce the key elements and concepts of dynamic discretization discovery (DDD) via their application to a fundamental problem, the traveling salesman problem with time windows, and to its generalization to arc travel times that are time dependent. In Sect. 3, we discuss the general principles of DDD, in both theory and algorithm design. Then, in Sects. 4, 5, and 6, we discuss alternative approaches and related research. Finally, in Sect. 7, we present opportunities for further research.

## 2 Dynamic discretization discovery: a traveling salesman case study

Time-indexed (TI) formulations require a discretization of time known to be fine enough to provide a correct model. We refer to such a discretization as *complete*, and to the corresponding model as a *complete TI model*. Its solution is *continuous-time optimal*.

### 2.1 Time-indexed formulations

To illustrate the relevant concepts, we consider the *traveling salesman problem with time windows* (TSPTW). Let $D = (N_0, A)$ be a complete directed graph with node set $N_0 = N \cup \{0\}$ and arc set $A$, where 0 denotes the *depot* node at which the tour starts and ends. Let the cost of traversing arc $a \in A$ be $c_a \in \mathbb{R}$ and the time to traverse arc $a \in A$ be $\tau_a \in \mathbb{Z}_{>0}$. The time required to serve each node is assumed to be included in the traversal time of each outgoing arc. Furthermore, each node $i \in N_0$ can only be visited during the interval $[e_i, l_i]$, where $e_i, l_i \in \mathbb{Z}_{>0}$, $e_i \leq l_i$, and, without loss of generality, we assume that $e_i \geq e_0 + \tau_{0i}$ and $l_i \leq l_0 - \tau_{i0}$ for $i \in N$. The TSPTW seeks a minimum cost tour starting and ending at 0, departing 0 at or after $e_0$ and returning to 0 at or before $l_0$, visiting each node $i \in N$, and departing each node $i \in N$ in its associated time window $[e_i, l_i]$.

Note that we have assumed that travel times and time window limits are integer. In this case, it is obvious that there is an optimal solution in which all departures occur at integer times. Thus, the integer discretization is complete. This assumption simplifies our presentation of DDD concepts, but is not needed to apply the DDD approach. In Sect. 3.1, we discuss the necessary conditions, in general.

A natural compact (nonlinear) formulation for the TSPTW is as follows:

$$\min \sum_i \sum_j c_{ij} y_{ij} \qquad ,$$

$$\sum_{j \in \delta^+(i)} y_{ij} = 1, \qquad \text{for all } i \in N_0, \tag{1}$$

$$\sum_{j \in \delta^-(i)} y_{ji} - \sum_{j \in \delta^+(i)} y_{ij} = 0 \qquad \text{for all } i \in N_0 \tag{2}$$

$$(t_j - t_i - \tau_{ij}) y_{ij} \geq 0 \qquad \text{for all } i, j \in N_0, j \neq 0, \tag{3}$$

$$e_i \leq t_i \leq l_i \qquad \text{for all } i \in N_0, \tag{4}$$

$$y_{ij} \in \{0, 1\} \qquad \text{for all } i, j \in N_0,$$

$$t_i \in \mathbb{R}_{\geq 0} \qquad \text{for all } i \in N_0,$$

where binary variable $y_{ij}$ indicates that arc $(i, j) \in A$ is used in the tour, variable $t_i$ models the departure time at node $i$ and $\delta^+(i)$ (respectively, $\delta^-(i)$) is used to denote the set of arcs leaving (respectively, entering) node $i \in N_0$. Constraints (1) and (2) force that every node is visited exactly once and Constraints (3) and (4) ensure that each node is visited within its time window and that travel times are properly accounted for. Nonlinear constraints (3) can be linearized in the standard way and converted to "big-$M$" constraints, resulting in the following integer programming formulation:

$$\min \sum_i \sum_j c_{ij} y_{ij} \qquad ,$$

$$\sum_{j \in \delta^+(i)} y_{ij} = 1, \qquad \text{for all } i \in N_0, \tag{5}$$

$$\sum_{j \in \delta^-(i)} y_{ji} - \sum_{j \in \delta^+(i)} y_{ij} = 0 \qquad \text{for all } i \in N_0, \tag{6}$$

$$t_j \geq t_i + \tau_{ij} - M_{ij}(1 - y_{ij}) \qquad \text{for all } i, j \in N_0, j, \neq 0, \tag{7}$$

$$e_i \leq t_i \leq l_i \qquad \text{for all } i \in N_0, \tag{8}$$

$$y_{ij} \in \{0, 1\} \qquad \text{for all } i, j \in N_0,$$

$$t_i \in \mathbb{R}_{\geq 0} \qquad \text{for all } i \in N_0,$$

where $M_{ij}$ can be set to a sufficiently large value, e.g., $M_{ij} = \ell_i + \tau_{ij} - e_j$ will do. Due to the presence of the big-$M$ constraints, (7), the LP relaxation tends to be weak, which leads to large search trees. Consequently, only small instances can be solved using this formulation.

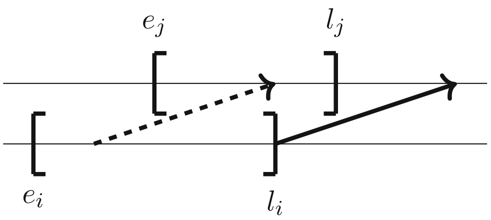

**Fig. 1** Over-constrained: the model makes the pessimistic assumption that the tour will not depart from node $i$ early enough to visit node $j$ in its time window, so arc $(i, j)$ is not included

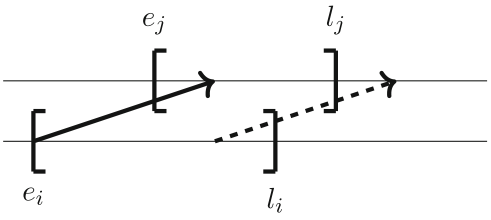

**Fig. 2** Under-constrained: the model makes the optimistic assumption that the tour will depart from node $i$ early enough to visit node $j$ in its time window, so arc $(i, j)$ is included

Rather than incorporating times explicitly in the model, i.e., rather than using the variables $t_i$, Wang and Regan (2002) suggest to model times in the set of arcs. That is, time window and travel time information are used to define the arcs connecting nodes. An over-constrained version is proposed, which provides an upper bound, if feasible, and an under-constrained version is proposed, which provides a lower bound. The over-constrained version arises when an arc $(i, j)$ is included if and only if $l_i + \tau_{ij} \leq l_j$. The under-constrained version arises when an arc $(i, j)$ is included if and only if $e_i + \tau_{ij} \leq l_j$. Figures 1 and 2 illustrate these concepts, where the solid arcs indicate those used to determine whether or not an arc $(i, j)$ is included in the set of feasible arcs, and the dashed arcs indicate why the resulting model is either over-constrained or under-constrained.

Depending on the width of the time windows, the bound obtained from the lower bound model may be weak and the upper bound model may not have a feasible solution. Therefore, Wang and Regan (2002) propose to partition the time windows into subwindows. For a given $\Delta$, a time window is partitioned into $\lceil \frac{l_i - e_i}{\Delta} \rceil$ equal size subwindows. The formulation is modified to reflect that only one of the nodes associated with each of the subwindows has to be visited. It is easy to see that the value of the lower model may improve and that the likelihood that the upper bound model has a feasible solution increases when the time windows are partitioned into subwindows. Wang and Regan (2009) prove the intuitive result that when $\Delta$ goes to zero, and, thus, the number of elements in the partition goes to infinity, the lower bound model produces an optimal solution as all its solutions will be feasible. In fact, when the number of elements in the partition is sufficiently large, a complete TI model results.

This complete TI model can be seen as the problem of finding a minimum-cost path through a time-expanded network, based on a complete discretization of time. Such a time-expanded network has nodes $(i, t)$ representing node $i \in N$ at time $t \in \{e_i, e_i + 1, \dots, l_i\}$ and nodes $(0, e_0)$ and $(0, l_0)$, representing the tour start and end, respectively. It has arcs $((0, e_0), (i, e_i))$ for $i \in N$, to model the possibility that $i$ is the first node in the tour, arcs $((i, t), (j, \max\{e_j, t + \tau_{ij}\}))$ for $i, j \in N$ and

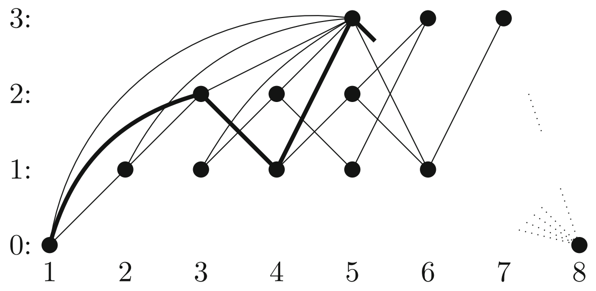

**Fig. 3** An example time-expanded network and a feasible path (in bold). Each node $(i, t)$ is represented by $t$ on the horizontal axis and $i$ on the vertical axis. Arcs into node 0 are omitted

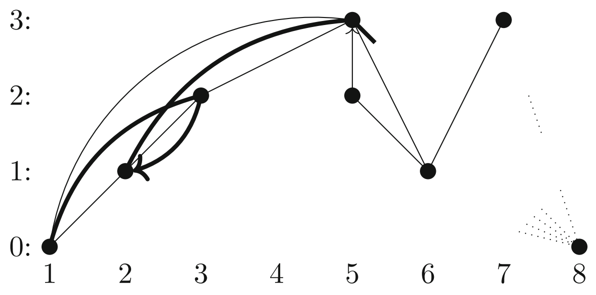

**Fig. 4** A relaxed network and a feasible path (in bold). Arrows indicate arcs that are not forward in time

$t = e_i, \ldots, \min\{l_i, l_j - \tau_{ij}\}$, to model travel from node $i$ to $j$ departing at time $t$, and arcs $((i, t), (0, l_0))$ for each $t = e_i, e_i + 1, \ldots, l_i$ to model the possibility that $i$ is the last node in the tour. Note that this network relies on the fact that travel times and time windows are specified as integers and exploit the fact that, without loss of generality, it can be assumed that any waiting occurs before the opening of the window at a node. (For a given instance, preprocessing can be used to remove some nodes and arcs that cannot appear in a feasible, or in every optimal, solution. For brevity, we will not discuss such preprocessing.)

Solving the TSPTW is equivalent to seeking a minimum cost path from $(0, e_0)$ to $(0, l_0)$ that visits each node in $N$ exactly once. An example time-expanded network and a feasible path can be seen in Fig. 3 for in instance of the TSPTW with $N_0 = \{0, 1, 2, 3\}$, travel times $\tau_a = 1$ for all $a \in A$, and $\{e_0, l_0\} = \{1, 8\}$, $\{e_1, l_1\} = \{2, 6\}$, $\{e_2, l_2\} = \{3, 5\}$, and $\{e_3, l_3\} = \{5, 7\}$.

A similar model was introduced by Pessoa et al. (2010) for solving parallel identical machine scheduling problems. Their arc time-indexed formulation has variables $x_{ij}^t$ that indicate whether job $i$ completes and job $j$ starts at time $t$ on the same machine. A path through the underlying time-expanded network represents the schedule of one of the machines.

One of the reasons for considering IP formulations based on time-expanded networks is that they are known to be strong, i.e., have tight LP relaxations. Our tests on a set of TSPTW benchmark instances shows the LP gap to be less than 1.1% on average. Unfortunately, such time-expanded networks are usually too large for the resulting integer program to be solved in an acceptable run time. Thus, specialized algorithms need to be developed for their solution. We focus on one such class of algorithms: dynamic discretization discovery algorithms. However, before doing so, we discuss some other benefits of IP formulations based on time-expanded networks.

It is easy to see that, in principle, time-dependent travel times, i.e., settings in which the travel time on an arc depends on the time of travel, can easily be modeled in a time-expanded network. The need to handle time-dependent travel times is increasingly important, as it allows modeling of travel time patterns commonly encountered in practice, e.g., morning and evening rush hour traffic. It also allows modeling of travel times on arcs that represent a scheduled service, such as a ferry.

In such cases, the time to traverse an arc $(i, j)$ is not a constant value, $\tau_{ij}$, but is instead modeled as a function, $\tau_{ij}(t)$, of the departure time, $t$, at the tail node of the arc, node $i$. Such functions are usually assumed to satisfy the *first-in first-out* property (FIFO), which implies that for travel along an arc, departing later at its tail node can never lead to arriving earlier at its head node.

Provided that we assume node visit times must be integer (which is now no longer without loss of generality), the only change required to model the time-dependent traveling salesman problem with time windows (TD-TSPTW) is to use arcs $((i, t), (j, \max\{e_j, t + \tau_{ij}(t)\}))$, for $i, j \in N$ and $t = e_i, \ldots, \min\{l_i, l_j - \tau_{ij}^{-1}(l_j)\}$, to model travel from node $i$ to $j$ departing at time $t$, rather than $((i, t), (j, \max\{e_j, t + \tau_{ij}\}))$. Here, we abuse notation in writing $\tau_{ij}^{-1}(t)$ to denote the time needed to traverse arc $(i, j)$ as a function of the arrival time, $t$, at the head node of the arc, node $j$. Note that $\tau^{-1}$ is not the inverse of $\tau$, but rather $\tau_{ij}^{-1}(t + \tau_{ij}(t)) = \tau_{ij}(t)$. (As an aside, we note that the time-dependent traveling salesman problem, as introduced by Picard and Queyranne (1978), does not refer to time-dependent travel times, but to position-dependent costs—an unfortunate misnomer.)

In Sect. 3.1, we discuss how complete TI models may still be derived for TD-TSPTW, without the assumption of integer node visit times. Next, we introduce key concepts needed for the use of dynamic discretization to solve the complete TI model for the TSPTW.

*Partial time discretization*

As mentioned above, IP formulations based on time-expanded networks are known to be strong, but are often too large to be solved in an acceptable amount of time. Therefore, we consider partially time-expanded networks rather than fully time-expanded networks.

A partially time-expanded network $\mathcal{D}_{\mathcal{T}} = (\mathcal{N}_{\mathcal{T}}, \mathcal{A}_{\mathcal{T}})$ is derived from subsets of the time points that could be modeled at each node. Specifically, we denote the collection of modeled time points as $\mathcal{T} = \{\mathcal{T}_i\}_{i \in N_0}$, with $\mathcal{T}_i = \{t_1^i, \ldots, t_{n_i}^i\} \subseteq \{e_i, \ldots, l_i\}$, representing the time points modeled at node $i$. Given $\mathcal{T}$, the *timed* node set $\mathcal{N}_{\mathcal{T}}$ then has a node $(i, t)$ for each $i \in N_0$ and $t \in \mathcal{T}_i$. (We assume that $\mathcal{T}_0 = \{(0, e_0), (0, l_0)\}$ always.)

The timed arc set of a partially time-expanded network consists of arcs of the form $((i, t), (j, \bar{t}))$, where $(i, j) \in A$, $t \in \mathcal{T}_i$, and $\bar{t} \in \mathcal{T}_j$. Note that arc $((i, t), (j, \bar{t}))$ does *not* have to satisfy $\bar{t} = t + \tau_{ij}$ (even if there is no waiting at $j$). In fact, the flexibility to introduce arcs $((i, t), (j, \bar{t}))$ with a travel time that is different from the actual travel time $\tau_{ij}$ is an essential feature of our partially time-expanded networks, and provides a mechanism to control both the size of the time-expanded network and the approximation properties of the IP model based on it.

For now, we leave $\mathcal{D}_\mathcal{T}$ unspecified, and give a general IP model based on the partially time-expanded network. Let $x_a$ for arc $a \in \mathcal{A}_\mathcal{T}$ be a binary variable indicating whether the arc is used ($x_a = 1$) or not ($x_a = 0$). Then we consider IP formulations, having the objective of minimizing $\sum_{a \in \mathcal{A}_\mathcal{T}} c_a x_a$, which require $x$ to induce a path from $(0, e_0)$ to $(0, l_0)$ in $\mathcal{D}_\mathcal{T}$ that visits all nodes in $N$:

$$\sum_{t \in \mathcal{T}_i} \sum_{a \in \delta^+((i,t))} x_a = 1, \qquad \text{for all } i \in N_0, \tag{9}$$

$$\sum_{a \in \delta^-((i,t))} x_a - \sum_{a \in \delta^+((i,t))} x_a = 0 \quad \text{for all } (i,t) \in \mathcal{N}_\mathcal{T},\ (i,t) \notin \{(0,e_0),(0,l_0)\}, \tag{10}$$

where $\delta^-()$ and $\delta^+()$ denote the set of timed arcs in $\mathcal{A}_\mathcal{T}$ coming into and going out of a timed node. Constraints (9) force one unit of flow to leave each node. Constraints (10) are flow balance constraints forcing the flow into a node at some time point to be equal to the flow out of the node at that time point.

Observe that if the time discretization is complete, so $\mathcal{T}_i = \{e_i, e_i + 1, \ldots, l_i\}$, and $\mathcal{A}_\mathcal{T}$ consists of all $((i,t),(j,\bar{t}))$ with $\bar{t} = \max\{e_j, t + \tau_{ij}\}$ for all $i, j \in N$ and appropriate $t$, together with arcs to and from the depot start and end nodes (as described earlier), then the time-expanded network is acyclic and the above IP formulation is an exact formulation for the TSPTW.

*IP models based on partial time discretization*
As shown by Wang and Regan (2002), by choosing $\mathcal{N}_\mathcal{T}$ and $\mathcal{A}_\mathcal{T}$ carefully, the IP with constraints (9) and (10) may be guaranteed to provide either a lower or an upper bound on the optimal value of the TSPTW.

To obtain a lower bound, the concept of a "short" arc is helpful: $((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}$ is *short* if $\bar{t} \leq \max\{e_j, t + \tau_{ij}\}$. Three conditions, together, guarantee a lower bound from the IP: (i) $(i, e_i) \in \mathcal{N}_\mathcal{T}$ for all $i \in N$, (ii) for all $(i,t) \in \mathcal{N}_\mathcal{T}$ and all $j \in N$ with $t + \tau_{ij} \leq l_j$, there exists $\bar{t}$ with $((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}$, and (iii) every arc in $\mathcal{A}_\mathcal{T}$ is short (Vu et al. 2018). We call a time-expanded network satisfying these properties a *relaxed* (time-expanded) network. Note that if the time discretization at node $j$ in a relaxed network is quite coarse, it may be that $\bar{t}$ is less than $t$, suggesting travel backward in time! Nevertheless, good lower bounds can result. It can be proved that the best such lower bound is obtained by setting $\bar{t} = \max\{t' : t' \leq \max\{e_j, t + \tau_{ij}\},\ (j,t') \in \mathcal{N}_\mathcal{T}\}$, and permitting no other arc from $(i,t)$ to $j$ to be included in $\mathcal{A}_\mathcal{T}$. We say a relaxed network created in this way has the *longest-arc* property. Figure 4 shows a relaxed network for the instance used in Fig. 3 where for each node, $i \in N$, the time points are $\mathcal{T}_i = \{e_i, l_i\}$. Observe that all arcs are directed forward in time, except for the arc $((2,3),(1,2))$ and the arc $((2,5),(3,5))$.

A condition that guarantees an upper bound from the IP, provided the IP is feasible, is that all arcs are "long": $((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}$ is *long* if $\bar{t} \geq \max\{e_j, t + \tau_{ij}\}$. We call a time-expanded network satisfying this properties a *restricted* (time-expanded) network.

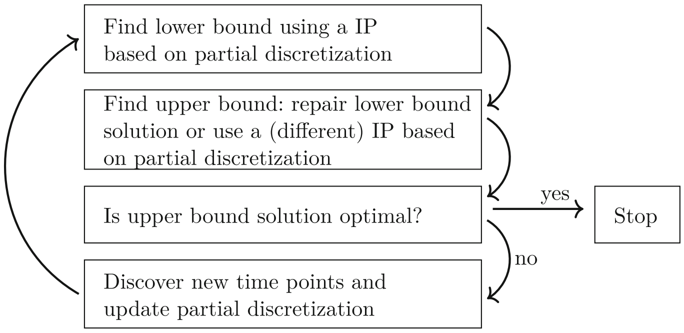
Find lower bound using a IP based on partial discretization
Find upper bound: repair lower bound solution or use a (different) IP based on partial discretization
Is upper bound solution optimal?
yes
Stop
no
Discover new time points and update partial discretization

**Fig. 5** Flowchart of a dynamic discretization discovery algorithm

## 2.2 Dynamic discretization discovery

The fundamental idea underlying the dynamic discretization discovery approach is to work with a partial discretization so as to ensure that the resulting TI models can be solved efficiently, but to guarantee that, upon termination of the algorithm, an optimal (continuous-time) solution is (or can be) produced. That is, the idea is to solve a sequence of small IPs, rather than a single large IP. A flowchart of the high-level structure of a dynamic discretization discovery algorithm is given in Fig. 5.

For TD problems, in general, a dynamic discretization discovery algorithm requires:

- a lower-bound IP model based on a partial discretization that is guaranteed to provide a lower bound on the optimal value of the complete TI model;
- a mechanism that, given an optimal solution to a lower-bound IP model based on a partial discretization, determines whether the solution is feasible and/or optimal for the complete TI model; and
- a mechanism that, given an optimal solution to a lower-bound IP model based on a partial discretization that cannot be converted into a solution that is optimal to the complete TI model, identifies time points that can be added to the partial discretization so that the current optimal solution is no longer feasible for the lower-bound IP model based on the new, refined, partial discretization.

Optionally, it may also use

- an upper-bound IP model based on a partial discretization that is guaranteed to provide an upper bound on the optimal value of a complete TI model, if feasible; and/or
- a mechanism that, given an optimal solution to a lower-bound IP model based on a partial discretization, seeks to "repair" it to improve its feasibility and/or objective value, giving an upper bound on the optimal value of the complete TI model if feasibility is attained.

We have already described how to obtain a lower-bound IP model based on a partial discretization for the TSPTW: use Constraints (9) and (10), defined over a relaxed (time-expanded) network, together with integrality of the variables. The optimal solution to this IP will induce a simple path in the relaxed network that starts and ends at the depot node. If all arcs $((i, t), (j, \bar{t}))$ in the relaxed network have positive "length", so $\bar{t} - t > 0$, then the simple path must visit every node. Otherwise, the IP solution may induce cycles in the relaxed network, which correspond to subtours. If the IP solution does not induce any cycles, then the simple path it induces can easily be checked for feasibility to the TSPTW: step through the node sequence in the path, setting the departure time at each node as early as possible given the true travel times and time windows, and check that it is no later than the end of the time window at the node. If feasibility to the TSPTW is confirmed by this procedure, then the resulting tour must be optimal, since the costs of timed arcs in the relaxed network is precisely their cost in the original network.

The remaining element essential to defining a DDD algorithm for the TSPTW is a mechanism to refine the partial discretization whenever the lower-bound IP model's optimal solution induces cycles or induces a path that violates time window constraints. We discuss that next.

Note that we have already described an optional upper-bound IP model for the TSPTW: using Constraints (9) and (10), defined over a restricted (time-expanded) network, and integrality of the variables.

*Refining a partial time discretization*
Wang and Regan (2002) have shown that using finer and finer (regular) discretizations of time, i.e., partitioning the time windows into $\lceil \frac{l_i - e_i}{\Delta} \rceil$ subwindows for smaller and smaller values of $\Delta$, will ultimately lead to a complete TI model. From a computational efficiency perspective, however, that is not very satisfying, because it requires solving larger and larger IPs, in the end an IP over the full time-expanded network. Therefore, we consider a different strategy for refining the discretization of time.

Consider a solution to the lower-bound IP model that induces cycles. Each of these cycles must use at least one timed arc with non-positive length. Since the true travel times are positive, at least one timed arc in each cycle must be *too short*, meaning that $\bar{t} - t < \tau_{ij}$ for timed arc $((i, t), (j, \bar{t}))$ in the relaxed network. Similarly, in the case that the solution induces some subpath $(((i_1, t_1), (i_2, t_2)), \ldots, ((i_{k-1}, t_{k-1}), (i_k, t_k)))$ in the relaxed network having $e_{i_1} + \sum_{h=1}^{k-1} \tau_{i_h, i_{h+1}} > l_{i_k}$, it must be that at least one of the timed arcs $((i_h, t_h), (i_{h+1}, t_{h+1}))$ is too short.

Thus, whenever the lower-bound IP model does not generate a feasible (and hence optimal) TSPTW solution, its solution uses some timed arc that is too short. By refining the discretization at the node at the head of that arc, the timed arc that is too short can be made to "disappear", i.e., not be present in the relaxed network associated with the refined partial discretization. If the longest-arc property is enforced for the relaxed network constructed at each iteration, the timed arc that is too short, $((i, t), (j, \bar{t}))$ say, can be removed simply by adding $\hat{t}$ to $\mathcal{T}_j$ for any $\hat{t}$ satisfying $\bar{t} < \hat{t} \leq t + \tau_{ij}$. The effect will be to *lengthen* the timed arc, to $((i, t), (j, \hat{t}))$. The natural choice is to take $\hat{t} = t + \tau_{ij}$.

This represents the simplest example of partial time discretization refinement for dynamic discretization discovery. In other contexts, discovering time points to add may be more involved. For example, an LP had to be solved in the case of service network design (Boland et al. 2017). Furthermore, it is often the case that there are many candidate time points to add, as there can be many that will eliminate the current lower-bound IP solution, and any subset of these could be added at each iteration. There might also be time points that seem likely to be needed and so could be added, heuristically. How many and which time points to add in a refinement step affect the efficiency of a dynamic discretization discovery algorithm. Effective implementation strategies for dynamic discretization discovery algorithms are discussed further in Sect. 3.3.

That the dynamic discretization discovery algorithm for the TSPTW outlined above will terminate in a finite number of iterations with an optimal solution as well as a proof that that solution is optimal can be seen as follows. First, note that when the algorithm terminates, it does so with a solution that is optimal, because it has been verified that the solution over the lower bound partially time-expanded network remains feasible when true travel times are used. Second, observe that when the algorithm does not terminate in an iteration, the partially time-expanded network is refined by identifying an arc $((i, t), (j, t'))$ with $t' < t + \tau_{ij}$, i.e., an arc that is too short, and adding timed node $(j, t + \tau_{ij})$. Arcs are then added and removed to ensure that the network remains a relaxed time-expanded network that satisfies the longest-arc property. For example, the arc $((i, t), (j, t'))$ is deleted and replaced by arc $((i, t), (j, t + \tau_{ij}))$. Given that timed nodes are never removed, after a finite number of iterations a complete TI model will have been generated. When that happens, an optimal solution will be found in the next iteration and the algorithm will terminate.

## 3 General principles of dynamic discretization discovery

### 3.1 On the existence of a complete TI model

For DDD to be guaranteed to produce an optimal solution to the continuous time problem, it is sufficient to know that a complete TI model exists for the problem. For some problem classes, the existence of such a model is obvious; in others, it is not.

For the TSPTW with integer travel times and time window limits, the existence is obvious (and, thus, it is also obvious for the TSPTW with rational travel times and time window limits—using a simple scaling argument). A slightly different, and more general, argument proceeds as follows. Any feasible solution to the TSPTW can be viewed as a path $(0, \pi_1, \pi_2, \ldots, \pi_n, 0)$, with $\pi$ a permutation of the node set $N$, departing from the depot at time $e_0$, and departing from each node $\pi_i$ as early as possible, i.e., as soon as the salesman arrives from $\pi_{i-1}$ if the salesman arrives within the time window, or at the opening of the time window if the salesman arrives before the opening of the time window. Now, because there are only finitely many feasible solutions, there exists a TI model based on such earliest departure times at each node in the path for any possible feasible solution. Such a TI model must be complete; a complete TI model must exist. Importantly, this argument does not rely on integer, or even rational, travel times and time window limits. Such a combinatorial argument may suffice in many settings to argue the existence of a complete TI model.

Next, consider the TD-TSPTW. With time-dependent travel times, different objective functions can be considered for the TD-TSPTW. The three natural objectives are (1) to minimize the return time to the depot, (2) to minimize the time away from the depot, i.e., the duration of the tour, and (3) to minimize the travel time, i.e., ignoring any waiting time. The three objectives differ in terms of the "contribution" of waiting time in an optimal tour. Given that the arc travel time functions satisfy the FIFO property, when the goal is to return to the depot as early as possible, it is never beneficial to wait unnecessarily at any location (because of the time windows at locations, it may be necessary to wait until the opening of a window). When the goal is to minimize the duration, the only location where it may be beneficial to wait is the depot itself, i.e., it may be possible to reduce waiting at other locations by departing later from the depot. When the goal is to minimize travel time, it may be beneficial to wait at any location.

When the objective is to return to the depot as early as possible, and it is never beneficial to wait unnecessarily at any location, the combinatorial argument presented above suffices to argue the existence of a complete TI model. However, when (unnecessary) waiting may be beneficial, the argument has to be more involved as it is no longer obvious that the number of feasible solutions is finite. Furthermore, additional properties of the arc travel time functions may be needed. The assumption that the arc travel time functions are piecewise linear is particularly useful, and is commonly made. When the objective is to minimize the duration, for example, the existence of a time-expanded network in which a minimum-cost path from $(0, e_0)$ to $(0, l_0)$ visiting each node in $N$ exactly once corresponds to a continuous-time optimal solution is based on the observation that there is a finite number of paths, that each arc travel time function consists of a finite number of linear pieces, and so has a finite number of breakpoints, and that *there exists an optimal path that departs from at least one node at a breakpoint*.

This was shown formally for the time-dependent shortest path problem (TD-SPP) by Foschini et al. (2014), who consider the problem of finding the minimum duration path when arc travel times are piecewise linear functions satisfying the FIFO property. The intuition behind their result is that if a time-dependent source–sink path does not depart at a breakpoint from at least one node in the path, then the *path travel time function*, formed by function composition of the arc travel time functions along the path to give the travel time on the path as a function of the departure time at the source, is locally linear (diffentiable) at the path's source departure time. Thus, by changing the departure time at every node by a small amount, either forward or backward in time, an improved path is obtained, or a path with the desired property is obtained, i.e., that it departs at a breakpoint from at least one node in the path. This result is extended to the case of the minimum travel time objective in He et al. (2019).

Another setting in which the existence of a complete TI model is not straightforward is the *continuous time inventory routing problem* (Lagos et al. 2018). For ease of exposition, assume that two customers with storage capacity $C_1$ and $C_2$, product usage rate $u_1$ and $u_2$, and initial inventory $I_1^0$ and $I_2^0$, respectively, need to be supplied from a depot with unlimited product supply using vehicles of capacity $Q$ during a planning period $[0, T]$, and that the travel times between the depot and the two customers are $\tau_{01}$ and $\tau_{02}$ and the travel time between the two customer is $\tau_{12}$. The goal is to minimize the transportation cost, which is assumed to be a linear function of the travel time, ensuring that the customers do not run out of product during the planning period. Consider an instance with storage capacities $C_1 = 7$ and $C_2 = 5$, usage rates $u_1 = u_2 = 2$, and initial inventories $I_1^0 = 3$ and $I_2^0 = 5$. The vehicle capacity is $Q = 12$. The travel

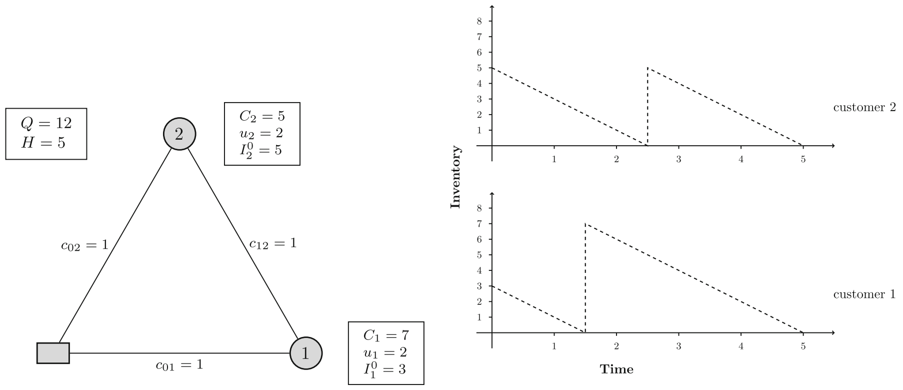

**Fig. 6** Instance with integer data for which the optimal solution has non-integer delivery times

times and costs are identical and symmetric: $\tau_{01} = c_{01} = \tau_{12} = c_{12} = \tau_{02} = c_{02} = 1$. The time horizon is $H = 5$. The optimal solution has a single route of cost 3, visiting Customer 1 at time 1.5, delivering seven units of product, and visiting Customer 2 at time 2.5, delivering five units. This solution is optimal because any feasible solution must visit each customer at least once, and the cheapest way to do this is with a single route. There cannot be an optimal solution in which the deliveries take place only at integer times, since Customer 1 must have a delivery on or before time 1.5, when it runs out of product, and at time 1, Customer 1 only has capacity for six units of product. If the remaining one unit, it requires it to be delivered without incurring extra cost, the vehicle must wait at Customer 1 until time 2 to deliver this unit, at which time it is too late to reach Customer 2 by time 2.5, when Customer 2 runs out of product. Thus the solution with delivery to Customer 1 and time 1.5 and Customer 2 and time 2.5 is the unique optimal solution (Fig. 6).

In this setting, the fact that a complete TI model exists when all instance data are rational relies on the fact that given an optimal set of routes, a linear program can be defined that determines the optimal vehicle departure, customer visit, and return times. The existence now follows from linear programming theory, because a linear program with rational data will have an optimal rational extreme point. The fact that there must exist time-expanded network based on a rational discretization of time in which the optimal solution can be represented, and, thus, found, however, does not mean that we know how to choose the discretization. We only know it exists, which is sufficient to guarantee that a dynamic discretization discovery algorithm will find a continuous time optimal solution.

Existence of a complete TI model has also been explored in the context of a “network scheduling” problem that marries max flow with scheduling: the problem is to schedule arc shutdowns in an arc-capacitated network so as to maximize the total flow over time (Boland et al. 2015). Some arcs have associated shutdown jobs, each with release date, deadline and processing time. A job, once started, cannot be interrupted, and its effect is to change the capacity of its associated arc to zero. The problem is to determine a feasible schedule of start times for all shutdown jobs, so that the integral of the max flow problem in the network over all times in a given time horizon is maximized. When flow cannot be stored at nodes, Boland et al. (2015) give combinatorial arguments to show that a complete TI model exists. They observe that if all job data are integer then the integer discretization supports a complete TI model. By contrast, when flow *can* be stored at nodes having a given storage capacity, even when all job data and node storage capacities are integer, there may not exist any optimal solution that has integer job start times. However, the existence of a rational solution, and hence a complete TI model, is proved by Boland et al. (2015), using the linear programming argument.

### 3.2 General principles in the construction of a lower-bound IP model

Like integer programming in general, constructing an IP model based on a partial discretization of time so as to give a lower bound on the continuous-time optimal value may be an art as much as it is a science. However, some principles can be a useful guide in this endeavor.

From a high level perspective, the principle of "optimism" is key. Any modeling effort involves assumptions, and if an IP is carefully designed to embed only optimistic assumptions, erring on the side of including solutions that may not, in continuous time, be feasible, and erring on the side of underestimating continuous-time costs, then the result will be sure to yield a lower bound.

As an example, consider the case of service network design, in which each commodity has a time window: the commodity becomes available for pick-up at its origin at the start of the time window and must be delivered to its destination by the end. When using a partial discretization, to obtain a lower bound, if the start of a commodity's time window lies between two consecutive time points at its origin, then the commodity should be modeled as becoming available at the *earlier* of the two time points; if the end of its time window lies between two consecutive time points at its destination, then it should be modeled as due only at the *later* of the two times. The "error" in this modeling ensures that no time-feasible path for the commodity is excluded by the model, at the cost of permitting some paths that may not be feasible.

Next, consider the time-dependent shortest path problem with a minimum travel time objective. In this setting, the cost of a timed arc departing node $i$ at a time, $t$, in the partial discretization should represent the least travel time on that arc if departing at any time in the interval from $t$ to the next time at $i$ in the discretization. If the least travel time for one arc occurs at a time *later* than the least travel time for the next arc in a path, then both travel times cannot, in any properly timed solution, occur in the same path. The "error" in this modeling approach ensures that the cost of any path is no greater than its true travel time.

This example highlights another useful concept: that of time *intervals*. Instead of interpreting a variable associated with a time point in a discretization as a decision about an activity to occur *at that time*, it is helpful—for ensuring a lower bound—to interpret such a variable as modeling an activity to occur *at some time* in the interval $[t, t^+)$, where $t$ and $t^+$ are two consecutive times in the discretization. For example, consider a TSPTW model in which $x_{i,j,t}$ represents a departure along arc $(i, j) \in A$ at some time in the interval $[t, t^+)$, where $t$ and $t^+$ are two consecutive times in $\mathcal{T}_i$. This variable should be included in the model if and only if there exists $s \in [t, t^+)$

with $s \in [e_i, \ell_i)$ and $s + \tau_{ij} \leq \ell_j$, i.e., if and only if, based on the time windows for $i$ and $j$ and the travel time from $i$ to $j$, it is possible to travel on the arc $(i, j)$ departing from $i$ at some time in the interval $[t, t+)$ and arrive at $j$ before the end of its time window.

To obtain a lower bound model based on variables modeling decisions about an activity to occur at some time in an interval, the cost of such a variable can be taken to be the minimum cost for the activity it represents over any time in the interval. Constraints can be constructed so as permit anything that *might* occur. For example, for the TSPTW, for a given pair $(j, t)$ with $t \in \mathcal{T}_j$, we may define $\delta^+(j, t)$ to be the set of triples $(j, i, t)$ for which $x_{j,i,t}$ is included in the model. Similarly, we may define $\delta^-(j, t)$ to be the set of triples $(i, j, s)$ for which $x_{i,j,s}$ is included in the model, also having $s + \tau_{ij} < t^+$ and either $t \leq e_j$ or $s^+ + \tau_{ij} \geq t$ (where $s^+$ is next in $\mathcal{T}_i$ after $s$). In other words, $(i, j, s) \in \delta^-(j, t)$ if and only if $x_{i,j,s}$ is defined and there exists $s' \in [s, s+)$ and $t' \in [\max\{e_j, t\}, t^+)$ with $t' \geq s' + \tau_{ij}$. Thus, $(i, j, s) \in \delta^-(j, t)$ implies that it is possible, based on the time windows for $i$ and $j$ and the travel time from $i$ to $j$, to travel on the arc $(i, j)$ departing from $i$ at some time in the interval $[s, s+)$ and then to depart $j$ at some time in the interval $[t, t+)$. Now (the more important side of), the flow balance constraints in the lower-bound model will be

$$\sum_{(j,i,t)\in\delta^+(j,t)} x_{j,i,t} - \sum_{(i,j,s)\in\delta^-(j,t)} x_{i,j,s} \leq 0, \quad \forall j \in N, \ \forall t \in \mathcal{T}_j. \tag{11}$$

This interval interpretation of variables was found to have distinct advantages in the use of dynamic discretization discovery for service network design (Marshall et al. 2018). It was also central to the approach of He et al. (2019) for time-dependent shortest path problems, and advantageous in constructing dual-bound IP models[1] for machine scheduling problems (Boland et al. 2015).

A more general way to view the interval-based approach is as variable *aggregation*. In the TSPTW, for example, suppose binary variable $f^t_{i,j}$ represents travel on arc $(i, j)$ departing at time $t$ for $t$ in a *full* discretization, $\mathcal{T}^* = \{\mathcal{T}^*_i\}_{i\in N_0}$. So the $f^t_{i,j}$ variables form the basis for a complete TI model. Now, for any two consecutive time points, $t, t^+ \in \mathcal{T}_i$ in a partial discretization, $\mathcal{T} = \{\mathcal{T}_i\}_{i\in N_0}$, define

$$x^t_{i,j} := \sum_{t'\in\mathcal{T}^*_i : t\leq t'<t^+} f^{t'}_{i,j}.$$

Then, since $\sum_{t\in\mathcal{T}^*_i} f^t_{i,j} \leq 1$ is implied in the complete TI model, it must be that $x^t_{i,j}$ is binary and equals 1 if and only if the tour uses arc $(i, j)$ departing at some time in $[t, t^+)$, which is precisely the interval approach described above.

We show how to derive the constraints of the lower-bound IP model by means of a small example. Consider the partial network shown in Fig. 7 with nodes $a, b, c, d \in N$ and arcs $(a, c), (b, c), (c, d) \in A$, where we assume that these are the only arcs incident

[1] For minimization problems, dual bounds are lower bounds, which is the prevalent dual bound considered in this paper. However, the problem studied by Boland et al. (2015) has a maximization objective, so its dual bounds are upper bounds.

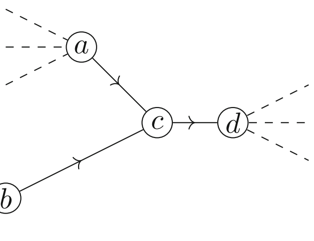

**Fig. 7** Partial network

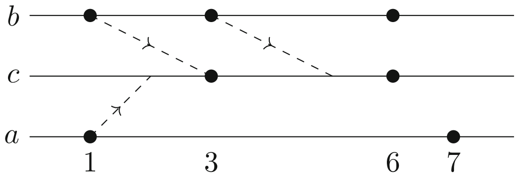

**Fig. 8** Partial partially time-expanded network

to node $c$, and (relevant) travel times $\tau_{ac} = 1$ and $\tau_{bc} = 2$. Furthermore, let the time windows of node $a$, $b$, $c$, and $d$ be $[1, \cdot]$, $[1, \cdot]$, $[3, \cdot]$, and $[1, \cdot]$, respectively, where $\cdot$ indicates that the end of the time window is far enough in the future so as not to be relevant for the example. Let the time points in the partially time-expanded network for nodes $a$, $b$, and $c$ be $\mathcal{T}_a = \{1, 7, \ldots\}$, $\mathcal{T}_b = \{1, 3, 6, \ldots\}$, and $\mathcal{T}_c = \{3, 6, \ldots\}$, as shown in Fig. 8. The relevant aggregated variables are

$$\begin{aligned}
x^1_{ac} &= f^1_{ac} + f^2_{ac} + f^3_{ac} + f^4_{ac} + f^5_{ac} + f^6_{ac},\\
x^1_{bc} &= f^1_{bc} + f^2_{bc},\\
x^3_{bc} &= f^3_{bc} + f^4_{bc} + f^5_{bc}, \text{ and}\\
x^3_{cd} &= f^3_{cd} + f^4_{cd} + f^5_{cd}.
\end{aligned}$$

The complete TI model has the following flow balance constraints for nodes $(c, 3)$, $(c, 4)$ and $(c, 5)$:

$$\begin{aligned}
f^3_{cd} &= f^1_{ac} + f^2_{ac} + f^1_{bc},\\
f^4_{cd} &= f^3_{ac} + f^2_{bc}, \text{ and}\\
f^5_{cd} &= f^4_{ac} + f^3_{bc}.
\end{aligned}$$

Aggregating these constraints, i.e., aggregating the constraints in the interval $[3, 6)$ at node $c$, gives

$$f^3_{cd} + f^4_{cd} + f^5_{cd} = f^1_{ac} + f^2_{ac} + f^3_{ac} + f^4_{ac} + f^1_{bc} + f^2_{bc} + f^3_{bc},$$

which can be relaxed to

$$f^3_{cd}+f^4_{cd}+f^5_{cd} \le f^1_{ac}+f^2_{ac}+f^3_{ac}+f^4_{ac}+f^5_{ac}+f^6_{ac}+f^1_{bc}+f^2_{bc}+f^3_{bc}+f^4_{bc}+f^5_{bc},$$

which is equivalent to

$$x^3_{cd} \le x^1_{ac} + x^1_{bc} + x^3_{bc}.$$

This is the constraint of the form (11) for $(c, 3)$ in the lower-bound IP model (associated with the partially time-expanded network). That is, by aggregating variables and constraints over the intervals, we can obtain the (interval-based) lower-bound IP model.

For a general complete TI model, say one having

- variables $\mathbf{f}_{i,t}$ (a vector of variables) for each $i$ and each $t \in \mathcal{T}_i^*$,
- objective minimize $\sum_i \sum_{t \in \mathcal{T}_i^*} (\mathbf{c}^{i,t})^T \mathbf{f}_{i,t}$, where each $\mathbf{c}^{i,t}$ is a vector, and
- constraints $\sum_i \sum_{t \in \mathcal{T}_i^*} A^{i,t} \mathbf{f}_{i,t} \geq \mathbf{b}$, where each $A^{i,t}$ is a matrix and $\mathbf{b}$ is a vector,

we may construct a lower-bound model based on a partial discretization, $\mathcal{T}$, as follows. Define variables $\mathbf{x}_{i,t} := \sum_{t' \in \mathcal{T}_i^* : t \leq t' < t^+} \mathbf{f}_{\mathbf{i},\mathbf{t}'}$ for each $i$ and $t \in \mathcal{T}_i$. Here, again, we use the notation $t^+$ to denote the next element after $t$ in $\mathcal{T}_i$. Now for simplicity, assume all variables are nonnegative. For every $i$, $t \in \mathcal{T}_i$, and variable index $j$, set $\underline{c}_j^{i,t} := \min_{t' \in \mathcal{T}_i^* : t \leq t' < t^+} c_j^{i,t}$ and $\overline{A}_{j,k}^{i,t} := \max_{t' \in \mathcal{T}_i^* : t \leq t' < t^+} A_{j,k}^{i,t'}$ for all constraints $k$. Then the model with

- objective minimize $\sum_i \sum_{t \in \mathcal{T}_i} (\underline{\mathbf{c}}^{i,t})^T \mathbf{x}_{i,t}$, and
- constraints $\sum_i \sum_{t \in \mathcal{T}_i} \overline{A}^{i,t} \mathbf{x}_{i,t} \geq \mathbf{b}$

will yield a lower bound. We have, so far, omitted the constraint aggregation step that we used for the TSPTW example to ensure that the left-hand side of the constraint included the whole set of $f$ variables aggregated to form an $x$ variable. Unfortunately, without that step, the model is likely to be very weak, due to cases of $t' \in [t, t^+)$ with $A_{j,k}^{i,t'} < \overline{A}_{j,k}^{i,t}$ having a "large" difference between the right and left sides. Constraint aggregation is needed to "even up" the coefficients of $f$ variables that are to be aggregated to a form single $x$ variable. There is no clear best strategy, or mechanism, to automating such constraint aggregation, so this general principle has, at present, limited applicability; it seems that, for now, arriving at constraints for a lower-bound model will be an art more than it will be a science.

Finally, we observe that even if a complete TI model is not known to exist, the interval-based modeling principle can still be applied. For example, in the continuous-time inventory routing problem (Lagos et al. 2018), the delivery quantity variable for a customer and time interval represents the total quantity delivered to the customer over the interval, implicitly integrating a delivery quantity "variable" that is a continuous function of time.

## 3.3 Algorithm engineering

Even though the underlying ideas of a dynamic discretization discovery algorithm are natural and intuitive, developing an effective and efficient implementation involves a certain amount of algorithm engineering. As mentioned above, the idea underlying a dynamic discretization discovery algorithm idea is to solve a sequence of small IPs, rather than a single large IP. This is effective if (1) the number of IPs solved is not too large, and (2) the IPs are solved in a reasonable amount of time. Algorithm engineering mostly focuses on finding a proper balance between the two.

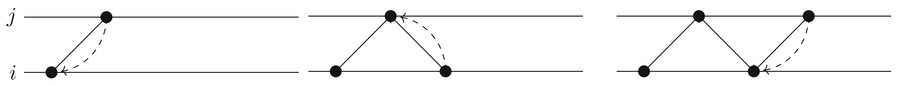

**Fig. 9** Cycle

*Adding multiple time points when refining the partially time-expanded network* Time points are added to the partially time-expanded network to “fix” a problem, e.g., in the TSPTW, the use of (existence of) an arc that is too short. On the one hand, fixing multiple problems at once may reduce the number of iterations. On the other hand, adding multiple time points when refining the partially time-expanded network increases the size of IPs that need to be solved. Furthermore, by fixing one problem, it is possible that other problems disappear naturally. Therefore, care has to be taken when developing strategies for adding multiple time points when refining the partially time-expanded network. Depending on the problem, it may be possible to recognize that two fixes are “independent”, i.e., that fixing one will not fix the other. In that case, it is likely that fixing both of them at the same time is beneficial. In the TSPTW, for example, if the solution consists of a path and multiple cycles, it is likely that trying to fix all the cycles will reduce the number of iterations.

Again, depending on the problem, it may be possible to fix a problem in different ways, i.e., by adding a different number of time points when refining the partially time-expanded network. In that case, it is likely that choosing the fix that requires the fewest time points to be added is preferable.

*Adding cuts rather than refining the partially time-expanded network* In certain situations, it may be advantageous to eliminating solutions by adding cuts rather than adding time points. Consider the situation illustrated in Fig. 9 with two nodes $i$ and $j$ and a travel time of 1 between nodes $i$ and $j$. In the left-most diagram, the dashed arc represents an arc that is too short and creates a cycle. That cycle can be removed by adding a time point, as shown in the center diagram, but that creates another cycle. That cycle, in turn, can be removed by adding another time point, as shown in the right most diagram, but that creates another cycle, and so on. This, of course, can be anticipated and many time points can be added at once, but it is better to add a constraint that eliminates these cycles. Specifically, the following constraint eliminates all timed copies of a cycle $C$

$$\sum_{(i,j)\in C\times C}\ \sum_{t,t':((i,t),(j,t'))\in\mathcal{A}_{\mathcal{T}}} x_{((i,t),(j,t'))} \leq |C| - 1.$$

*Adding cuts to strengthen the IP model* The enormous power of modern (commercial) IP solvers is due, in part, because of their ability to automatically generate cuts to strengthen an IP model. However, a modeler’s understanding of the problem at hand may often allow the modeler to develop valid inequalities that cannot be automatically detected and adding these may often result in reduced computation times.

*Dynamically updating the optimality tolerance of the IP model* In practice, it is often the case that for the first several iterations of a dynamic discretization discovery algorithm, there is a large gap between the upper and lower bounds. Thus, there is little benefit to using a small optimality tolerance in the IP solver when finding the next lower bound. Instead of having a fixed tolerance across all iterations, it often is more efficient to adapt the tolerance as the algorithm progresses.

*Updating the IP model in memory* Rather than regenerating the IP model in each iteration of the algorithm (after the partially time-expanded network has been refined), it is more efficient to update the IP model in memory, i.e., reusing variables and constraints as much as possible (e.g., relabel variables instead of deleting and adding variables). This also provides the IP solver with the opportunity to perform a warm start at the next iteration.

Marshall et al. (2018) provide an example of the potential benefits of algorithm engineering. In addition to introducing an interval-based view, rather than a time point-based view, of dynamic discretization discovery, the authors explore, in the context of the continuous-time service network design problem, various techniques to enhance the efficiency of their implementation of a dynamic discretization discovery algorithm. The combined benefit of these techniques is that high-quality or optimal solutions are often found earlier in the search and proving optimality can often be done orders of magnitude faster, compared to the original dynamic discretization discovery algorithm introduced by Boland et al. (2017).

Table 1 shows the results for the dynamic discretization discovery algorithm of Boland et al. (2017) (DDD) and the interval-based dynamic discretization discovery algorithm of Marshall et al. (2018) (DDDI) on five instances of the continuous-time service network design problem which were created using data from a large less-than-truckload carrier in the USA, restricted to the states listed. These were the most challenging instances presented in Boland et al. (2017), with $|N|$ denoting the number of nodes in the physical network, $|A|$ denoting the number of arcs in the physical network, and $|K|$ denoting the number of commodities that need to be routed through the network. Note that computation was limited to 2 h. The gap reported represents the integrality gap at termination (where the goal was to solve the instances to within 1% of optimality). The results clearly show that DDDI outperforms DDD and the full discretization (FD) for these instances. Not only does DDDI find high quality solutions quickly, but it is also able to prove quickly that these solution are of high quality. Though DDD is also able to find high-quality solutions, it is unable to prove that they are of high quality.

In the case study used to introduce the key elements and concepts of dynamic discretization discovery algorithms, i.e., the TSPTW, we mentioned that an advantage of using a TI model is that it is (relatively) easy to incorporate time-dependent travel times. This has been explored in Vu et al. (2018) and the dynamic discretization discovery algorithm developed and presented is superior to existing algorithms for solving instances of the TD-TSPTW, e.g., those presented in Arigliano et al. (2018) and Montero et al. (2017). Table 2 compares its performance to the performance of the branch-and-cut algorithm of Montero et al. (2017), denoted by TTBF-BC, on a set of standard benchmark instances, where “Inst” represents the number of instances in a class, “Slv” records the number of instances solved, and “Time” records the time needed to solve those instances (reported in seconds and averaged over the instances solved).

**Table 1** Performance of DDD and DDDI on instances derived from a US less-than-truckload carrier

| States | $\|N\|$ | $\|A\|$ | $\|K\|$ | FD | | | DDD | | | DDDI | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | Time (s) | Value | Gap (%) | Time (s) | Value | Gap (%) | Time (s) | Value | Gap (%) |
| ID, MT, OR, WA | 10 | 54 | 224 | 213 | 1,503,240 | 0.94 | 30 | 1,507,222 | 0.92 | 5 | 1,510,629 | 0.93 |
| CO, ID, MT, OR, WA | 14 | 81 | 341 | 7200 | 1,780,041 | 3.63 | 7200 | 1,757,287 | 1.68 | 78 | 1,757,395 | 0.69 |
| CO, ID, MT, OR, WA, NV | 16 | 109 | 469 | 7200 | 2,939,430 | 25.28 | 7200 | 2,291,691 | 2.74 | 803 | 2,273,707 | 0.65 |
| CO, ID, MT, OR, WA, UT | 15 | 104 | 458 | 7200 | 2,805,518 | 19.71 | 7200 | 2,325,602 | 1.45 | 223 | 2,320,710 | 0.78 |
| ID, MT, OR, WA, NV, UT | 13 | 97 | 429 | 7200 | 3,130,964 | 23.78 | 7200 | 2,501,827 | 3.00 | 137 | 2,493,752 | 0.89 |

**Table 2** Performance on Arigliano et al. (2018) instances

| Instance set | Traffic pattern | TTBF-BC | | | DDD | | |
|---|---|---|---|---|---|---|---|
| | | Inst | Slv | Time | Inst | Slv | Time |
| Set 1 | A | 478 | 470 | 31.99 | 478 | 478 | 12.91 |
| | B | 474 | 470 | 20.44 | 474 | 474 | 8.79 |
| Set *w100* | A | 108 | 107 | 100.13 | 480 | 480 | 1.31 |
| | B | 108 | 107 | 93.10 | 480 | 480 | 1.14 |

We see that the dynamic discretization discovery algorithm (DDD) solves all instances of Set 1, whereas TTBF-BC struggles with a few of them, and that DDD is faster. Furthermore, DDD solves all w100 instances and in far less time (where the solve times reported for DDD for w100 instances includes solve times for instances that TTBF-BC was unable to solve).

## 4 Exact models based on a partial discretization of time

Rather that using a partial discretization of time to construct a model that provides a lower or upper bound, another stream of research focuses on exact IP models based on a partial discretization of time. These ideas have primarily been explored in the context of lot sizing models Belvaux and Wolsey (2000, 2001). In this context, the interval of time between two consecutive time points in a discretization is referred to as a *bucket*. A distinction is made between (i) big bucket models, in which a relatively small number of buckets is needed, but many events can occur within a time interval, leading to difficulties in modeling, and (ii) small bucket models, in which modeling in detail what happens within, or between, time intervals, is easy, but which require a relatively large number of buckets. Bucket models, so far, have received little attention outside of lot sizing. The idea has been applied in the context of machine scheduling by Boland et al. (2016), who developed a small bucket model, and Baptiste and Sadykov (2009), who developed a big bucket model Baptiste and Sadykov (2009), which exploit special properties of the particular scheduling problem to overcome the modeling difficulty. These two bucket models consistently enable larger instances to be solved than is possible with other models.

To illustrate how a bucket model based on a partial discretization of time can be constructed, we again consider the TSPTW. Take a relaxed network, $\mathcal{D}_\mathcal{T}$, with the additional property that $\mathcal{A}_\mathcal{T}$ contains arc $((i,t),(j,\bar{t}))$ whenever it is possible to depart at some time in the bucket with lower end point $t$ at node $i$ and arrive at some time in the bucket with lower end point $\bar{t}$ at node $j$. In this case, we say that $\mathcal{D}_\mathcal{T}$ is a *bucket network*. We then introduce semi-continuous variables $d_a$, modeling the departure time at the tail of $a \in \mathcal{A}_\mathcal{T}$; if $a$ is used in the tour, and $d_a = 0$ otherwise, and add the constraints:

$$\sum_{a\in\delta^-((i,t))} (d_a + \tau_a x_a) = \sum_{a\in\delta^+((i,t))} d_a, \qquad \forall (i,t) \in \mathcal{N}_\mathcal{T}, i \neq 0, t \neq e_i \tag{12}$$

$$\sum_{a\in\delta^-((i,e_i))} (d_a + \tau_a x_a) \leq \sum_{a\in\delta^+((i,e_i))} d_a, \qquad \forall (i,t) \in \mathcal{N}_\mathcal{T}, i \neq 0, \tag{13}$$

$$\underline{M}_a x_a \leq d_a \leq \overline{M}_a x_a \qquad \forall a \in \mathcal{A}_\mathcal{T}. \tag{14}$$

Here $\tau_a = \tau_{ij}$ for $a = ((i,t),(j,\bar{t})) \in \mathcal{A}_\mathcal{T}$, and $\underline{M}_a$ and $\overline{M}_a$ are given by

$$\underline{M}_a = \max\{t, \bar{t} - 1 - \tau_{ij}\} \quad \text{and} \quad \overline{M}_a = \min\{t' - 1, \bar{t}' - 1 - \tau_{ij}\}. \tag{15}$$

Together, Constraints (9), (10) and (12)–(14) define an exact model. The strength of the model is dependent on the big-$M$ constraints, (14): the "tighter" the values of $\underline{M}_a$ and $\overline{M}_a$, the stronger the model is likely to be. Clearly, if a partial discretization is refined, so that $t' - t$ and $\bar{t}' - \bar{t}$ decrease, the model strengthens. In the extreme case, $t' - t = 1$ and $\bar{t}' - \bar{t} = 1$ and Constraints (14) become $d_a = tx_a$.

An alternative way to construct an exact IP model based on a partial discretization of time is to start from an IP based on a relaxed network, whose solution may contain subtours or time window-infeasible subpaths, and then "correct it" by the addition of new variables and/or inequalities. One particular approach is to "intersect" the partially time-expanded network formulation, based on $x_a$ variables for $a \in \mathcal{A}_\mathcal{T}$, with another, valid, TSPTW formulation that includes binary variables $y_{ij}$ for each $(i,j) \in A$ to indicate if arc $(i,j)$ is used or not, via

$$y_{ij} = \sum_{((i,t),(j,\bar{t}))\in\mathcal{A}_\mathcal{T}} x_a.$$

The conjunction of the valid and lower bound IP models can be interpreted as an extended formulation of the former. This is the approach taken by Dash et al. (2012), where the partially time-expanded network is obtained with an iterative LP-based heuristic, and the extended formulation is solved using branch and cut.

## 5 Implicit dynamic discretization via Lagrangian relaxation and dynamic programming

An alternative approach to solving a complete TI model starts by considering how to solve its LP relaxation. To illustrate, consider the LP relaxation of the complete TI model for TSPTW formed from Constraints (9) and (10), together with non-negativity of the variables, where $\mathcal{T}$ is taken to be the full discretization. Solving this LP directly will be impossible when the number of time points in the full discretization is large. Thus alternative techniques are required.

One such technique is to form the Lagrangian relaxation of the constraints that require each node to be visited, Constraints (9), (except for the case $i = 0$), and then solve the corresponding Lagrangian dual problem, either exactly or approximately. The Lagrangian relaxation subproblem is to find a shortest path from $(0, e_0)$ to $(0, \ell_0)$

in the fully time-expanded network, where the length of arc $((i, t), (j, \bar{t}))$ $(i \neq 0)$ is $c_{i,j} - \lambda_i$, for $\lambda_i$ the Lagrange multiplier of the node visit requirement constraint for $i$. The Lagrangian dual variables are unrestricted in sign, and are likely to create negative arc lengths: to reduce violation of node $i$'s visit requirement constraint, we can expect the Lagrangian dual solution method to encourage $\lambda_i$ to increase beyond $c_{i,j}$ for some $j$.

This Lagrangian relaxation subproblem can readily be solved by dynamic programming techniques, without the need to explicitly create the full time-expanded network. It has the form of a resource-constrained shortest path problem, for which there are highly effective exact and heuristic methods available (see Pugliese and Guerriero (2013); also see the work of Irnich and Desaulniers (2005) and Irnich (2008)). The dynamic program will create node labels of the form $(i, t, r)$, where $r$ is the length of the shortest path from 0 to $i$ arriving at a time no later than $t$. Only nondominated labels are retained by the algorithm; in practice, the set of labels for each node $i$ is expected to be sparse relative to the full discretization. Note that for positive travel times, the dynamic program must terminate finitely, and the path it generates in the fully time-expanded network will be simple (elementary). However, it may visit multiple copies of the same underlying node; the path induced in the original network may include cycles.

This approach is taken by Mahmoudi and Zhou (2016) (within a more complex vehicle pick-up and delivery context), where the Lagrangian dual problem is solved heuristically by subgradient optimization. It is also used for solving parallel machine scheduling problems by Pessoa et al. (2010), who solved the Lagrangian dual to optimality via its equivalent primal, column generation, reformulation. For the TSPTW, the lower bounding procedure of Baldacci et al. (2012) also uses a column generation reformulation to solve a Lagrangian dual problem like that described above, but strengthens the subproblem to (partially) eliminate cycles.

A strong attraction for this approach is that it implicitly solves the LP relaxation of the complete TI model, which enables reduced cost variable fixing to be applied. In the case of the TSPTW, the LP dual variables needed for reduced cost variable fixing come from the Lagrange multipliers, the $\lambda_i$, for the Constraints (9), and from the dynamic programming labels, $(i, t, r_{i,t})$, for Constraints (10). The value $r_{i,t}$ is, in effect, the dual multiplier for the flow balance constraint for $(i, t)$, so the reduced cost of the variable $x_{((i,t),(j,\bar{t}))}$ is $c_{i,j} - \lambda_i - (r_{j,\bar{t}} - r_{i,t})$. Importantly, if $s$ and $s^+$ are chronologically consecutive in the list of labels generated by the dynamic program for node $i$, and $t$, $t^+$ are consecutive in the labels for node $j$, then $c_{i,j} - \lambda_i - (r_{j,t} - r_{i,s})$ is the reduced cost of $x_{((i,s'),(j,t'))}$ for any $s' = s, \dots, s^+ - 1$ with $t' = s' + \tau_{ij} \in [t, t^+)$. Thus the logic of reduced cost variable fixing may be used to infer time *intervals* over which an arc need not be considered.

This kind of reduced cost variable fixing was used to great effect by both Pessoa et al. (2010) and Baldacci et al. (2012), building on the foundations in the work of Irnich et al. (2010).

Solving the LP relaxation does not, of course, solve the complete TI model as an IP; integrality is still required. In the case that column generation is used to solve the primal form of the Lagrangian dual problem, as in the paper of Pessoa et al. (2010), branch and price is the natural approach to obtaining integrality. Baldacci et al. (2012) use dynamic programming in which a state consists of a set of nodes, a final node visited in a time-feasible Hamiltonian path on the set, and a time at which the final node is reached. Essential to the computational success of this method is their lower-bounding procedure, described above, which is used to fathom states.

## 6 Related research

Time-expanded networks are an example of so-called *layered graphs*. Layered graphs enumerate (possibly after discretizing) the possible values of one or more characteristics of a problem and embed these values in the definition of the nodes of a graph. The model proposed by Picard and Queyranne (1978) for the time-dependent traveling salesman problem, mentioned earlier, uses a layered graph in which the *position* of the city in a tour is enumerated, i.e., a node of the form $(i, k)$ represents visiting city $i$ as the $k^{th}$ city in the tour. This allows modeling of "time-dependent" costs, i.e., a cost for going from city $i$ to city $j$ that depends on the position of $i$ in the tour. In capacitated routing problems, it is possible to enumerate the values of the load of a vehicle when it visits a location, i.e., a node of the form $(i, q)$ represents visiting a location $i$ with the load of the vehicle upon arrival at the location being $q$ (for an example, see Gouveia and Ruthmair 2015). Gouveia et al. (2019) provide an insightful and comprehensive survey of layered graph formulations for modeling and solving combinatorial optimization problems. Riedler et al. (2018) discuss refinement strategies for layered graphs and use, as we have done, the TSPTW as an illustrative example.

Clautiaux et al. (2017) present an iterative aggregation and disaggregation algorithm for problems that can be formulated as a circulation model with side constraints. Their starting point is similar to ours. They consider network flow model in which the number of nodes is large as the models are "characterized by nodes that correspond to values in a given scale". A discretization of time is a natural example of "values in a given scale", but the concept, as is the case with layered graphs, is more general. An aggregation occurs when certain values from an ordered set representing the values of a scale are removed and a disaggregation occurs when certain values are added to an ordered set representing the values of a scale. A distinction is made between *conservative* aggregation (over-constrained in the terminology of Wang and Regan 2002) and *heuristic* aggregation (under-constrained in the terminology of Wang and Regan 2002). Whereas dynamic discretization discovery focuses on refinement, i.e., disaggregation, the algorithm of Clautiaux et al. (2017) relies heavily on both aggregation and disaggregation. It is observed that checking the feasibility of a solution in the aggregated network is NP-hard.

The concept of partially time-expanded networks and dynamically refining partially time-expanded networks is also present in the work of Fischer and Helmberg (2014), which focus on discrete optimization problems in which the progress of objects over time is modeled by the shortest path problems in time-expanded networks.

Discretization of time has been examined in depth in the study of flows over time and related problems (Anderson et al. 1982; Philpott 1990; Philpott and Craddock 1995; Skutella 2009). It has been shown that a number of problems in continuous time, but not all, can be solved by discretizing time. Results are primarily theoretical in nature and it is observed that the pseudo-polynomial size of time-discretized formulations is likely to pose significant challenges computationally.

Other than the papers already mentioned in previous sections, there has been much prior work on the TSPTW, e.g., Christofides et al. (1981), Baker (1983), Savelsbergh (1985), Langevin et al. (1993), Pesant et al. (1998), Ascheuer et al. (2001), Focacci et al. (2002), and Kara and Derya (2015), and some prior work on the time-dependent TSP, e.g., Abeledo et al. (2013) and the time-dependent TSPTW, e.g., Albiach et al. (2008).

## 7 Research opportunities

The opportunities for further research on time-dependent integer programs are many, and we mention only a few here.

Dynamic discretization discovery is essentially a dual method. Are there primal variants, in which either infeasibility of an upper-bound IP model or suboptimality of its solution can be detected, and new time points determined whose addition would "cut off" the current, primal, solution?

Algorithm engineering can greatly enhance the computational performance of a dynamic discretization discovery algorithm. For example, in case of the TSPTW, the choice to add cutting planes rather than time points to eliminate (short) cycles in solutions to the lower-bound model was critical to its success. When is it best to cut and when is it best to refine the time discretization? What is the interplay between the roles of cuts and time points? There is, of course, a third way to address infeasibility in a lower bound solution: branch. Can effective branching rules be developed to complement the use of cutting planes and time discretization refinement in dynamic discretization discovery algorithms?

In dynamic discretization discovery algorithms, similar IPs are solved repeatedly. When time discretization refinement is viewed as variable disaggregation, it is clear that valid inequalities for one IP can readily be transferred to the next. Preprocessing information is also likely to be transferable, especially variable fixing or strengthened variable bounds. Can IP solver technology be developed to accommodate and exploit variable mapping information provided by users when they are solving IPs sequentially?

The implicit solution of a Lagrangian relaxation of a complete TI model via dynamic programming seems to be a particularly powerful idea. However, all the approaches developed along these lines, to date, use a Lagrangian relaxation subproblem that is a shortest path problem in the full time-expanded network. How far can this be generalized? How to do this, for example, when travel times are time dependent? Are there other problems, with quite different structure, to which similar concepts might apply? And how can this idea be exploited and/or integrated with the dynamic discretization discovery framework?

To conclude, we mention an aspect that is particularly intriguing. The contribution of delaying an activity to the problem objective appears to have a big impact on the design of effective discretization discovery algorithms. In the case of service network design, if a storage cost is incurred when a commodity waits at an intermediate node to be consolidated with other commodities on a later, outgoing, arc, then it is not at all clear how to design a finitely terminating dynamic discretization discovery algorithm. A similar issue arises in the case of continuous time inventory routing problems, where delaying a delivery allows more inventory to be consumed at the customer, making space for a larger quantity to be delivered on a vehicle trip, possibly reducing the number of trips that need to be made. In the case of time-dependent shortest path problems, when waiting can be advantageous to the objective, deep insight about the structure of optimal solutions is needed to permit finite dynamic discretization discovery algorithms to be designed. Without such insight, researchers have so far resorted to methods that are only convergent, using predefined tolerances to test for feasibility and/or optimality, in practice. New understanding and new ideas are needed to address this issue.

**Acknowledgements** This material is based upon work supported by the National Science Foundation under Grant no. 1662848.

## References

Abeledo H, Fukasawa R, Pessao A, Uchoa E (2013) The time dependent traveling salesman problem: polyhedra and algorithm. Math Program Comput 5:27–55

Albiach J, Sanchis J, Soler D (2008) An asymmetric TSP with time windows and with time-dependent travel times and costs: an exact solution through a graph transformation. Eur J Oper Res 189:789–802

Anderson E, Nash P, Philpott A (1982) A class of continuous network flow problems. Math Oper Res 7:501–514

Arigliano A, Ghiani G, Grieco A, Guerriero E, Plana I (2018) Time-dependent asymmetric traveling salesman problem with time windows: properties and an exact algorithm. Discrete Appl Math. https://doi.org/10.1016/j.dam.2018.09.017 In press

Ascheuer N, Fischetti M, Grötschel M (2001) Solving the asymmetric travelling salesman problem with time windows by branch-and-cut. Math Program 90(3):475–506

Baker EK (1983) Technical note—an exact algorithm for the time-constrained traveling salesman problem. Oper Res 31(5):938–945

Baldacci R, Mingozzi A, Roberti R (2012) New state-space relaxations for solving the traveling salesman problem with time windows. INFORMS J Comput 24:356–371

Baptiste P, Sadykov R (2009) On scheduling a single machine to minimize a piecewise linear objective function: a compact IP formulation. Naval Res Logist 56:487–502

Belvaux G, Wolsey L (2000) bc-prod: a specialized branch-and-cut system for lot-sizing problems. Manag Sci 46:724–738

Belvaux G, Wolsey L (2001) Modelling practical lot-sizing problems as mixed-integer programs. Manag Sci 47:993–1007

Boland N, Kalinowski T, Kaur S (2015) Scheduling network maintenance jobs with release dates and deadlines to maximize total flow over time: bounds and solution strategies. Comput Oper Res 64:113–129

Boland N, Clement R, Waterer H (2016) A bucket indexed formulation for nonpreemptive single machine scheduling problems. INFORMS J Comput 28:14–30

Boland N, Hewitt M, Marshall L, Savelsbergh M (2017) The continuous time service network design problem. Oper Res 65:1303–1321

Christofides N, Mingozzi A, Toth P (1981) State-space relaxation procedures for the computation of bounds to routing problems. Networks 11(2):145–164

Clautiaux F, Hanafi S, Macedo R, Voge M-E, Alves C (2017) Iterative aggregation and disaggregation algorithm for pseudo-polynomial network flow models with side constraints. Eur J Oper Res 258:467–477

Dash S, Günlük O, Lodi A, Tramontani A (2012) A time bucket formulation for the travelling salesman problem with time windows. INFORMS J Comput 24:132–147

Fischer F, Helmberg C (2014) Dynamic graph generation for the shortest path problem in time-expanded networks. Math Program Ser A 143:1–16

Focacci F, Lodi A, Milano M (2002) A hybrid exact algorithm for the TSPTW. INFORMS J Comput 14(4):403–417

Foschini L, Hershberger J, Suri S (2014) On the complexity of time-dependent shortest paths. Algorithmica 68(4):1075–1097

Gouveia L, Ruthmair M (2015) Load-dependent and precedence-based models for pickup and delivery problems. Comput Oper Res 63:56–71

Gouveia L, Leitner M, Ruthmair M (2019) Layered graph approaches for combinatorial optimization problems. Comput Oper Res 102:22–38

He E, Boland N, Nemhauser G, Savelsbergh M (2019) Dynamic discretization discovery algorithms for time-dependent shortest path problems. Optimization Online 2019-7082

Irnich S (2008) Resource extension functions: properties, inversion, and generalization to segments. OR Spectr 30(1):113–148

Irnich S, Desaulniers G (2005) Shortest path problems with resource constraints. In: Desaulniers G, Desrosiers J, Solomon MM (eds) Column generation. Springer, Boston, MA

Irnich S, Desaulniers G, Desrosiers J, Hadjar A (2010) Path-reduced costs for eliminating arcs in routing and scheduling. INFORMS J Comput 22(2):297–313

Kara I, Derya T (2015) Formulations for minimizing tour duration of the traveling salesman problem with time windows. Procedia Econ Finance 26:1026–1034

Lagos F, Boland N, Savelsbergh M (2018) The continuous time inventory routing problem. Optimization Online 2018-6439

Langevin A, Desrochers M, Desrosiers J, Gélinas S, Soumis F (1993) A two-commodity flow formulation for the traveling salesman and the makespan problems with time windows. Networks 23(7):631–640

Mahmoudi M, Zhou X (2016) Finding optimal solutions for vehicle routing problem with pickup and delivery services with time windows: a dynamic programming approach based on state-space-time network representations. Transp Res Part B Methodol 89:19–42

Marshall L, Boland N, Hewitt M, Savelsbergh M (2018) Interval-based dynamic discretization discovery for solving the continuous-time service network design problem. Optimization Online 2018-6883

Montero A, Méndez-Dıaz I, Miranda-Bront J (2017) An integer programming approach for the time-dependent traveling salesman problem with time windows. Comput Oper Res 88:280–289

Pesant G, Gendreau M, Potvin J, Rousseau J (1998) An exact constraint logic programming algorithm for the traveling salesman problem with time windows. Transp Sci 32(1):12–29

Pessoa A, Uchoa E, Poggi de Aragao M, Rodrigues R (2010) Exact algorithm over an arc-time indexed formulation for parallel machine scheduling problems. Math Program Comput 2:259–290

Philpott AB (1990) Continuous-time flows in networks. Math Oper Res 15:640–661

Philpott A, Craddock M (1995) An adaptive discretization algorithm for a class of continuous knapsack problems. Networks 26:1–11

Picard J-C, Queyranne M (1978) The time-dependent traveling salesman problem and its application to the tardiness problem in one-machine scheduling. Oper Res 26:86–110

Pugliese LDP, Guerriero F (2013) A survey of resource constrained shortest path problems: exact solution approaches. Networks 62(3):183–200

Riedler M, Ruthmair M, Raidl G (2018) Strategies for iteratively refining layered graph models. Researchgate 328314707

Savelsbergh M (1985) Local search in routing problems with time windows. Ann Oper Res 4(1):285–305

Skutella M (2009) An introduction to network flows over time. In: Cook W, Lovász L, Vygen J (eds) Research trends in combinatorial optimization. Springer, Berlin, pp 451–482

Vu DM, Boland N, Hewitt M, Savelsbergh M (2018) Solving time dependent traveling salesman problems with time windows. Optimization Online 2018-6640. Transp Sci **(to appear)**

Wang X, Regan A (2002) Local truckload pickup and delivery with hard time window constraints. Transp Res B 36:97–112

Wang X, Regan A (2009) On the convergence of a new time window discretization method for the traveling salesman problem with time window constraints. Comput Ind Eng 56:161–164