Branch-Price-and-Cut Algorithms for the Team Orienteering Problem with Interval-Varying Profits

Jiaojiao Li[a], Jianghan Zhu[a], Guansheng Peng[b], Jianjiang Wang[a,*], Lu Zhen[c], Erik Demeulemeester[d]

[a] *National Key Laboratory of Information Systems Engineering, National University of Defense Technology, Changsha, China*

[b] *Defense Innovation Institute, Chinese Academy of Military Science, Beijing, China*

[c] *School of Management, Shanghai University, Shanghai, China*

[d] *Research Center for Operations Management, Department of Decision Sciences and Information Management, Faculty of Economics and Business, KU Leuven, Belgium*

Abstract: This paper studies the team orienteering problem (TOP), wherein each vertex requires two visits, and the service profit of each vertex depends on the time interval between these two visits. Motivated by scenarios like perishable product delivery, home health care scheduling, and earth observation satellite scheduling, the paper addresses two variants: the periodic orienteering problem with a focus on the number of days (NoDs) and the team orienteering problem with attention to a combination of time windows (CoTWs). It develops mixed-integer linear programming (MILP) models for both variants and devises exact Branch-Price-and-Cut (BPC) algorithms tailored to their block diagonal structures. Furthermore, this paper proposes two strategies to improve algorithm performance. A simplification strategy streamlines the directed graph network by removing redundant vertices and arcs without compromising optimality, thereby accelerating the solution of pricing problems. Additionally, a matheuristic algorithm is proposed to obtain integer solutions quickly. Computational results demonstrate the effectiveness of these algorithms, showcasing their superiority over CPLEX. The BPC algorithm, coupled with the simplification strategy, exhibits an average computational time reduction of 70% compared to the basic strategy. The effectiveness of the matheuristic algorithm is also confirmed. Finally, the findings presented herein offer valuable insights for refining and applying BPC algorithms.

Keywords: Team orienteering problem; Interval-varying profits; Mixed-integer linear programming; Dantzig-Wolfe decomposition; Branch-Price-and-Cut.

1. Introduction

The orienteering problem (OP) is normally formulated using a graph, where profits are assigned to the vertices representing the geographical locations, and each edge is associated with a travel time. The main goal of solving the OP is to maximize the total profits collected from visited vertices while adhering to maximum travel time constraints (Golden et al., 1987). On the other hand, the team orienteering problem (TOP) is a multi-route version of the OP (Chao et al., 1996), with each vertex often representing a customer. In the vast majority of (T)OPs and their variants, the profit of each customer is a constant and is completely collected after a visit

(Keshtkaran et al., 2016). Limited scholarly attention has been directed towards (T)OPs with time-varying profits, where profits depend on arrival, service, and tour times (Hammami et al., 2020; Erkut & Zhang, 1996; Erdogan & Laporte, 2013; Yu et al., 2022; Santini & Archetti, 2023).

This study assumes that each customer requires two visits. The profit collected from visiting a customer depends on the time interval between the two visits. Therefore, to maximize the total profits collected, three key decisions must be made: (1) selecting customers for visits and allocating a fleet of vehicles for these visits; (2) sequencing the assigned customers for each vehicle; and (3) determining the arrival times for the two visits to each selected customer to determine the time interval. This problem is denoted as the OP with interval-varying profits.

The problem is motivated by various practical applications such as perishable product delivery, home health care scheduling, and earth observation satellite scheduling. Due to their broad practical relevance and inherent interconnections, we focus on addressing two variants of the OP with interval-varying profits. One variant is the periodic orienteering problem (POP), while the other is a TOP. For the POP with interval-varying profits, the time interval between the two visits is defined as the number of days (NoDs), and a POP-NoDs is investigated. An illustration of a POP-NoDs instance spanning four days with three customers is presented in Fig. 1. For the sake of description, *s* and *e* denote the dummy starting and ending depots, respectively. As shown in the figure, customer A is visited on the first and third days (resulting in a two-day interval), yielding a collected profit of 30. Additionally, customer C is visited on the first and second days, accumulating a collected profit of 21. In the realm of perishable product delivery, grocery stores normally require deliveries twice a week to maintain optimal inventory levels and product freshness (Carter et al., 1996). A lengthy time interval between deliveries can result in product shortages and loss of profits. However, a short interval may lead to stock items not being sold in time, and the quality of perishable products may deteriorate as a result of prolonged storage in stores.

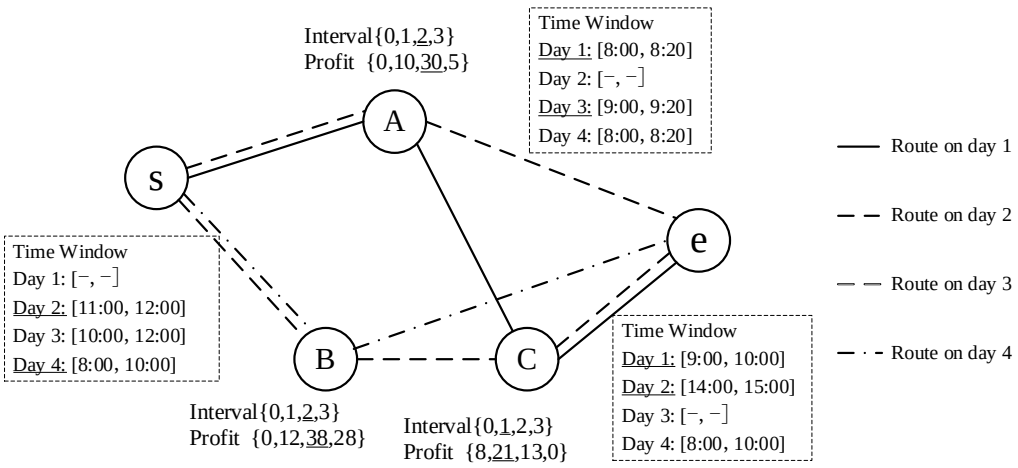

Fig. 1. An illustration of the POP-NoDs spanning four days.

Furthermore, this study explores another variant of the OP with interval-varying profits. In cases where the time windows are relatively tight, the impact of choosing different starting times within the time windows on profits can be disregarded. Therefore, the time interval can be essentially represented by determining the time windows for the two visits, approximating the time intervals with different combinations of time windows

(CoTWs). We use CoTWs to represent time intervals of the TOP and denote this problem as TOP-CoTWs. An illustration of TOP-CoTWs with four vehicles and three customers is depicted in Fig. 2. Given the limited availability of customers, each vehicle is allocated one time window to visit each customer. Additionally, if two visits are completed within the same time window, the time interval is deemed too short. Therefore, customers cannot be visited twice during the same time window. For the instance, customer A is visited by the first and third vehicles (i.e., the first and third windows), yielding a collected profit of 30. In the context of home healthcare scheduling, it is crucial to arrange two visits for each customer within a reasonable time interval. For example, elderly and disabled individuals may require assistance in getting into and out of bed for a break, operating the washing machine and drying clothes, preparing meals, and washing dishes afterward (Rasmussen et al., 2012). Consequently, each customer necessitates two visits, each of which must be allocated within their given time windows. If the time interval between the two visits is too long, the customer's satisfaction may decrease. Conversely, if the interval is too short, worker's time will be wasted due to waiting. Therefore, the overall satisfaction and well-being of the customer hinge on finding the optimal combinations of time windows for the two visits, ultimately influencing the profitability of the service provided to the customer.

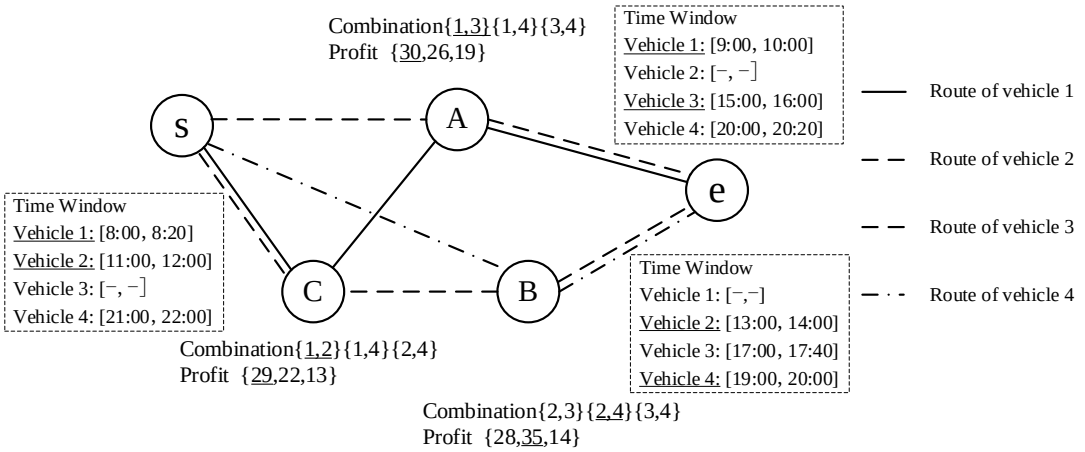

Fig. 2. An illustration of the TOP-CoTWs with four vehicles.

Another important motivation for this study arises from the scheduling of earth observation satellites (EOSs) (Wang et al., 2016; Peng et al., 2019; Wang et al., 2023), where target observations are selected and sequenced to maximize observation profits. If the targets are viewed as customers, the satellites are viewed as vehicles, the observations are viewed as services, and the setup times between observations are viewed as travel times, the scheduling of EOSs can be formulated as a variant of the TOP. Especially, in practice, some significant targets require multiple observations, such as for capturing stereoscopic images or monitoring dynamic changes. It is essential to manage and coordinate the time intervals between successive observations of the same target carefully. For example, in scenarios like floods, periodic observations of vital targets like reservoirs, rivers, and dams are essential for early warning and disaster response. If time intervals are too short, it is almost impossible for any changes to occur between consecutive observations, leading to a wastage of limited resources. Conversely, overly long-time intervals may result in crucial changes being detected too late or even lost. The above two representations (POP-NoDs and TOP-CoTWs) and corresponding algorithm can effectively address EOS scheduling challenges. On one hand, for non-agile satellites, each orbit has a single visible time window

for each target, and the time window is relatively tight, enabling the approximation of time intervals through different combinations of time windows, as in TOP-CoTWs. On the other hand, since the duration of each orbit is almost the same, the interval between two observations can be estimated by the number of orbits for separation. Given that orbits are continuous and non-overlapping, they can be characterized as days, allowing the utilization of POP-NoDs in EOS scheduling problems.

To solve the problems, we originally employ the optimization solver CPLEX. However, our preliminary experiments indicate that the solver's capability to optimally solve instances is limited by their scale. While assessing the optimality gap of heuristics and construction algorithms poses challenges; therefore, the quality of derived solutions cannot be guaranteed. In dealing with large-scale (mixed) integer linear programming ((M)ILP) problems, a Branch-Price-and-Cut (BPC) algorithm emerges as a common and effective approach. BPC algorithms excel in solving models with block diagonal structures, making them suitable for our proposed variants. Hence, we propose exact BPC algorithms tailored to address these variants. Considering their characteristics of interval-varying profits, the procedure of BPC algorithms are slightly different due to different models, both variants are thoroughly investigated in our study.

Furthermore, in scenarios involving the POP or TOP with interval-varying profits, the profit collected from visiting a customer depends on whether two visits are completed and on the time interval between the two visits. Each customer can be visited twice by different vehicles, each contributing to the collected profit. Therefore, the collected profit from a customer does not only depend on a single vehicle, but also on other vehicles visiting the same customer. Consequently, there exists route interdependency among different vehicles, which increases the complexity of the modeling and poses a great challenge to BPC algorithm design.

The main contributions of this study can be summarized as follows: (1) The POP and TOP with time interval-varying profits are proposed and analyzed in detail. For the POP and TOP, the time intervals are described as the number of days (NoDs) and the combinations of time windows (CoTWs). This leads to the formulation of POP-NoDs and TOP-CoTWs problems. (2) For the POP-NoDs and TOP-CoTWs problems, discrete profit functions are proposed to model the correlations between collected profits and time intervals between two visits to a customer. Furthermore, two compact mixed-integer linear programming (MILP) models for these problems are developed. (3) Considering the block diagonal structure of these models, extended set-partitioning models are also presented, and exact BPC algorithms tailored to each problem are proposed. To accelerate the solution of pricing problems, a strategy is proposed to simplify the directed graph network without sacrificing optimality, and a dynamic *ng*-path relaxation technique is incorporated. (4) A matheuristic algorithm is designed to rapidly obtain integer solutions. Experimental results demonstrate that the gaps between the solutions obtained by the matheuristic algorithm and the optimal solutions are quite small, which is deemed acceptable in practice. Furthermore, significant reductions in computational time are observed.

The remainder of this study is organized as follows. Section 2 presents an overview of related literatures. Section 3 discusses the mathematical formulations. Section 4 describes the proposed exact BPC algorithms. Section 5 analyzes the results of the computational experiments. Finally, Section 6 provides the conclusions of this study.

2. Literature review

The OP was firstly defined by Butt & Cavalier (1994) as a Multiple Tour Maximum Collection Problem. Two years later, Chao et al. (1996) formally introduced this problem. Comprehensive reviews of the TOP were provided by Vansteenwegen et al. (2011) and Gunawan et al. (2016). Furthermore, several challenging practical problems, such as the maximum collection problem (Butt & Cavalier, 1994) and a cash logistics problem (Orlis et al., 2020), were modeled as OPs. To tackle OPs, various heuristic approaches (Urrutia-Zambrana et al., 2021; Yahiaoui et al., 2023; Duhamel et al., 2024) and exact approaches (Bianchessi et al., 2018; Sundar et al., 2022) have been proposed recently.

In the classical TOP, each customer requires a single visit, which has been studied extensively. However, studies on problems involving multiple visits are limited. Yahiaoui et al. (2019) proposed the clustered TOP, wherein customers are grouped into clusters, each associated with a distinct profit achievable only upon visiting all customers within the cluster. They presented an exact algorithm based on a cutting plane strategy, an adaptive large neighborhood search algorithm, and an efficient split procedure. Afterwards, Dontas et al. (2023) proposed an adaptive memory matheuristic algorithm to solve the clustered TOP. Furthermore, Wu (2024) proposed an effective hybrid evolutionary algorithm integrating various operators and a solution-based tabu search procedure enhanced by a reinforcement learning mechanism. Hanafia et al. (2020) studied a variant of the TOP wherein each customer had a set of tasks to be completed by a heterogeneous fleet of vehicles in a predefined order. They proposed an improved kernel search framework using different sequencing strategies. Xu et al. (2020) firstly formulated the generalized TOP, allowing each customer to be served by different vehicles, with the profit of serving a customer being a submodular function of the number of vehicles serving them. They proposed an improved approximation algorithm for a special case where the profits of providing one or multiple services to a customer were equal. Rothenbächer (2019) addressed the periodic vehicle routing problem where customers required one or several visits during a planning horizon and proposed an exact BPC algorithm. In the existing literature with multiple visits, profits are determined based on the number of visits serving customers. However, the impact of the time interval between multiple visits to a customer on the profit remains unexplored.

The studies that are most relevant to our paper focus on the OP with time-varying profits, an area with limited literature. Referring to Yu et al. (2022), OPs with time-varying profits can be broadly categorized into four streams: (i) OPs with arrival-time-varying profits (OPATP), where profits depend on arrival times; (ii) OPs with service-time-varying profits (OPSTP), where profits depend on service times; and (iii) OPs with arrival-service-time-varying profits (OPASTP), where profits depend on both arrival and service times; (iv) OPs with tour-time-varying profits (OPTTP), where profits depend on the tour time after visiting a customer. A summary of papers on the OPs and TOPs with time-varying profits is presented in Table 1.

Erkut and Zhang (1996) studied OPATP, where collected profits decreased linearly with the arrival time, and proposed a Branch-and-Bound (B&B) algorithm to address it. Tang et al. (2005) developed a piecewise-linear profit function to express the relationships between profits and arrival times, along with an adaptive memory tabu search for solving it. Afterwards, for the OPATP, Afsar and Labadie (2013) introduced column generation and local evolutionary search algorithms for calculating lower and upper bounds, respectively. Vincent et al. (2019) investigated a variant of the OP and proposed a hybrid artificial bee colony algorithm to search optimal

solutions for small-scale instances and near-optimal solutions for large-scale instances. Kim et al. (2020) proposed a hybrid heuristic algorithm for addressing this problem.

Erdogan and Laporte (2013) studied the OPSTP, where profits increased with the service time in a convex or concave manner, and proposed a Branch-and-Cut (B&C) algorithm for solving the linear models. Pietz and Royset (2013) discussed the OPSTP with profits increasing with service times in a concave manner, proposing a B&B algorithm for its solution. Gunawan et al. (2018) explored the OPSTP, focusing on its relationship with customer service time and presenting an ILP model alongside an iterated local search algorithm. Yu et al. (2019) developed a two-phase matheuristic algorithm consisting of a tabu search algorithm and a nonlinear programming technique to solve the OPSTP with concavely increasing profits.

For the OPASTP, the collected profit is affected by both the arrival and the service times of visited customers. Yu et al. (2022) investigated the OPASTP, leading to a nonconcave profit function. They proposed a Benders B&C algorithm based on a mixed-integer nonconcave programming model and derived inequalities to tighten the upper bound of their problem.

Regarding the OPTTP, the collected profit depends on the tour time after visiting a customer. Santini and Archetti (2023) studied hazardous OPs, where parcels gathered from specific customers could result in catastrophic events, with the probability varying based on the parcel's duration on the vehicle. Addressing this issue, they put forward a mixed-integer nonlinear formulation along with techniques capable of obtaining dual bounds and producing primal solutions.

Table 1

Summary of studies on the OPs and TOPs with time-varying profits.

Literature	Problem				Profit function	Algorithm
	OPATP	OPSTP	OPASTP	OPTTP		
Erkut and Zhang (1996)	√				linear	Branch-and-Bound
Tang et al. (2005)	√				linear	Adaptive memory tabu search
Afsar and Labadie (2013)	√				linear	Column generation Local evolutionary search algorithm
Vincent et al. (2019)	√				discrete	Hybrid artificial bee colony
Kim et al. (2020)	√				discrete	Hybrid heuristic
Erdogan and Laporte (2013)		√			discrete	Branch-and-Cut
Pietz and Rovset (2013)		√			nonlinear	Branch-and-Bound
Yu et al. (2019)		√			nonlinear	Tabu search
Gunawan et al. (2018)		√			nonlinear	Iterated local search
Yu et al. (2022)			√		nonconcave	Benders Branch-and-Cut
Santini and Archetti (2023)				√	nonlinear	Multiple ways for deriving both upper bounds and lower bounds

Differently from previous studies, our research focuses on visiting each customer twice, with the collected profit of each customer dependent on whether both visits are completed and on the time interval between the two visits. To the best of our knowledge, this feature has not been explored yet. Since the two visits of a customer can be performed using different routes, the collected profit of the customer does not only depend on a single route, but also on the other route that also visits it. Hence, there exist route interdependencies, which implies that changes in one route can impact others, potentially rendering some visit sequences infeasible. This complexity poses great challenges for constructing linear models and designing BPC algorithms. The column

generation algorithm that is deemed to be most effective for routing problems cannot be directly applicable to our proposed problems. A key similarity to route interdependencies is synchronous visits to customers, where multiple visits to a customer must be in the same time window. To solve the vehicle routing problems with synchronization constraints by column generation, Dohn et al. (2009), Luo et al. (2016) and Li et al. (2020) all firstly relaxed synchronization constraints before addressing them through branching strategies or valid inequalities. Differently from their studies, we propose novel and specific formulations for POPs and TOPs with route interdependencies, allowing these interdependence constraints to be retained in the master problem. Additionally, to solve these problems efficiently, we design efficient column generation decomposition mechanisms.

3. Mathematical model

In this paper, the collected profit from visiting a customer depends on the time interval between the two visits. As mentioned above, for a POP, the time interval is defined as the number of days (NoDs), leading to a POP-NoDs model. Similarly, for a TOP, the time interval is represented as the combination of time windows (CoTWs), resulting in a TOP-CoTWs model. To evaluate the models, their complexities are compared. Moreover, a transformation method between the two formulations is introduced. Therefore, for the two problems, an appropriate representation (NoDs or CoTWs) can be chosen to represent the time interval.

3.1 POP-NoDs model

Let $G=(N,A)$ represent a directed graph, where a vertex set $N=\{0,1,\cdots,2n+1\}$ and an arc set $A=\{(i,j)|i,j\in N, i\neq j\}$ are defined. Vertices 0 and $2n+1$ correspond to the starting and terminating depots without profits, respectively. Further, assume that $\hat{N}=\{1,\cdots,2n\}$ represents a set of profitable vertices, denoting customers. To clarify, the two visits of customer i are denoted by $i\in\hat{N}_1$ and $i+n\in\hat{N}_2$. The customer sets for these two visits are given by $\hat{N}_1=\{1,\cdots,n\}$ and $\hat{N}_2=\{n+1,\cdots,2n\}$, respectively. Each customer i has a duration d_i, and the planning horizon is denoted as T. During the travel on day t, $t\in T$, a vehicle departs from the starting depot and returns to the terminating depot. Traversing any pair of customers i and j costs t_{ij} in travel time. On each day t, customer i has at most one available time window $[e_{it},l_{it}]$. Afterwards, we define set $M_i=\{0,1,\cdots,p_i\}$ ($m\in M_i$) to represent the number of days (NoDs) between two visits to customer i. p_i represents the maximum time interval for customer i. To capture the correlations between time intervals and profits, a discrete profit function $f_i(m)$ is defined as follows:

$$f_i(m)=\begin{cases} a_0 & m=0 \\ a_1 & m=1 \\ \vdots & \vdots \\ a_{p_i} & m=p_i \end{cases} \quad \forall i\in\hat{N}_1. \tag{1}$$

To maximize the collected profits, three sets of key decisions need to be made: (1) selecting customers for visits and allocating a fleet of vehicles for these visits; (2) sequencing the assigned customers for each vehicle; (3) determining the arrival times for the two visits to each selected customer to determine the time interval. Therefore, three main decision variables are defined as follows:

x_{ijt} : binary variables, where $x_{ijt}=1$ if a vehicle on day t travels directly from customer i to customer j, and is $x_{ijt}=0$ otherwise, $i, j \in N, t \in T$;

y_{im} : binary variables, where $y_{im}=1$ if the time interval between the two visits of customer i is m, and is 0 otherwise, $i \in \hat{N}_1, m \in M_i$;

s_{it} : real variables, representing the start time of the visit to customer i on day t, $i \in N, t \in T$.

Thus, the POP-NoDs model is given as follows:

$$\max \sum_{i \in \hat{N}_1} \sum_{m \in M_i} f_i(m) y_{im}, \tag{2a}$$

$$\sum_{j \in N} x_{0jt} = 1, \forall t \in T, \tag{2b}$$

$$\sum_{j \in N} x_{j(2n+1)t} = 1, \forall t \in T, \tag{2c}$$

$$\sum_{j \in N} x_{ijt} - \sum_{j \in N} x_{jit} = 0, \forall i \in \hat{N}, t \in T, \tag{2d}$$

$$\sum_{t \in T} \sum_{j \in N} x_{ijt} \le 1, \forall i \in \hat{N}, \tag{2e}$$

$$\sum_{t \in T} \sum_{j \in N} x_{ijt} - \sum_{t \in T} \sum_{j \in N} x_{(i+n)jt} = 0, \forall i \in \hat{N}_1, \tag{2f}$$

$$\sum_{t \in T} \sum_{j \in N} t x_{(i+n)jt} - \sum_{t \in T} \sum_{j \in N} t x_{ijt} - \sum_{m \in M_i} m y_{im} = 0, \forall i \in \hat{N}, \tag{2g}$$

$$\sum_{m \in M_i} y_{im} \le 1, \forall i \in \hat{N}_1, \tag{2h}$$

$$s_{it} + d_i + t_{ij} - s_{jt} - G(1 - x_{ijt}) \le 0, \forall i, j \in N, t \in T, \tag{2i}$$

$$s_{it} - \sum_{j \in N} e_{it} x_{ijt} \ge 0, \forall i \in \hat{N}, t \in T, \tag{2j}$$

$$s_{it} - \sum_{j \in N} (l_{it} - d_i) x_{ijt} \le 0, \forall i \in \hat{N}, t \in T, \tag{2k}$$

$$x_{ijt} \in \{0,1\}, \forall i \in N, t \in T, \tag{2l}$$

$$s_{it} \ge 0, \forall i \in N, t \in T, \tag{2m}$$

$$y_{im} \in \{0,1\}, \forall i \in \hat{N}_1, m \in M_i. \tag{2n}$$

The objective function (2a) maximizes the collected profits over the planning horizon. Constraints (2b) and (2c) ensure that each vehicle on each day starts and terminates its route at the depot. Constraints (2d) ensure flow conservation when visiting a customer. Constraints (2e) ensure that each customer is visited at most once. Constraints (2f) indicate that each customer is either visited twice or not visited because only a single visit is not profitable. Constraints (2g) define the relationships between variables x_{ijt} and y_{im}. Constraints (2h) indicate that each customer i has to select at most one time interval between the two visits. Constraints (2i) specify that if a vehicle travels from customer i to customer j, the starting time of the visit to customer j must be no earlier than the terminating time of the visit to customer i, plus the travel time from customer i to customer j (where G represents a sufficiently large number). Constraints (2j) and (2k) restrict the starting time of the visit to customer i on day t within the time window $[e_{it}, l_{it}]$. Finally, constraints (2l)–(2n) define the domains of all decision variables.

3.2 TOP-CoTWs model

For the TOP-CoTWs, similarly with the POP-NoDs model, the two visits to a customer can be formulated as two separate customers at the same location, each visited once. The customer sets for the two visits are denoted

by $\hat{N}_1=\{1,\cdots,n\}$ and $\hat{N}_2=\{n+1,\cdots,2n\}$, where $\hat{N}_1=\{1,\cdots,n\}$ and $\hat{N}_2=\{n+1,\cdots,2n\}$ represent the first and the second visits to the customers, respectively. The vehicle set is represented as $T=\{1,\cdots,t\}$. Each vehicle departs from the starting depot and returns to the terminating depot. Considering the limited availability of customers, each vehicle t is given one time window to visit each customer i, i.e., the available time window is defined as $[e_{it},l_{it}]$. Additionally, if the two visits are completed in the same time window, the time interval will be too short. Therefore, customers cannot be visited twice during the same time window. Consequently, the two distinct time windows corresponding to customer i are denoted as a combination $w\in S_i$ ($S_i=\{0,1,\cdots,q_i\}$). q_i represents the maximum number of combinations for customer i. The combination w for customer i is expressed in formula (3). Afterwards, we define a profit function $g_i(w)$ to represent the correlation between combinations and profits.

$$w=\begin{cases}1 & [e_{i1},l_{i1}]\&[e_{i2},l_{i2}]\\ 2 & [e_{i1},l_{i1}]\&[e_{i3},l_{i3}]\\ \vdots & \vdots\\ q_i & [e_{i(t-1)},l_{i(t-1)}]\&[e_{it},l_{it}]\end{cases}. \tag{3}$$

It is worth noting that the profit function $g_i(w)$ is analogous to the discrete function $f_i(m)$ shown in formula (1). Therefore, for the sake of clarity, we also adopt the notation m to denote the time interval and $f_i(m)$ to denote the profit function. It should be noted that although the customer splitting can enhance the problem's structural clarity, it will result in an increased number of customers. Consequently, two variants of the TOP-CoTWs model are investigated: one with customer splitting, designated as TOP1-CoTWs, and another without customer splitting, designated as TOP2-CoTWs.

3.2.1 TOP1-CoTWs model

For clarity, we define two additional parameters for each customer $i\in\hat{N}_1$ as follows: $a_{t_1}^{im}=1$ if combination m contains the time window of vehicle t_1, and 0 otherwise; $b_{t_2}^{im}=1$ if combination m contains the time window of vehicle t_2, and 0 otherwise.

Afterwards, the TOP1-CoTWs model is formulated. The objective function remains the same as in formula (2a), while satisfying the following constraints: (2b)–(2e), (2h)–(2n), and

$$y_{im}-\frac{1}{2}\left(\sum_{t\in T}\sum_{j\in N}a_t^{im}x_{ijt}+\sum_{t\in T}\sum_{j\in N}b_t^{(i+n)m}x_{(i+n)jt}\right)\le 0,\forall i\in\hat{N}_1,m\in M_i. \tag{4}$$

Constraints (4) represent the relationship between variables x_{ijt} and y_{im}, i.e., if variable $y_{im}=1$, according to the definitions of the two parameters $a_{t_1}^{im}$ and $b_{t_2}^{(i+n)m}$, customers i and $i+n$ must be visited by vehicles t_1 and t_2, respectively.

3.2.2 TOP2-CoTWs model

In this model, the two visits to each customer are not split. Therefore, the set of customers is defined as $\hat{N}=\{1,\cdots,n\}$. Furthermore, for each customer $i\in\hat{N}$, we define an additional parameter denoted as $a_{t_1t_2}^{im}$, which are defined as follows: $a_{t_1t_2}^{im}=1$ if combination m includes the two time windows of vehicles t_1 and t_2, and 0 otherwise.

Afterwards, the TOP2-CoTWs model is formulated as follows:

$$\max \sum_{i\in\hat{N}} \sum_{m\in M_i} f_i(m) y_{im}. \tag{5a}$$

s.t. Constraints (2b)–(2e), (2i)–(2m), and

$$\sum_{t\in T} \sum_{j\in N} x_{ijt} \le 2, \forall i \in \hat{N}, \tag{5b}$$

$$\sum_{m\in M_i} \sum_{t_2\in T} a^{im}_{t_1 t_2} y_{im} - \sum_{j\in N} x_{ijt_1} \le 0, \forall i \in \hat{N}, t_1 \in T, \tag{5c}$$

$$\sum_{m\in M_i} \sum_{t_1\in T} a^{im}_{t_1 t_2} y_{im} - \sum_{j\in N} x_{ijt_2} \le 0, \forall i \in \hat{N}, t_2 \in T, \tag{5d}$$

$$y_{im} \in \{0,1\}, \forall i \in \hat{N}, m \in M_i . \tag{5e}$$

Constraints (5b) indicate that each customer can be visited at most twice. Constraints (5c) and (5d) establish the relationships between decision variables x_{ijt} and y_{im}, i.e., if variable $y_{im}=1$, according to the definition of the parameter $a^{im}_{t_1 t_2}$, customer i must be visited by both vehicles t_1 and t_2.

3.3 Remarks on different models

In this section, a vehicle in the TOP-CoTWs can be regarded as a day in the POP-NoDs. Therefore, for the POP, the time interval between two visits can also be represented using a combination of time windows, enabling the formulation of a corresponding model. When a customer is visited twice on two separate days, we refer to this as a combination. The conversion method between these two representations is illustrated in Fig. 3. For instance, if a customer is visited twice with a one-day interval (NoDs), this can be transformed into a combined form (CoTWs) such as {1,2}, representing the first and second days. It is assumed that the profits of combinations with an identical time interval between their time windows are the same. Similarly, for a TOP, the combinations of time windows corresponding to a customer visited twice can be converted into numbers of days. Therefore, for these two problems, an appropriate method (NoDs or CoTWs) can be chosen to represent time intervals.

As presented in Fig. 3, three combinations {1,2}, {2,3} and {3,4} yield identical profits due to their one-time window separation. To better capture the relationship between profits and time intervals, it is proposed that the profits of these combinations vary, as presented in Fig. 4. The time interval of approximately five hours between the second and third windows optimally yields the maximum profit (20) for the customer. Conversely, the time interval of approximately one hour between the first and second windows is too short; while the time interval of approximately seven hours between the third and fourth windows is too long, resulting in decreased profits for the customer. Therefore, assuming uniform profits across unevenly dispersed time windows is unreasonable. Instead, a more precise representation can be achieved by defining the time interval as a combination of time windows (TOP-CoTWs). In this way, the correlations between profits and time intervals can be depicted more accurately.

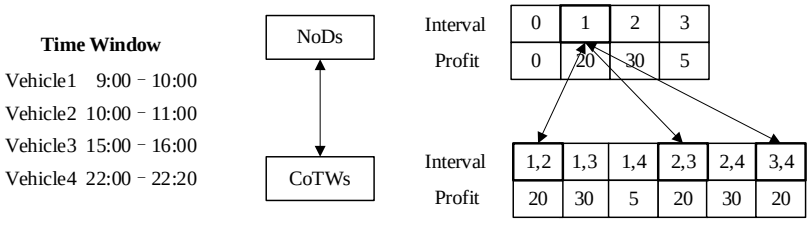

Fig. 3. Conversion method of the two representations.

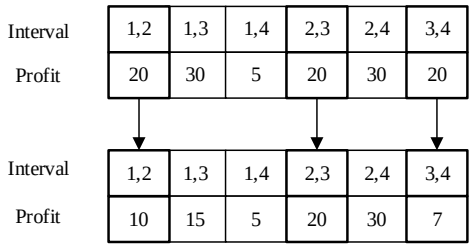

Fig. 4. Two profit cases of the CoTWs.

Afterwards, define n as the total number of customers. For each customer i, let α_i and β_i denote the number of time intervals in the POP-NoDs and TOP-CoTWs models, respectively. Let t represent both the planning horizon and the number of vehicles. To compare the three models introduced in this section, the numbers of variables and constraints are reported in Table 2. For instance, considering a scenario with ten customers, a planning horizon spanning five days. In the POP-NoDs model, the time interval α_i of each customer $i \in \hat{N}_1$ can take a potential value from $\{0,1,2,3,4\}$. Additionally, with five vehicles available, the TOP-CoTWs model involves $C_5^2 = 10$ possible combinations for the time interval β_i of each customer $i \in \hat{N}_1$. Under these conditions, the TOP2-CoTWs model comprises 820 variables, whereas the POP-NoDs and TOP1-CoTWs models involve 2,470 and 2,520 variables, respectively. Concerning constraints, the TOP2-CoTWs, POP-NoDs, and TOP1-CoTWs models involve 2,785, 2,865, and 985 constraints, respectively. It is observed that the TOP2-CoTWs model that does not split customers significantly reduces the number of variables and constraints. Therefore, it entails relatively fewer binary variables and constraints compared to the other two models.

Table 2

Comparison of different models.

Problem	Model	No. of variables	No. of constraints
POP	POP-NoDs	$(2n+2)^2 t + \sum_{i \in \hat{N}_1} \alpha_i$	$(2n+2)^2 t + 6nt + 6n + t$
TOP	TOP1-CoTWs	$(2n+2)^2 t + \sum_{i \in \hat{N}} \beta_i$	$(2n+2)^2 t + 6nt + 4n + t + \sum_{i \in \hat{N}} \beta_i$
TOP	TOP2-CoTWs	$(n+2)^2 t + \sum_{i \in \hat{N}} \beta_i$	$(n+2)^2 t + 5nt + n + t$

4. Branch-Price-and-Cut algorithm

The Branch-Price-and-Cut algorithm is a prevalent approach in combinatorial optimization for addressing large-scale MILP problems. In particular, it performs well in solving MILP models with block diagonal structures. Notably, it has been successfully employed to achieve optimality in solving the TOP (Keshtkaran et al., 2016) and the VRP (Wang & Sheu, 2019). Given its effectiveness, we develop exact BPC algorithms tailored to solve the proposed problems. Firstly, the standard Dantzig-Wolfe decomposition technology is utilized to reformulate the original problem into a master problem (MP) and multiple pricing subproblems. Because the number of feasible routes is exponentially large, it is prohibitively time-consuming and almost impossible to enumerate all routes. Therefore, it is imperative to dynamically add promising routes into the MP (Lübbecke & Desrosiers, 2005). To this end, we adopt the column generation (CG) approach, which manages the solution of the linear programming (LP) relaxation of the MP. Subsequently, to enhance the tightness the LP bound, subset-

row (SR) inequalities can be incorporate to the formulations. Finally, we describe the branching strategies employed to derive integer solutions. It is worth noting that Sections 4.1–4.3 will describe the BPC algorithm for the POP-NoDs model in detail, and the BPC algorithms for the other two models (TOP1-CoTWs and TOP2-CoTWs) are slightly different, as presented in Section 4.4.

4.1 Column generation

This study performs column generation for routing variables. Firstly, a restricted master problem (RMP) with a limited column set is solved iteratively using the CPLEX solver. Afterwards, a bidirectional label-setting algorithm is employed to identify solutions of pricing subproblems, and new columns are progressively added until the optimal solution is achieved. To accelerate the solution of the pricing subproblems, a simplification strategy is designed to adapt the directed graph network, and a dynamic *ng*-path relaxation technique is implemented.

4.1.1 Dantzig-Wolfe decomposition

The original POP-NoDs model is decomposed into a master problem and multiple pricing subproblems. The routes on each day are regarded as columns. Denote the set of columns on day $t \in T$ as R_t; afterwards, each column $r \in R_t$ contains parameters x_{ijt}^r, where $x_{ijt}^r = 1$ if column r on day t visits customers i and j consecutively and 0 otherwise, $i, j \in N, r \in R_t, t \in T$. The following decision variables are defined: 1) $z_{tr} = 1$ if route r is selected on day t and 0 otherwise, $r \in R_t, t \in T$; 2) $y_{im} = 1$ if the time interval between two visits to customer i is m, and 0 otherwise, $i \in \hat{N}_1, m \in M$. Subsequently, for the original POP-NoDs problem, its MP can be formulated as follows:

$$\max \sum_{i \in \hat{N}_1} \sum_{m \in M_i} f_i(m) y_{im}, \tag{6a}$$

$$\sum_{t \in T} \sum_{r \in R_t} \sum_{j \in N} x_{ijt}^r z_{tr} \le 1, \forall i \in \hat{N}, \tag{6b}$$

$$\sum_{t \in T} \sum_{r \in R_t} \sum_{j \in N} x_{ijt}^r z_{tr} - \sum_{t \in T} \sum_{r \in R_t} \sum_{j \in N} x_{(i+n)jt}^r z_{tr} = 0, \forall i \in \hat{N}_1, \tag{6c}$$

$$\sum_{m \in M_i} m y_{im} - \sum_{t \in T} \sum_{r \in R_t} \sum_{j \in N} t x_{(i+n)jt}^r z_{tr} + \sum_{t \in T} \sum_{r \in R_t} \sum_{j \in N} t x_{ijt}^r z_{tr} = 0, \forall i \in \hat{N}_1, \tag{6d}$$

$$\sum_{r \in R_t} z_{tr} \le 1, \forall t \in T, \tag{6e}$$

$$\sum_{m \in M_i} y_{im} \le 1, \forall i \in \hat{N}_1, \tag{6f}$$

$$z_{tr} \in \{0,1\}, \forall r \in R_t, t \in T, \tag{6g}$$

$$y_{im} \in \{0,1\}, \forall i \in \hat{N}_1, m \in M_i. \tag{6h}$$

Constraints (6b) indicate that each customer should be visited at most once. Constraints (6e) indicate that at most one column can be chosen for each day. Constraints (6g)–(6h) show the domains of all decision variables. The other constraints, i.e., (6c), (6d), (6f), are similar to (2f)–(2h) of the original POP-NoDs model.

Subsequently, the integrality constraints are relaxed to obtain the LP problem. Let dual variables μ_i, ν_i, δ_i, and γ_t be associated with constraints (6b)–(6e), respectively. Considering the special structure of the dual problem of the MP, the pricing subproblem on day $t \in T$ is defined as follows:

$$\min \ -\sum_{i\in\hat{N}_1}\sum_{j\in N}(\mu_i+\nu_i+t\delta_i)x_{ij}+\sum_{i\in\hat{N}_1}\sum_{j\in N}(\nu_i-\mu_{i+n}+t\delta_i)x_{(i+n)j}-\gamma_t. \tag{7}$$

s.t. Constraints (2b)–(2d) and (2i)–(2l).

The pricing subproblem is a Resource-Constrained Elementary Longest Path Problem (RCELPP). At each CG iteration, columns with positive reduced costs are identified. The iteration process continues until all pricing subproblems obtain nonpositive reduced costs, demonstrating that the LP relaxation problem has been solved to optimality.

4.1.2 Reduced directed graphs

Given the limited availability of customers, each day is allocated at most one time window to visit each customer for the POP-NoDs problem. The pricing network for day $t\in T$ is defined as a graph $G^t=(V^t,A^t)$, where V^t represents the set of vertices (customers) that can be visited on day t and A^t represents the set of arcs that can satisfy the travel time constraints between two vertices. Additionally, in certain scenarios, such as the delivery of perishable products, each customer cannot be visited twice on the same day due to the short interval between the two visits. Based on these characteristics, redundant vertices and arcs are eliminated. The vertex corresponding to the second visit can be removed from the directed graph at its earliest time window. Similarly, at its latest time window, the vertex corresponding to the first visit can be removed. For each day's directed graph, the procedure of the simplification strategy is shown in Algorithm 1. This simplification strategy is proposed to simplify the directed graph network, thus accelerating the solution of pricing subproblems.

Algorithm 1 Simplification strategy.

initialize: $t_1=1$, $t_2=|T|$
for $i\in\hat{N}_1$ **do**
  for $t_1\in T$ **do**
    if $TW_{it_1}\neq\varnothing$ **then** // Find the first available time window
      Delete vertex $i+n$ from V_{t_1}
      for $j\in N$ **do**
        Delete the arc between vertices $i+n$ and j from A_{t_1}
        break
      end for
    end if
  end for
  for $t_2\in T$ **do**
    if $TW_{it_2}\neq\varnothing$ **then** // Find the last available time window
      Delete vertex i from V_{t_2}
      for $j\in N$ **do**
        Delete the arc between vertices i and j from A_{t_2}
        break
      end for
    end if
  end for
end for

The example presented in Fig. 1 is utilized to illustrate the directed graphs before and after the reduction process. For the sake of clarity, only two vertices A and B are included. Vertex A is divided into A^1 and A^2,

and a similar decomposition strategy is employed for vertex B. Fig. 5(a) represents the directed graph spanning four days, i.e., prior to the reduction, the directed graph remains the same for all four days. Fig. 5(b) illustrates the directed graph for each individual day after reduction, highlighting the differences from day to day. Notably, each vertex is not allowed to be visited twice on the same day due to the short interval between the two visits. Additionally, a few customers do not have a designated time window on a specific day due to limited availability, as shown in Fig. 1. Consequently, vertices (B^1, B^2) in the directed graph of the first day can be eliminated, as along with vertices (A^1, A^2) in the directed graph of the second day. Furthermore, based on the characteristics that each vertex cannot be visited twice on the same day, the directed graph can be simplified by removing the vertex corresponding to the second visit (A^2) in the directed graph of the first day, the vertex corresponding to the second visit (B^2) of the second day, and the vertices of the first visit (A^1, B^1) of the fourth day.

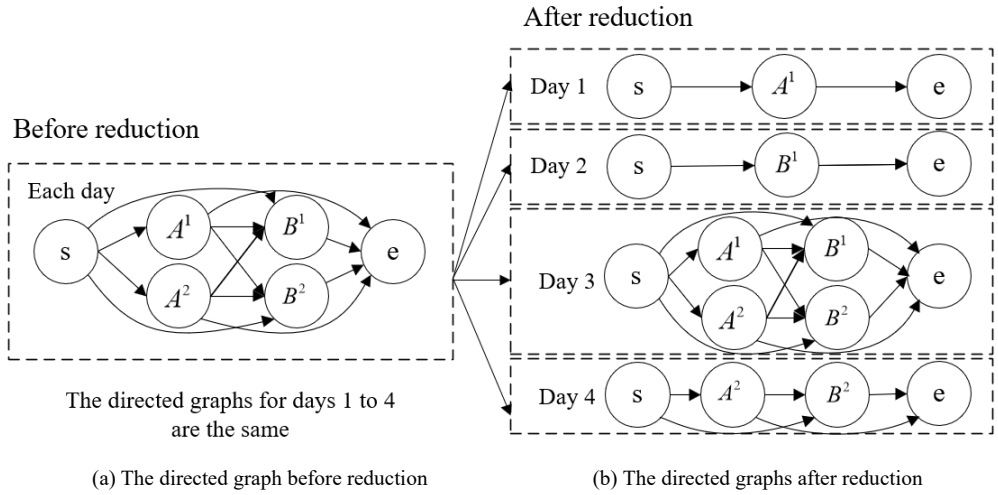

Fig. 5. An instance of directed graphs spanning four days.

4.1.3 Bidirectional label-setting algorithm for pricing subproblems

The proposed label-setting algorithm is inspired by Righini and Salani (2006). Furthermore, this study incorporates time as a critical resource for the bidirectional label-setting algorithm. The forward extensions, backward extensions and merging procedures are elaborated in the following.

A forward feasible partial route $p^f = (i_0 = 0, i_1, \cdots, i_{s-1}, i_s)$ is denoted by L^f and is associated with the last vertex visited i_s, where $L^f = (i^f, s^f, c^f, R^f)$, and the definitions of the four elements are described as follows:

i^f : the last vertex visited with $i^f = i_s$;

s^f : the earliest starting time of the visit to i^f;

c^f : the reduced cost of the partial route;

$R^f \subseteq \hat{N}$: the set of visited vertices.

When extending forward label L^f at vertex i to its adjacent vertex j, $j \notin R^f$, if $s^f \le P/2$ (half of the planning horizon P) and $s_i + d_i + t_{ij} \le we_{jt} - d_j$, a novel forward label L^f_{new} will be created by the following resource extension functions:

$$s^f_{new} = \max\left\{s^f + d_i + t_{ij}, ws_{jt}\right\}, \tag{8a}$$

$$c_{new}^{f} = c^{f} + \begin{cases} -\left(\mu_j + v_j + t\delta_j\right), & if \ \ 1 \le j \le n \\ v_{j-n} - \mu_j + t\delta_{j-n}, & if \ \ n+1 \le j \le 2n \end{cases}, \tag{8b}$$

$$R_{new}^{f} = R^{f} \cup i. \tag{8c}$$

The forward labels are generated commencing from the initial label $L_0^f = (0,0,0,\varnothing)$. Furthermore, in the process of forward extension, a label L_2^f terminating at vertex i is dominated by another label L_1^f terminating at the same vertex only if $s_2^f \le s_1^f$, $c_2^f \ge c_1^f$, $R_2^f \subseteq R_1^f$, and at least one of the three inequalities must be strictly satisfied. Consequently, dominated labels can be safely removed without sacrificing optimality.

The forward label-setting algorithm is presented in Algorithm 2. The backward extension is similar to the forward extension, but in the opposite direction. For simplicity, the details of the backward extension are omitted.

A forward label $L^f = (i^f, s^f, c^f, R^f)$ terminating at vertex j and a backward label $L^b = (i^b, s^b, c^b, R^b)$ terminating at the same vertex can be merged into a complete route if $s^f \le s^b$ and $R^f \cap R^b = \varnothing$. The reduced cost c_r of the complete route r can be calculated as follows:

$$c_r = c^f + c^b + \begin{cases} -\left(\mu_j + v_j + t\delta_j\right), & if \ \ 1 \le j \le n \\ v_{j-n} - \mu_j + t\delta_{j-n}, & if \ \ n+1 \le j \le 2n \end{cases}. \tag{9}$$

Algorithm 2 Forward label-setting algorithm

Define Γ_i^f : the set of forward labels terminating at vertex i ; *D:* the set of vertices to be extended.

Initialize partial route: $\Gamma_0^f = \{(0,0,0,\varnothing)\}$

$D \leftarrow N$; // the set of all vertices

repeat

  Select $i \in D$;

  for $L_i^f \leftarrow (i, s_i, c_i, R_i) \in \Gamma_i^f$ **do**

    for $j \in N$, $j \notin R_i$ **do**

      if $s^f \le P/2$, $s_i + d_i + t_{ij} \le we_{jt} - d_j$ **then**

        $L_j^f \leftarrow L_i^f$; // Label L_i^f is used to generate new label L_j^f

        if label L_j^f satisfies the dominance rule **then**

          $\Gamma_j^f \leftarrow L_j^f$; // Label L_j^f is saved to set Γ_j^f

          $D \leftarrow D \cup \{j\}$;

        end if

      end if

    end for

  end for

  $D \leftarrow D \setminus \{i\}$;

until $D = \varnothing$

4.1.4 Dynamic *ng*-path relaxation

In this study, to reduce the number of neighborhood vertices and to complete the CG procedure on a smaller-scale as much as possible, a dynamic *ng*-path relaxation technique proposed by Roberti and Mingozzi (2014) is adopted. Firstly, define V_i to represent the neighborhoods of vertex $i \in N_i$. Let $\hat{\upsilon}_{tij}$ denote the "profit" of

generating a subtour between vertices i and j on day t, to represent the neighborhood relationship between the two vertices. In this study, the magnitude of the intersection between the time windows of the two visits directly correlates with the value of $\hat{\upsilon}_{tij}$. Therefore, the "profit" $\hat{\upsilon}_{tij}$ is calculated as follows:

$$\hat{\upsilon}_{tij} = \begin{cases} 0, & e_{jt} \geq l_{it} \text{ or } l_{jt} \leq e_{it} \\ \min\{l_{it}, l_{jt}\} - \max\{e_{it}, e_{jt}\}, & \text{others} \end{cases}. \tag{10}$$

All the neighborhoods of vertex i are sorted by the value of $\hat{\upsilon}_{tij}$. Let λ represent the size of the neighborhood set V_i. Namely, for vertex i, the set V_i contains i and its $(\lambda-1)$ closest neighborhood vertices. After each iteration of column generation, based on the obtained fractional solution, a subtour $p^s = (j, i_1, \cdots, i_s, j) \in \Omega$ (Ω represents the set of all subtours) will be generated. Afterwards, for each vertex $i \in \{i_1, \cdots, i_s\}$, vertex j repeatedly visited in the subtour p^s is added to the set V_i. Subsequently, the pricing subproblem and the RMP are addressed repeatedly until there are no subtours in the derived solution.

4.2 Subset-row inequality

This study adds SR inequalities introduced by Jepsen et al. (2008) to cut off partial fractional solutions and thus to enhance the LP relaxation. According to the model of the MP, the SR inequality on subset S with $0 < k \leq |S|$ can be defined as follows:

$$\sum_{t\in T}\sum_{r\in R_t}\left\lfloor \sum_{i\in S}\sum_{j\in N}\frac{x_{ijt}^r}{k}\right\rfloor z_{tr} - \frac{|S|}{k} \leq 0, \tag{11}$$

where parameters x_{ijt}^r indicate whether column r on day t visits customers i and j consecutively. Let q be the dual variable associated with inequality (11). The reduced cost of day $t \in T$ is modified according to the following formula:

$$-\sum_{i\in\bar{N}_1}\sum_{j\in N}(\mu_i + \nu_i + t\delta_i)x_{ij} + \sum_{i\in\bar{N}_1}\sum_{j\in N}(\nu_i - \mu_{i+n} + t\delta_i)x_{(i+n)j} - \gamma_t - q\left\lfloor \sum_{i\in S}\sum_{j\in N}\frac{x_{ij}}{k}\right\rfloor. \tag{12}$$

If the current fractional solution violates inequality (11), the inequality is added to the RMP. Afterwards, the RMP is solved again. Based on the modified reduced cost, we solve pricing subproblems. In this way, the columns will not appear again, and the linear relaxation upper bound is tighter.

Drawing from the related literature (Rothenbächer, 2019), only SR inequalities involving subsets with cardinalities $S = 3$ and $k = 2$ are taken into account, as they can be efficiently separated through enumeration. Additionally, the number of valid inequalities is limited to prevent the pricing subproblem from becoming too difficult to solve. During the process of separating and adding valid inequalities, we ensure that the subset S_ζ associated with each added valid inequality ζ overlap as minimally as possible, allowing us to obtain better results with a reduced number of valid inequalities.

4.3 Branching strategy

Given that the CG relaxes integer constraints, it is necessary to branch all relaxed decision variables in the MP to integers upon solving. This study proposes a hierarchical three-level (HTL) branching strategy, including day-based, arc-based, and interval-based branching. Notably, the third level of branching is specially developed to address the unique characteristic of interval-varying profits. Furthermore, we propose a matheuristic branching strategy (MHB) as an alternative to the third level of branching, obtaining integer feasible solutions

faster.

4.3.1 Hierarchical three-level branching strategy

This study develops an HTL branching strategy, where the first two levels of branching are borrowed from Santini et al. (2018), and the last level of branching is specially designed.

With respect to the first level of branching, the specific day for the visit to each customer is determined. Let $h_i = \sum_{t\in T}\sum_{r\in R}\sum_{j\in N} x_{ijt}^r z_{tr}$ be the day for the visit to customer i. If there exist customers with fractional values of h_i, the customer i with h_i closest to 0.5 is chosen, and two branches are created as a result.

Proceeding to the second level of branching, when the day for the visit to each customer is an integer, the branching is based on the total flow between pairs of customers. Assume $g_{ij} = \sum_{t\in T}\sum_{r\in R} x_{ijt}^r z_{tr}$ represents the total flow directed from customer i to customer j. If there exists at least one pair of customers with a fractional value for g_{ij}, the pair (i, j) whose g_{ij} with a fractional part nearest to 0.5 is selected, leading to the formation of two branches.

With respect to the third level of branching, a specific branching strategy focusing on time intervals is proposed. If there exists at least one customer i with a fractional value for y_{im}, the time interval m of customer i whose y_{im} with a fractional part closest to 0.5 is selected, and two branches are created. In one branch, customers i and $(i+n)$ must be visited at time interval m, i.e., the constraint $y_{im}=1$ is appended to the MP. In the other branch, the time interval between the two visits to customers i and $(i+n)$ cannot be at interval m. Consequently, the constraint $y_{im}=0$ is incorporated into the MP. Meanwhile, the two customers retain the flexibility to be visited at any other time interval, ensuring that only one time interval is selected for each customer.

4.3.2 Matheuristic branching strategy

Increasing the number of branching levels significantly enlarges the size of the branch-and-bound tree and raises the number of column generation iterations. Therefore, the proposed branching strategy can be further improved by reducing the number of branching levels. Considering that the third level of branching has a considerable influence on computational time, a matheuristic branching strategy is developed as an alternative. Specifically, once the first two levels of branching are completed, a column is generated for each day. Subsequently, for customer i, the time interval m between the two visits is calculated using the currently available columns, and the corresponding constraint $y_{im}=1$ is incorporated into the RMP. Finally, the variable y_{im} also attains an integer value, thereby obtaining the optimal integer solution.

4.4 BPC algorithms for the TOP-CoTWs models

When the BPC algorithm is adopted to solve the TOP1-CoTWs and TOP2-CoTWs models, the algorithm procedures differ from those of the POP-NoDs model.

4.4.1 The BPC algorithm for the TOP1-CoTWs model

In solving the TOP1-CoTWs model, the MP and pricing subproblems are different from those of the POP-NoDs model. The bidirectional label-setting algorithm, dynamic *ng*-path relaxation and branching strategy remain the same.

The MP can be formulated with the same objective function as in formula (2a), along with the following constraints:

s.t. Constraints (6b), (6e), (6g)–(6h), and

$$y_{im}-\frac{1}{2}\left(\sum_{t\in T}\sum_{r\in R_t}\sum_{j\in N}a_t^{im}x_{ijt}^{r}z_{tr}+\sum_{t\in T}\sum_{r\in R_t}\sum_{j\in N}b_t^{(i+n)m}x_{(i+n)jt}^{r}z_{tr}\right)\le 0,\forall i\in\hat{N}_1,m\in M_i. \tag{13}$$

Constraints (13) are similar to constraints (4) of the original TOP1-CoTWs model.

Let dual variables μ_i, ν_i, ρ_{im}, and γ_t be associated with constraints (6b), (6c), (13), and (6e), respectively. For each route, the objective function of the pricing subproblem is defined as follows:

$$\min\ -\sum_{i\in\hat{N}_1}\sum_{j\in N}\left(\mu_i+\nu_i-\frac{1}{2}\sum_{m\in M_i}\rho_{im}\right)x_{ij}+\sum_{i\in\hat{N}_1}\sum_{j\in N}\left(\nu_i-\mu_{i+n}+\frac{1}{2}\sum_{m\in M_i}\rho_{im}\right)x_{(i+n)j}-\gamma. \tag{14}$$

4.4.2 The BPC algorithm for the TOP2-CoTWs model

In solving the TOP2-CoTWs model, the MP and pricing subproblems differ from those of the other two models. In addition, the basic directed graph is adopted to solve pricing subproblems, and only the last two levels of the HTL branching strategy are utilized. The other elements of the algorithm remain the same as previously presented.

The MP can be formulated with the same objective function as formula (5a), along with the following constraints:

s.t. Constraints (6e)–(6h)

$$\sum_{t\in T}\sum_{r\in R_t}\sum_{j\in N}x_{ijt}^{r}z_{tr}\le 2,\forall i\in\hat{N}, \tag{15a}$$

$$\sum_{m\in M_i}\sum_{t_2\in\{t|t>t_1,t\in T\}}a_{t_1t_2}^{im}y_{im}-\sum_{r\in R_{t_1}}\sum_{j\in N}x_{ijt_1}^{r}z_{t_1r}=0,\forall i\in\hat{N},t_1\in T, \tag{15b}$$

$$\sum_{m\in M_i}\sum_{t_1\in\{t|t<t_2,t\in T\}}a_{t_1t_2}^{im}y_{im}-\sum_{r\in R_{t_2}}\sum_{j\in N}x_{ijt_2}^{r}z_{t_2r}=0,\forall i\in\hat{N},t_2\in T. \tag{15c}$$

Constraints (15a)–(15c) are equivalent to constraints (5b)–(5d) of the original TOP2-CoTWs model.

Finally, let four dual variables μ_i, w_{it}, λ_{it}, and γ_t be associated with constraints (15a)–(15c) and (6e), respectively. For each route, the objective function of the pricing subproblem is defined as follows:

$$\min\ -\sum_{i\in\hat{N}}\sum_{j\in N}(\mu_i-w_{it}-\lambda_{it})x_{ij}-\gamma. \tag{16}$$

5. Computational experiments

In this study, numerical experiments were conducted to test the performance of the proposed algorithms. The original problem proposed in Section 3 was solved using CPLEX 12.8 solver with its default settings. The master problem of column generation was also solved using CPLEX 12.8. The algorithms were coded in C++ and interfaced with CPLEX through its C++ APIs. The experiments were conducted on a Lenovo PC, equipped with a 2.8 GHz Intel Core i7 CPU and 8 GB of RAM, operating in a single-thread environment. Each instance was allowed a maximum computation time of one hour. "TL" represented that the computation time of the instance has reached the upper limit. Additionally, "#" indicated that the optimal solution of the instance was not obtained within the time limit. All computational times presented in this section are measured in seconds.

5.1 Test instances

Given the absence of benchmark instances for the POP-NoDs and TOP-CoTWs, a subset of instances from Souffriau et al. (2013) was used to create two distinct test datasets, designated as PS and TS, specifically tailored for our study. These datasets differed significantly in their parameter configurations. In dataset PS, the number of customers varied from 10 to 40, increasing in increments of 10. The planning horizon was fixed at three and five days, and each customer was assigned a maximum of one time window per day. Meanwhile, dataset TS featured a varying number of customers, encompassing 10, 20, 30, and 50. The fleet size comprised three or five vehicles, and the planning horizon was set at 24 hours. Each customer in this dataset had a maximum time window allocated for each vehicle. The number of time window combinations corresponded to the number of vehicles, with three combinations for three vehicles and ten for five vehicles.

Despite these differences, both datasets shared certain common parameters. The durations were randomly generated within the range of 100 to 200 seconds. Profits were assigned within the range of 0 to 50. Additionally, the overall number of SR inequalities was capped at 100 to ensure computational efficiency. For each unique parameter configuration, we randomly generated three problem instances and computed their average performance metrics for a more robust and representative analysis.

5.2 Evaluation of the BPC Algorithms

The performance of the BPC algorithms was thoroughly evaluated for both POP-NoDs and TOP-CoTWs. In addition, CPLEX was directly used to solve the three MILP models presented in Section 3. Table 3 provides comprehensive details of the experimental results. Specifically, it includes information on the instance sets, the number of customers ($|N|$), and the planning horizon ($|T|$). For the BPC algorithms, the following metrics were determined: the relaxed upper bound at the root node (z_{rub}), the best feasible solution found (z^*), the total number of nodes ($Node$) explored in the B&B tree, and the computational time required (CT). Furthermore, for CPLEX, the following additional metrics were determined: the lower bound on the optimal solution (z_{lb}), the upper bound (z_{ub}), and the gap between the lower and upper bound expressed as $Gap = (z_{ub} - z_{lb}) / z_{lb} \times 100\%$. All these results are presented in Appendix A.1 of the Supplementary Material.

For POP-NoDs, the results obtained by both the BPC algorithm and CPLEX for different numbers of customers and planning horizons are presented in Table 3. Notably, the BPC algorithm demonstrated superior performance compared to CPLEX. For instances involving 10 customers, both the BPC algorithm and CPLEX could obtain optimal solutions. However, the BPC algorithm was more efficient than CPLEX, solving instances P1-1 and P1-2 in one and six seconds respectively, whereas CPLEX required 11 and 125 seconds, respectively. For 20-customer instances, the BPC algorithm was able to achieve optimal solutions, whereas CPLEX could solve only instance P2-1 within an hour. Additionally, when attempting to solve instance P2-2, CPLEX reached the time limit. For instances with 30 and 40 customers, CPLEX failed to obtain an optimal solution within an hour. As the instance size increased, the gaps between the upper and lower bounds increased significantly. In contrast, the BPC algorithm could optimally solve all instances within an hour. When addressing instance P4-2, no optimal solution was obtained due to the time limit, but the feasible solution provided by the BPC algorithm was significantly superior to that of CPLEX.

As shown in Table 3, the profits increased with both the increase of the number of customers ($|N|$) and the extension of the planning horizon ($|T|$). It was discovered that incorporating two additional days of operation yielded greater profits than acquiring 20 additional customers, while keeping all other resources constant. Consequently, from a strictly profit-oriented perspective, decision makers could significantly increase their profits by extending the planning horizon, with specific measures such as allocating more vehicles and extending working hours. Additionally, as shown in results, the number of B&B nodes significantly increased with the increase of the planning horizon $|T|$, and the gap between the optimal solution and the upper bound of the relaxation at the root node was widened. Therefore, the extension of the planning horizon $|T|$ resulted in a significant increase in computational time.

Table 3

Results of the proposed BPC algorithm and CPLEX for the POP-NoDs model.

Instance	$\|N\|$	$\|T\|$	BPC				CPLEX				
			z_{rub}	z^*	*Node*	*CT*	z_{rub}	z_{lb}	z_{ub}	*Gap*	*CT*
P1-1	10	3	52.5	50.7	7	1	205.6	50.7	50.7	0	11
P1-2	10	5	213.4	189.5	75	6	398.7	189.5	189.5	0	125
P2-1	20	3	58.3	54.8	12	9	418.9	54.8	54.8	0	480
P2-2	20	5	179.8	165.1	80	81	759.8	130.7	491.3	296	TL
P3-1	30	3	80.9	75.0	20	44	604.0	71.9	163.3	132	TL
P3-2	30	5	246.1	213.2	313	376	1,194.6	116.0	1,009.6	936	TL
P4-1	40	3	60.7	55.2	35	88	794.5	38.6	409.2	1,165	TL
P4-2	40	5	259.2	230.6	#	#	1,498.8	0.0	1,498.8		TL

Table 4

Results of the proposed BPC algorithm and CPLEX for the TOP-CoTWs model.

Model	Instance	$\|N\|$	$\|T\|$	BPC				CPLEX				
				z_{rub}	z^*	*Node*	*CT*	z_{rub}	z_{lb}	z_{ub}	*Gap*	*CT*
TOP1-CoTWs	T1-1	10	3	68.6	50.7	60	11	170.4	50.7	50.7	0	17
	T1-2	10	5	258.2	176.7	122	97	398.7	176.7	176.7	0	106
	T2-1	20	3	79.6	54.8	296	483	418.9	54.8	54.8	0	709
	T2-2	20	5	215.7	183.8	#	#	758.2	178.5	249.8	52	TL
TOP2-CoTWs	T1-1	10	3	51.6	50.7	5	3	207.2	50.7	50.7	0	15
	T1-2	10	5	177.5	176.7	3	5	400.0	176.7	176.7	0	151
	T2-1	20	3	55.6	54.8	7	10	423.8	54.2	261.6	0	TL
	T2-2	20	5	178.9	178.1	3	14	763.7	167.2	551.8	254	TL
	T3-1	30	3	72.4	69.2	10	31	590.1	65.7	307.6	451	TL
	T3-2	30	5	240.8	240.8	1	26	1,193.1	202.5	921.9	370	TL
	T4-1	50	3	75.0	74.5	3	70	981.7	60.6	770.4	1,209	TL
	T4-2	50	5	272.1	271.8	4	123	1,890.8	192.5	1,730.4	825	TL

Afterwards, for TOP1-CoTWs and TOP2-CoTWs problems, the solution results of the proposed BPC algorithm were compared with those of CPLEX separately, as shown in Table 4. The columns in the table were consistent with those of Table 3. For the TOP1-CoTWs, the BPC algorithm could solve instances T1-1 and T1-2 in 11 and 97 seconds, respectively, while CPLEX spent 17 and 106 seconds in solving these two instances, respectively. When solving instance T2-2, the optimal solution was not obtained by both the BPC algorithm and

CPLEX due to the time limit. For larger-scale instances, the BPC algorithm terminated prematurely, obtaining unsatisfactory results. This implies that the BPC algorithm is suitable only for instances with a limited number of customers. On the other hand, when dealing with TOP2-CoTWs, CPLEX reached the time limit when solving instances involving 20 customers, while the BPC algorithm efficiently obtained optimal solutions for all instances.

Furthermore, the solution efficiency of CPLEX for the TOP2-CoTWs model was lower than that for the TOP1-CoTWs model. Conversely, for TOP1-CoTWs, the BPC algorithm demonstrated significantly poorer performance compared to TOP2-CoTWs. Based on these experimental results, it is observed that splitting customers increased the complexity of the directed graph and reduced the solution efficiency of the pricing subproblems. Therefore, when employing the BPC algorithm, it is advisable to minimize the number of vertices.

5.3 Comparisons of different models

To compare the three proposed models over different instances, this study referred to Chen et al. (2019) to calculate the number of variables and constraints, respectively. The results are shown in Table 5 and Appendix A.2 of the Supplementary Material. The following details are provided: the number of variables (#Variables) and the number of constraints (#Constraints) for the three models. Compared to the other models, the TOP2-CoTWs model had considerably fewer variables and constraints for each instance, because customers were not split. Furthermore, the effectiveness of the proposed models was verified on the dataset PS. The results of the CPLEX for the three models are presented in Table 6, where all symbols are defined as shown in Section 5.2. The results indicated that CPLEX could solve the problem more efficiently employing the POP-NoDs model compared to the other two models. More specifically, a superior solution could be obtained by using the TOP1-CoTWs model compared to the TOP2-CoTWs model.

The solution of the TOP2-CoTWs model cost the maximum computational time among those of these three models, although its number of variables and number of constraints were both minimal. This was because the polyhedrons of the feasible region represented by the linear relaxations of different models were different. Moreover, the inherent B&C algorithm of CPLEX might perform better for the model with customer splitting. Therefore, the integrality gap and the computational time at the root node should be combined to express the difficulty of a model. In addition, more variables and constraints indicated a larger memory consumption. When the available memory resources were limited, the solution of the TOP2-CoTW model was more advantageous compared to those of the other two models. Furthermore, compared to the representation of NoDs, when time intervals were represented by CoTWs, the relationship between profits and time intervals could be better described.

To compare the performance of the two BPC algorithms for POP-NoDs and TOP2-CoTWs, we tested them on the dataset PS. Table 7 summarize the results obtained by the two BPC algorithms, and all symbols in the table are defined as shown in Section 5.2. Notably, both algorithms successfully solved all instances to optimality. When solving instances P1-1, P1-2 and P2-1, the BPC algorithm for the POP-NoDs model was more efficient. However, for instances P2-2, P3-1 and P3-2, the BPC algorithm for the POP-NoDs model obtained optimal solutions in 81, 44, and 376 seconds, respectively. However, for the TOP2-CoTWs model, it obtained optimal solutions in 15, 35, and 40 seconds, respectively. Therefore, for small-scale instances, the BPC

algorithm for the POP-NoDs model was advantageous over the other two models, but the difference of the computational times was very small. For medium-scale instances, the BPC algorithm for the TOP2-CoTWs model was superior to those for the other models. Specially, when solving large-scale instances, the advantages of the BPC algorithms for the TOP2-CoTWs were significant. A notable difference was observed in the computational time required by the BPC algorithm when applied to the two distinct modeling approaches (POP-NoDs and TOP2-CoTWs). This disparity stemmed from the significantly lower number of branch-and-bound (B&B) nodes encountered by the BPC algorithm for TOP2-CoTWs.

Table 5

Statistics of the three models.

Instance	POP-NoDs		TOP1-CoTWs		TOP2-CoTWs	
	#Variables	#Constraints	#Variables	#Constraints	#Variables	#Constraints
P1-1	1,482	1,695	1,482	1,705	462	595
P1-2	2,470	2,785	2,520	2,865	820	985
P2-1	5,352	5,775	5,352	5,795	1,512	1,775
P2-2	8,920	9,545	9,020	9,705	2,620	2,945
P3-1	11,622	12,255	11,622	12,285	3,162	3,555
P3-2	19,370	20,305	19,520	20,545	5,420	5,905

Table 6

Results of CPLEX for three models.

Instance	POP-NoDs					TOP1-CoTWs					TOP2-CoTWs				
	z_{rub}	z_{ub}	z_{lb}	*Gap*	*CT*	z_{rub}	z_{ub}	z_{lb}	*Gap*	*CT*	z_{rub}	z_{ub}	z_{lb}	*Gap*	*CT*
P1-1	203	51	51	0	11	206	51	51	0	17	207	51	51	0	15
P1-2	386	190	190	0	125	399	190	190	0	374	401	20	190	0	TL
P2-1	419	55	55	0	480	419	55	55	0	709	424	262	54	0	TL
P2-2	760	491	131	296	TL	760	374	137	186	TL	738	635	165	290	TL
P3-1	604	163	72	132	TL	587	284	53	445	TL	590	308	66	451	TL
P3-2	1,195	1,010	116	936	TL	1,195	1,106	88	1,177	TL	1,204	1,047	184	472	TL

Table 7

Results of the BPC algorithms for the POP-NoDs and TOP2-CoTWs models.

Instance	POP-NoDs				TOP2-CoTWs			
	z_{rub}	z^*	*Node*	*CT*	z_{rub}	z^*	*Node*	*CT*
P1-1	53	51	7	1	52	51	5	3
P1-2	213	190	75	6	190	190	2	6
P2-1	58	55	12	9	56	55	7	19
P2-2	180	165	80	81	171	165	1	15
P3-1	81	75	20	44	82	75	10	35
P3-2	246	213	313	376	233	213	11	40

5.4 Computational performance of the simplification strategy

To evaluate the performance of the proposed simplification strategy, the BPC algorithms were employed to solve the POP-NoDs problem using both the simplified and basic directed graphs. The results are shown in

Table 8 and Appendix A.3 of the Supplementary Material. Table 8 presents the computational results of the instances solved using these two types of directed graphs, including z^*, z_{rub}, *Node* (The three symbols are defined as shown in Section 5.2), and the computational time (CT_1, CT_2). The values in the "*Gap*" column are calculated as: $(CT_1 - CT_2)/CT_1 \times 100\%$. The results demonstrated that the simplification strategy saved an average of 70% of the computational time compared to the original basic strategy. Furthermore, the complexity of the directed graphs played a crucial role in the efficiency of solving subproblems. The complexities of the subproblems significantly influenced the effectiveness of the CG. Hence, reducing the number of unnecessary vertices and arcs in subproblems can improve the solution efficiency of the CG.

Table 8

Results of BPC algorithms based on the basic and simplified directed graphs.

Instance	Basic directed graph			Simplification directed graph			*Gap*
	z^*/z_{rub}	*Node*	CT_1	z^*/z_{rub}	*Node*	CT_2	
P1-1	0.96	8	3	0.96	7	0	8
P1-2	0.91	496	237	0.89	75	6	98
P2-1	0.94	20	25	0.94	12	9	65
P2-2	0.92	34	119	0.92	80	81	49
P3-1	0.91	20	119	0.92	20	44	63
P3-2	0.72	#	#	0.86	313	376	
P4-1	0.90	35	207	0.92	35	88	58
P4-2	0.89	#	#	0.89	#	#	
Average	0.89	102	118	0.91	77	86	70

5.5 Computational time analysis

To evaluate the impact of the parameter λ representing the relaxation degree of the elementary constraints on the dynamic *ng*-path relaxation technique, this study conducted tests to analyze the computational time required by the BPC algorithm using different values of the parameter. Baldacci et al. (2011) experimentally suggested that λ within the range of 10–15 was adequate for generating optimal elementary routes. The results, as presented in Table 9 and further detailed in Appendix A.4 of the Supplementary Material, illustrated the computational time for different values of the parameter λ. The computational time was minimized for $\lambda = 12$ and maximized for $\lambda = 15$. The reason was that the solution efficiency of the BPC algorithm decreased if λ was too large or too small. A parameter that was too small led to poor solution quality and increased the number of iterations required to obtain an optimal solution. Conversely, an excessively large parameter value resulted in high solution complexity.

Table 9

Computational time results of the elementary constraint and relaxation technique.

Instance	Elementary constraints	*Ng*-path relaxation	Dynamic *ng*-path relaxation		
			$\lambda = 10$	$\lambda = 12$	$\lambda = 15$
P1-1	1.74	1.73	1.88	0.36	1.75
P1-2	20.30	20.09	21.47	5.83	19.97
P2-1	10.73	8.89	10.14	8.76	8.61
P2-2	90.61	77.21	73.32	80.55	75.75

Moreover, the analysis also considered the computational time of the BPC algorithms using *ng*-path relaxation and elementary constraints. The problem dimension could be reduced by the relaxation of the elementary constraints, restricting only a subset of customers to be visited once on the same route. As a result, the computational time of the BPC algorithm using relaxation techniques could be reduced. Furthermore, the dynamic *ng*-path relaxation ($\lambda = 12$) offered greater time savings compared to the traditional *ng* -path relaxation by minimizing additional neighborhoods and conducting the CG on a smaller-scale wherever possible. However, in solving certain instances ($\lambda = 10$ for P1-1, P1-2 and P2-1, and $\lambda = 15$ for P1-1), the computational time of the BPC algorithm using dynamic *ng*-path relaxation was longer than that of *ng*-path relaxation. This discrepancy highlighted the potential implications of selecting an inappropriate parameter value, which could hinder the efficiency of the BPC algorithm when utilizing dynamic *ng*-path relaxation.

Both the bidirectional and unidirectional label-setting algorithms were used to solve the pricing subproblems. To improve the solution efficiency of the BPC algorithms, this study also incorporated some valid inequalities into the formulation of the master problem on the fly. Hence, the performance of these algorithmic components was evaluated, and all results were illustrated in Table 10. The following details were outlined: z^*, z_{rub}, *Node* (The three symbols were defined as shown in Section 5.2), and the computational time (CT_1, CT_2, and CT_3) of different strategies, respectively. The values presented in the " Gap_1 " column were calculated as $(CT_1 - CT_3)/CT_3 \times 100\%$, and the values presented in the " Gap_2 " column were calculated as $(CT_2 - CT_3)/CT_3 \times 100\%$. Obviously, the BPC algorithm using the bidirectional label-setting strategy could offer 51.18%–99.28% of computational time savings compared to the BPC algorithm using the unidirectional label-setting strategy. Furthermore, the SR inequalities were incorporated into the master problem to cut off partial fractional solutions, ultimately improving the efficacy of the LP relaxation. Consequently, the introduction of these inequalities led to a significant reduction in the computational time required by the BPC algorithm.

Table 10

Results of the unidirectional label-setting algorithm and the algorithm without the valid inequality.

Instance	Unidirectional label-setting		Without the valid inequality			BPC algorithm			Gap_1	Gap_2
	Node	CT_1	z^*/z_{rub}	*Node*	CT_2	z^*/z_{rub}	*Node*	CT_3		
P1-1	7	7	0.96	7	2	0.96	7	1	95.02	80.38
P1-2	693	810	0.89	85	23	0.89	75	6	99.28	75.02
P2-1	12	43	0.94	14	9	0.94	12	9	79.55	5.47
P2-2	55	337	0.92	90	91	0.92	80	81	76.12	11.65
P3-1	22	90	0.91	19	40	0.92	20	44	51.18	-10.88
P3-2	#	#	0.86	619	1,247	0.86	313	376		69.83
P4-1	32	257	0.90	41	95	0.92	35	88	65.84	8.01
P4-2	#	#	0.89	#	#	0.89	#	#		

5.6 Performance evaluation for the branching strategies

In order to evaluate the effectiveness of the MHB strategy, this study conducted a comparative analysis between the BPC algorithms employing the MHB strategy and the hierarchical HTL branching strategy. The

results are presented in Table 11, where all symbols are defined as shown in Section 5.2. The detailed results for all instances are presented in Appendix A.5 of the Supplementary Material. Notably, the MHB strategy exhibited fewer branching levels compared to the HTL branching strategy, resulting in a decrease of the node size of the B&B search tree and the number of CG iterations, as shown evidenced by the experimental results. Therefore, the solution efficiency of the BPC algorithm could be significantly improved by reducing the levels of branching. Additionally, the results indicated that the gaps between solutions obtained by the MHB strategy and the optimal solutions obtained through the HTL strategy were relatively small, deemed practical, and significantly reduced computational time.

Moreover, due to the CG's relaxation of integer constraints, decision variables z_{tr} and y_{im} in the MP required branching to obtain integer values. The third level of the HTL branching was utilized for variable y_{im}. However, using the MHB strategy allowed for the direct integer conversion of variable y_{im}, replacing the third-level branching. Results demonstrated that the number of decision variables affected the efficiency of the BPC algorithm. Therefore, the number of decision variables should be as small as possible when formulating the model. After determining the decision variables of route sequences, the time interval between two visits and the corresponding profit were obtained. Based on this feature, the MHB strategy was used to replace the third level of branching. Therefore, the branching strategy could be improved by designing targeted methods based on the specific feature of problems.

Table 11

Results of the HTL branching strategy and MHB strategy.

Instance	HTL branching strategy			MHB strategy		
	z^*	*Node*	*CT*	z^*	*Node*	*CT*
P1-1	51	7	1	50	4	1
P1-2	190	75	6	185	2	3
P2-1	55	12	9	55	6	7
P2-2	165	80	81	165	17	21
P3-1	75	20	44	73	15	23
P3-2	213	313	376	178	170	348
P4-1	55	35	88	55	20	49
P4-2	231	#	#	231	#	#

6. Conclusions

This paper investigated the intricate orienteering problem characterized by interval-varying profits, a prevalent challenge in various application scenarios, such as perishable product delivery, home health care scheduling, and earth observation satellite scheduling. To address the complexity, we have undertaken a thorough examination of this feature and focused our efforts on two specific variants of the orienteering problem: POP-NoDs and TOP-CoTWs. For these variants, we have presented mixed-integer linear programming models tailored to capture their unique characteristics. Furthermore, we have developed Branch-Price-and-Cut algorithms to achieve optimal solutions for each problem variant. Additionally, to solve the pricing subproblem, a modified directed graph has been introduced, where a bidirectional label-setting algorithm was performed to discover new columns. Subsequently, to enhance the tightness the LP relaxation bound, subset-row (SR)

inequalities were integrated into our formulations. Finally, we described the hierarchical three-level (HTL) branching strategies employed to derive integer solutions and proposed a matheuristic branching (MHB) strategy as an alternative to the third level of branching, obtaining integer feasible solutions faster.

Experimental results demonstrated that the proposed algorithms perform much better than CPLEX in solving instances of various scales. Notably, the BPC algorithm for the TOP2-CoTWs model could provide superior decision support in shorter planning horizons compared to its counterparts. Furthermore, employing an improved directed graph network offered an average of 70% computational time savings compared to using the basic directed graph network. Therefore, decision makers are advisable to preprocess their data, such as prescheduling customer visits on specific days based on certain rules, to simplify the directed graphs and enhance efficiency. Lastly, the experimental results indicated that the feasible solutions obtained through the matheuristic branching strategy are close to optimal, with minimal gaps. This practical acceptability, coupled with significant computational time savings, makes our approach a viable and efficient tool for decision makers.

Several aspects could be considered in the future research. In this study, two representations of NoDs and CoTWs were utilized to approximatively formulate time intervals, employing a discrete profit function. However, there are scenarios where longer time windows and significant interval-varying profits render the current approach suboptimal. Consequently, a continuous profit function that better captures customer satisfaction and ultimately increases profits will be a focal point of future work. Furthermore, we aim to expand the scope of our problem to incorporate multiple visits, recognizing that profits are influenced not only by time intervals but also by the number and frequency of visits. While this extension holds broad applicability, it presents significant challenges in constructing profit functions and designing solution algorithms. Lastly, given the inevitable presence of uncertainty in duration and travel times in practical settings, we will delve into robust TOP under uncertainty. Our future efforts will be directed towards developing a robust formulation of the optimization problem to mitigate the impact of these uncertainties and enhance the resilience of our approach.

Acknowledgments

This research was supported by the Science and Technology Innovation Program of Hunan Province (Grant No. 2022RC1241), the National Natural Science Foundation of China (Grant No. 72071075), the National Defense Basic Scientific Research Program (Grant No. WDZC20235250411), the China Postdoctoral Science Foundation (Grant No. 2023M744331), the National Key Laboratories Fund of China (Grant No. 6142101230306) and the General Project of Postgraduate Scientific Research Innovation Project of Hunan Province (Grant No. CX20230069).

References

Afsar, H., & Labadie, N. (2013). Team orienteering problem with decreasing profits. *Electronic Notes in Discrete Mathematics*, 41, 285–293.

Baldacci, R., Bartolini, E., & Mingozzi, A. (2011). An exact algorithm for the pickup and delivery problem with time windows. *Operations Research*, 59(2), 414–426.

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

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

Carter, M. W., Farvolden, J. M., Laporte, G., & Xu, J. F. (1996). Solving an integrated logistics problem arising in grocery distribution. *INFOR: Information Systems and Operational Research*, 34(4), 290–306.

Chao, I. M., Golden, B., & Wasil, E. A. (1996). The team orienteering problem. *European Journal of Operational Research*, 88(3), 464–474.

Chen, X. Y., Reinelt, G., Dai, G. M., & Spitz, A. (2019). A mixed integer linear programming model for multi-satellite scheduling. *European Journal of Operational Research*, 275(2), 694–707.

Dohn, A., Kolind, E., & Clausen, J. (2009). The manpower allocation problem with time windows and job-teaming constraints: A branch-and-price approach. *Computers & Operations Research*, 36(4), 1145–1157.

Dontas, M., Sideris, G., Manousakis, E. G., & Zachariadis, E. E. (2023). An adaptive memory matheuristic for the set orienteering problem. *European Journal of Operational Research*, 309(3), 1010–1023.

Duhamel, C., & Santos, A. C. (2024). The strong network orientation problem. *International Transactions in Operational Research*, 31(1), 192–220.

Erdogan, G., & Laporte, G. (2013). The orienteering problem with variable profits. *Networks*, 61, 104–116.

Erkut, E., & Zhang, J. (1996). The maximum collection problem with time dependent rewards. *Naval Research Logistics*, 43(5), 749–763.

Golden, B. L., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, 34(3), 307–318.

Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research*, 255(2), 315–332.

Gunawan, A., Ng, K. M., Kendall, G., & Lai, J. H. (2018). An iterated local search algorithm for the team orienteering problem with variable profits. *Engineering Optimization*, 50(7), 1148–1163.

Hammami, F., Rekik, M., & Coelho, L. C. (2020). A hybrid adaptive large neighborhood search heuristic for the team orienteering problem. *International Transactions in Operational Research*, 123, 105034.

Hanafia, S., Mansinib, R., & Zanottib, R. (2020). The multi-visit team orienteering problem with precedence constraints. *European Journal of Operational Research*, 282(2), 515–529.

Jepsen, M., Petersen, B., Spoorendonk, S., & Pisinger, D. (2008). Subset-row inequalities applied to the vehicle-routing problem with time windows. *Operations Research*, 56(2), 497–511.

Keshtkaran, M., Ziarati, K., Bettinelli, A., & Vigo, D. (2016). Enhanced exact solution methods for the team orienteering problem. *International Journal of Production Research*, 54(2), 591–601.

Kim, H., Kim, B. I., & Noh, D. (2020). The multi-profit orienteering problem. *Computers & Industrial Engineering*, 149, 106808.

Li, J. L., Qin, H., Baldacci, R., & Zhu, W. B. (2020). Branch-and-price-and-cut for the synchronized vehicle routing problem with split delivery, proportional service time and multiple time windows. *Transportation Research Part E: Logistics and Transportation Review*, 140, 101955.

Lübbecke, M., & Desrosiers, J. (2005). Selected topics in column generation. *Operations Research*, 53(6), 1007–1023.

Luo, Z. X., Qin, H., & Zhu, W. B. (2016). Branch-and-price-and-cut for the manpower routing problem with synchronization constraints. *Naval Research Logistics*, 63(2), 138–171.

Orlis, C., Bianchessi, N., Roberti, R., & Dullaert, W. (2020). The team orienteering problem with overlaps: An application in cash logistics. *Transportation Science*, 54(2), 470–487.

Peng, G. S., Dewil, R., Verbeeck, C., Gunawan, A., Xing, L. N., & Vansteenwegen, P. (2019). Agile earth observation satellite scheduling: An orienteering problem with time-dependent profits and travel times. *Computers & Operations Research*, 111, 84–98.

Pietz, J., & Royset, J. O. (2013). Generalized orienteering problem with resource dependent rewards. *Naval Research Logistics*, 60(4), 294–312.

Rasmussen, M. S., Justesen, T., Dohn, A., & Larsen, J. (2012). The home care crew scheduling problem: Preference-based visit clustering and temporal dependencies. *European Journal of Operational Research*, 219(3), 598–610.

Righini, G., & Salani, M. (2006). Symmetry helps: Bounded bi-directional dynamic programming for the elementary shortest path problem with resource constraints. *Discrete Optimization*, 3(3), 255–273.

Roberti, R., & Mingozzi, A. (2014). Dynamic ng-path relaxation for the delivery man problem. *Transportation Science*, 48(3), 413–424.

Rothenbächer, A. K. (2019). Branch-and-price-and-cut for the periodic vehicle routing problem with flexible schedule structures. *Transportation Science*, 53(3), 850–866.

Santini, A., Plum, C. E. M., & Ropke, S. (2018). A branch-and-price approach to the feeder network design problem. *European Journal of Operational Research*, 264(2), 607–622.

Santini, A., & Archetti, C. (2023). The hazardous orienteering problem. *Networks*, 81(2), 235–252.

Souffriau, W., Vansteenwegen, P., VandenBerghe, G., & VanOudheusden, D. (2013). The multiconstraint team orienteering problem with multiple time windows. *Transportation Science*, 47(1), 53–63.

Sundar, K., Sanjeevi, S., & Montez, C. (2022). A branch-and-price algorithm for a team orienteering problem with fixed-wing drones. *EURO Journal on Transportation and Logistics*, 11, 100070.

Tang, H., & Miller-Hooks, E. (2005). A tabu search heuristic for the team orienteering problem. *Computers & Operations Research*, 32(6), 1379–1407.

Urrutia-Zambrana, A., Tirado, G., & Mateos, A. (2021). Variable neighborhood search to solve the generalized orienteering problem. *International Transactions in Operational Research*, 28(1), 142–167.

Vansteenwegen, P., Souffriau, W., & Oudheusden, D. V. (2011). The orienteering problem: A survey. *European Journal of Operational Research*, 209(1), 1–10.

Vincent, F. Y., Jewpanya, P., Lin, S. W., & Redi, A. A. N. P. (2019). Team orienteering problem with time windows and time-dependent scores. *Computers & Industrial Engineering*, 127, 213–224.

Wang, J. J., Demeulemeester, E., & Qiu, D. S. (2016). A pure proactive scheduling algorithm for multiple earth observation satellites under uncertainties of clouds. *Computers & Operations Research*, 74, 1–13.

Wang, J. J., Song, G. P., Liang, Z., Demeulemeester, E., Hu, X. J., & Liu, J. (2023). Unrelated parallel machine scheduling with multiple time windows: An application to earth observation satellite scheduling. *Computers & Operations Research*, 149, 106010.

Wang, Z., & Sheu, J. B. (2019). Vehicle routing problem with drones. *Transportation Research Part B: Methodological*, 122, 350–364.

Wu, Q. H. (2024). An effective hybrid evolutionary algorithm for the clustered orienteering problem. *European Journal of Operational Research*, 313(2), 418–434.

Xu, W., Liang, W., Xu, Z., Peng, J., Peng, D. Z., Liu, T., Jia, X. H., & Das, S. K. (2020). Approximation algorithms for the generalized team orienteering problem and its applications. *IEEE/ACM Transactions on Networking*, 29(1), 176–189.

Yahiaoui, A. E., Moukrim, A., & Serairi, M. (2019). The clustered team orienteering problem. *Computers & Operations Research*, 111, 386–399.

Yahiaoui, A. E., Moukrim, A., & Serairi, M. (2023). GRASP-ILS and set cover hybrid heuristic for the synchronized team orienteering problem with time windows. *International Transactions in Operational Research*, 30(2), 946–969.

Yu, Q. X., Adulyasak, Y., Rousseau, L. M., Zhu, N., & Ma, S. F. (2022). Team orienteering with time-varying profit. *INFORMS Journal on Computing*, 34(1), 262–280.

Yu, Q. X., Fang, K., Zhu, N., & Ma, S. F. (2019). A matheuristic approach to the orienteering problem with service time dependent profits. *European Journal of Operational Research*, 273(2), 488–503.