Design of diversified package tours for the digital travel industry : a branch-cut-and-price approach

Yanlu Zhao, Laurent Alfandari*

ESSEC Business School, Paris

Abstract

Motivated by the revolution brought by the internet and communication technology in daily life, this paper examines how the online travel agencies (OTA) can use these technologies to improve customer value. We consider the design of a fixed number of package tours offered to customers in the digital travel industry. This can be formulated as a Team Orienteering Problem (TOP) with restrictions on budget and time. Different from the classical TOP, our work is the first one to introduce controlled diversity between tours. This enables the OTA to offer tourists a diversified portfolio of tour packages for a given period of time, each potential customer choosing a single tour in the selected set, rather than multiple independent tours over several periods as in the classical TOP. Tuning the similarity parameter between tours enables to manage the trade-off between individual preferences in consumers' choices and economies of scale in agencies' bargaining power. We propose compact and extended formulations and solve the master problem by a branch-and-price method, and an alternative branch-cut-and-price method. The latter uses a delayed dominance rule in the shortest path pricing problem solved by dynamic programming. Our methods are tested over benchmark TOP instances of the literature, and a real dataset collected from a Chinese OTA. We explore the impact of tours diversity on all stakeholders, and assess the computational performance of various approaches.

Keywords: Integer programming, team orienteering problem, diversity, branch-cut-and-price, digital travel industry

1. Introduction

This paper considers a variant of the Team Orienteering Problem (TOP) where pairwise diversity constraints hold between tours. The TOP has been extensively studied in the literature in the last two decades (Chao et al., 1996a,b; Archetti et al., 2006, 2007, 2013, 2015; Boussier et al.,

*Corresponding author
Email addresses: yanlu.zhao@essec.edu (Yanlu Zhao), alfandari@essec.edu (Laurent Alfandari)

2007; Vansteenwegen et al., 2009a,b, 2011; Poggi et al., 2010; Labadie et al., 2012; Lin & Vincent, 2012; Dang et al., 2013; Ke et al., 2013; Luo et al., 2013; Tarantilis et al., 2013; Hu & Lim, 2014; Verbeeck et al., 2014a; Divsalar et al., 2014; Mei et al., 2016; Gunawan et al., 2016; Riera-Ledesma & Salazar-González, 2017). Contrary to the classical TOP, our problem is to design a set of tours for an Online Travel Agency (OTA) such that any potential customer will choose a single tour in the portfolio at a given period. This allows common cities in some tours to ensure economies of scale for the OTA when negotiating prices with suppliers, in a competitive context of satisfying customers' needs with more attractive and diverse travel products. The problem is motivated by the needs of a specific Chinese OTA to improve its portfolio of online products. The explosive progress in information and communication technology (ICT), especially in the accessible Internet and mobile devices, facilitates consumers to search and discover desirable tour products instantly (Lewis et al., 1998; Abbaspour & Samadzadegan, 2011). According to a survey conducted over US adults, more than 95% of respondents said they prefer an online channel to search for vacations and over 58% said they use user-generated content on the Internet for travel planning. Overall 564.87 billions USD of travel products, including air flight, hotel, accommodation, car rental, etc., were sold online worldwide in 2016 (Statista, 2016). The huge increase in digital transactions of travel products naturally leads to a general revenue growth for online travel agencies such as Expedia and Priceline (Prieto, 2017). The OTAs, or so-called digital tour operators, provide travel-related information and service to consumers through the Internet. Their online products, for example, weekly or monthly tours bundling transportation, accommodations and sightseeing, release vast consumers from the tedious work of searching trip components and evaluating tour compositions (Sheldon & Mak, 1987; Wong & Kwong, 2004). However, there is still improvement potential for OTAs. Another study in the US reveals that over 68% and 49% of consumers completed their final flight and hotel bookings from direct suppliers, rather than from OTA channels (Statista, 2015). An emerging shopping trend is that a large proportion of consumers resort to the travel recommendation (guidance, reviews, suggestions, etc.) on the OTAs' websites at the information acquisition and pre-purchase stages, while they skip their digital products and go for the suppliers to place customized orders at the purchase stage (Chiappa, 2013). Such a transformation reduces the transaction success rate, worsens the operational environment and poses great challenges for the OTA players (Werthner, 2002). Therefore, to yield higher revenue, OTAs need to provide more attractive travel products with ambitious quality and competitive price.

In this article, we describe a new problem of travel products design. Our research is motivated by the digital tour design operations of a Chinese OTA (http://www.niding.net), whose annual sales value was around USD 100 million on average from 2016 to 2018. Customers indicate their

requirement on a few *resources* (typically, time and budget) and potential vacation *destinations* to the company. After collecting information on candidate destinations, dwelling times, travel horizon, travel data concerning flights and hotels, the company offers customers a selected set of best itineraries under resource restriction.

Currently, the company owns a data-driven point of interests (POIs) identification system to help capture more attention from customers. It seeks to increase its transaction success rate to improve online sales performance: as they reported, 4% of page views on average generate a completed transaction order in the online travel industry. Conflicting criteria are at stake: customization of tours versus uniformity. Indeed, if the OTA designs personalized tours for each individual, the over-dispersed trips will blow up the number of cities visited. This not only increases the workload of the OTA employees and cost, but also prevents the company from taking advantage of the economies of scale (discounting) benefits when bargaining with upstream suppliers, thus leading to sales losses because of high prices (Clemons et al., 2002). On the reverse, if the OTA website offers only one standard set of cities meeting some resource requirement, the homogeneous products cannot cover the diversified demand from various consumers, which may decrease sales due to a narrow range of products. As a result, the company is trapped into a dilemma about the diversity of its digital products.

The contributions of the paper are the following. We study a new problem named TOP-DC, which is a Team Orienteering Problem with additional Diversity Constraints between the m selected tours that meet the resource requirements. To manage the trade-off between individual preferences of customers (customization) and purchase discount from suppliers (uniformity and economies of scale), we introduce a maximum similarity parameter s on the number of common cities between any two tours, and present a compact formulation for the TOP-DC incorporating the diversity constraints. We propose a two-index extended formulation and branch-and-price algorithm. Because of symmetries arising in the former formulation, we design a more complex branch-cut-and-price algorithm based on a one-index reformulation that avoids symmetry phenomena. This method uses row generation with diversity cuts generated "on the fly", which requires to adapt the dynamic programming procedure with a tailored delayed dominance rule for the pricing subproblem. Finally, we conduct numerical experiments with a benchmark dataset from the literature and a real dataset from the Chinese OTA, to analyze the performance of the above methods and to derive managerial insights.

The paper is structured as follows. Section 2 provides a literature review on the related TOP. Section 3 presents the compact formulation for the TOP-DC. Section 4 describes the two-index extended formulation and associated branch-and-price method, with a focus on the pricing sub-

problem. Section 5 gives the one-index master reformulation, the associated branch-cut-and-price method and modification of the dominance rule for the subproblem. Section 6 describes the results of numerical experiments and detailed analysis. Section 7 gives concluding remarks.

2. Literature review

Our research lies on the intersection of several related streams of literature: travel industry, tour planning, and solution methods. In this section, we review key contributions of each stream of literature and position our paper with the existing research.

2.1. Research on the travel industry

The travel industry has been extensively studied in the literature. It comprises, among others, studies on tour packages (Sheldon, 1986; Sheldon & Mak, 1987; Morrison, 1989), tour operators (Sheldon, 1986; Heung & Chu, 2000), and tour planning (Laporte & Martello, 1990; Fischetti et al., 1998; Abbaspour & Samadzadegan, 2011). See Law et al. (2004) for detailed reviews.

In the travel industry, tour operators negotiate with hotels, transportation companies and other suppliers, then combine their products into a tour product (Sheldon, 1986). A tour product is a combination of several heterogeneous components in a vacation, such as transportation, accommodation, sightseeing and meals, which are sold to customers at a single price (Sheldon & Mak, 1987; Heung & Chu, 2000). The tour products enable tourists to visit a large number of sites on a trip with constrained resources through a relatively safe way (Enoch, 1996; Wong & Kwong, 2004). However, the design of tour products is not only determined by the popularity of its destinations, but also by its profitability for operators. If operators could acquire a large volume discount from suppliers, they would offer better prices to customers. The existence of tour operators is beneficial to both suppliers and consumers, not only because they increase sales and decrease promotional costs for suppliers, but also they reduce customers' transaction costs by limiting communications and bookings to one operator rather than many suppliers (Sheldon, 1986).

However, a few inevitable drawbacks can be found with the travel products provided by tour operators. First of all, operators prefer to provide uniform products in order to condense same destinations and exploit economies of scale (Lee et al., 2013), which does not favor customization. If tourists require some personalized trips, such products have to be designed individually for each customer, and by diversifying the travel products, the OTA may disperse customers to many over-spreading short trips, losing economies of scale in purchasing from upstream suppliers. On the reverse, concentrating the demand on fewer tours would enable the OTA to negotiate better prices with suppliers. Finding a good compromise or trade-off between customization and uniformization of

tour products has not been thoroughly investigated so far, to the best of our knowledge. Addressing this issue is one main objective of this paper.

2.2. Research on tour planning and orienteering problems

Tour planning usually refers to generating a schedule or itinerary to visit some POIs in a transportation network while satisfying some objectives and constraints (e.g., time and budget) (Gavalas et al., 2014). The operations research literature on tour planning is vast. We describe the state-of-the-art problems most related to ours, namely the Orienteering Problem and Team Orienteering Problem, then we focus on exact solving approaches for these problems.

Orienteering Problem. The Orienteering Problem (OP), also called the selective traveling salesperson problem (Laporte & Martello, 1990; Chao et al., 1996a; Righini & Salani, 2006), or the maximum collection problem (Butt & Cavalier, 1994), has received increasing attention during the last decades. We refer the reader to the survey by Gunawan et al. (2016) for an extensive review on this problem and its practical applications. In this problem, a set of control points is given, along with associated scores and a connecting network. The OP deals with finding a path from specific start and end points with maximum total score, subject to a given set of constraints (Boussier et al., 2007). Obviously, due to the resource constraints, the decision-maker might exclude some POIs (Abbaspour & Samadzadegan, 2011). According to Chao et al. (1996a), the OP can be seen as a two-level optimization problem. At the first decision level, one chooses a subset of nodes to visit. At the second level, one solves a maximization Traveling Salesman Problem (TSP) over the selected nodes. Since the TSP is NP-hard, so is the OP (Laporte & Martello, 1990). Often in practice, the tour operator has to design different tours over several days, turning the one-tour Orienteering Problem into a multiple-tour Team Orienteering Problem.

Team Orienteering Problem. The extension of the OP to multiple tours, which is a special case of the VRP with profits, was introduced under the name of Team Orienteering Problem by Chao et al. (1996b). The TOP (Tang & Miller-Hooks, 2005) or multiple tour maximum collection problem (MTMCP) (Butt & Cavalier, 1994) is to find a set of m paths (tours), each constrained by a time limit T, that maximizes the total collected scores of selected nodes. The main difference between the TOP and classical vehicle routing problem (VRP) is that not all nodes have to be visited in the TOP (Boussier et al., 2007). Consequently, the TOP can be seen as a three-level optimization problem. The first decision level is to select the subset of nodes to be visited overall, the second level is to assign nodes to each team member, and the third level is to construct a path connecting the nodes assigned to each team member (Chao et al., 1996b). The TOP is at least as difficult as the OP, which is a special TOP with $m = 1$.

An application of the TOP in the travel industry is called the Tourist Trip Design Problem (TTDP), which is defined as a route-planning problem for tourists interested in visiting multiple POIs (Godart, 1999; Gunawan et al., 2016). As mentioned before, most of the TTDP correspond to the OP or TOP, we refer readers to the survey by Gavalas et al. (2014) for a comprehensive review on this problem.

Recently, Souffriau et al. (2008) developed a mobile tourist guide to help travelers design their one-day or several-days trip. Herzog & Wörndl (2014) studied another OP variant for an individual user where trips are composed of multiple regions. Verbeeck et al. (2014b) introduced another OP-like variant called Cycle Trip Planning Problem (CTPP), which is to find a closed path maximizing the total collected score. At last, Malucelli et al. (2015) studied the problem of designing the most attractive itineraries for different classes of users with different preference patterns, which was formulated as a multi-commodity OP with a single origin-destination pair.

To our knowledge, most of the tour planning papers focus on the optimal trip design in a vertical view, that is, to provide one/several-days tours for a tourist. When the TOP is applied to the travel industry, a customer follows each of the m tours, one every day. However, in this paper, we design tours in a horizontal view. Our aim is to propose a set of m possible tours with given time and budget so that each customer chooses only one tour in the set, but these m tours should be diverse enough to offer the customer a wide variety of choice for her tour.

2.3. Research on exact methods

Previous studies listed several methods for the TOP and OP, ranging from exact methods to heuristics and metaheuristics. Heuristics include local search (Chao et al., 1996a), tabu search (Tang & Miller-Hooks, 2005), variable neighborhood search (Vansteenwegen et al., 2009b) and simulated annealing (Sylejmani et al., 2014). We could also find metaheuristics in Archetti et al. (2007, 2015) and Vansteenwegen et al. (2009a). In this paper, we only cover studies on exact methods. For a general review on solution approaches, we refer readers to Gavalas et al. (2014) and Gunawan et al. (2016).

Branch and Price (BP). Branch-and-Price is widely used when solving vehicle routing problems with exact methods (Costa et al., 2018). A few BP algorithms were proposed to solve the TOP. Butt & Ryan (1999) gave a set-partitioning formulation, and was the first to use a column generation approach, solving instances with up to 100 nodes. Later, branch-and-price, i.e, branch-and-bound where the LP-relaxation is solved by column generation, was applied in Boussier et al. (2007) for solving the TOP to optimality. This BP scheme was also used to solve the TOP with time-windows. Keshtkaran et al. (2016) proposed an enhanced branch-and-price method to solve the TOP on the

basis of Boussier's work. Recently, Riera-Ledesma & Salazar-González (2017) proposed two exact column generation algorithms to solve the team orienteering arc routing problem efficiently.

Branch Cut and Price (BCP). Cutting plane methods were also adopted to solve TOP instances. Fischetti et al. (1998) presented a branch-and-cut algorithm by adding some valid inequalities and conditional cuts, and their method was able to solve instances with 500 cities to optimality. Poggi et al. (2010) proposed a branch-cut-and-price algorithm for solving the classical TOP, where a few additional inequalities and cuts for the problem were incorporated to improve the bounds obtained by column generation. Archetti et al. (2013) proposed some valid and facet-inducing inequalities in their branch-and-cut algorithm and tested their algorithm on large-scale instances.

We only found one paper, Song et al. (2018), mentioning similarity among tours for the TOP. However, they measured similarity of solutions ex post, whereas similarity is explicitly modeled in the problem definition in our case. The introduction of diversity constraints generates higher complexity and potential symmetry in the solutions depending on the formulation, which will be further discussed in the following sections.

3. Compact formulation

The Team Orienteering Problem with Diversity Constraints (TOP-DC) can be defined on a complete directed graph $G = (V, A)$, where $V = \{0, ..., n\}$ is the set of nodes representing cities, and $A = \{(i, j)\}(i \neq j)$ is the set of arcs. Among the node set V, nodes 0 and n represent the hub city as initial and terminal node of each tour. For convenience, we note $\overline{V} = V \setminus \{0, n\}$. Travel times and costs on arcs $(i, j) \in A$ are noted t_{ij} and c_{ij}. Travel times t_{ij} include both the flight time to connect i and j, and the time spent at city i to make the scheduled visits and resting. Moreover, u_i is the customer utility or satisfaction collected when visiting city $i \in \overline{V}$, measuring the attractiveness of the city. The time and budget limits of a tour are noted t_{max} and c_{max}. We define the TOP-DC problem as follows.

Definition 1. *The Team Orienteering Problem with Diversity Constraints is to find m paths (tours) with time (and/or budget) no more than t_{max} (resp., c_{max}) such that any two paths share no more than s common nodes in $\overline{V}$, while maximizing the total utility collected.*

Obviously, the classical TOP is a special TOP-DC with similarity parameter $s = 0$, which implies that a node is in no more than one of the m tours. Therefore, the TOP-DC is also NP-hard. When $s = |\overline{V}|$, i.e., similarity between tours is unconstrained, the problems turns into the OP as the m tours will be identical for maximizing score, so the model will output a single tour. The TOP-DC

also differs from the split-delivery VRP in that it obtains all the utility once visiting a node rather than a proportion (Archetti et al., 2006). To model the TOP-DC as a Mixed-Integer Program (MIP), we index the m tours by $k = 1, \ldots, m$ and define the following decision variables:

$$x_{ij}^k = \begin{cases} 1, & \text{if arc } (i,j) \text{ is selected in tour } k,\ (i,j) \in A, k = 1, ..., m, \\ 0, & \text{otherwise.} \end{cases}$$

$z_i^{kk'} =$ if node i is visited by both tours k and k', $\forall k' > k, k = 1, ..., m$,

$f_{ij}^k =$ cumulative cost spent until visiting arc (i,j) in tour $k, k = 1, ..., m$.

The objective is to maximize the total utility of the m tours:

$$\max \sum_{k=1}^{m} \sum_{(i,j)\in A} u_i x_{ij}^k \tag{1}$$

The constraints can be partitioned in two parts: (i) *local* tour-definition constraints for each tour, and (ii) *global* diversity constraints linking all tours. The tour definition constraints are as follows:

$$\sum_{(i,j)\in A} t_{ij} x_{ij}^k \leq t_{max}, k = 1, ..., m \tag{2}$$

$$\sum_{(i,j)\in A} c_{ij} x_{ij}^k \leq c_{max}, k = 1, ..., m \tag{3}$$

$$\sum_{j\in \overline{V}} x_{0j}^k = 1, k = 1, ..., m \tag{4}$$

$$\sum_{i\in \overline{V}} x_{in}^k = 1, k = 1, ..., m \tag{5}$$

$$\sum_{(i,j)\in A} x_{ij}^k = \sum_{(j,l)\in A} x_{jl}^k, \forall j \in \overline{V}, k = 1, ..., m \tag{6}$$

$$\sum_{j\in V, i\neq j} x_{ij}^k \leq 1, \forall i \in V, k = 1, ..., m \tag{7}$$

$$x_{ij}^k + x_{ji}^k \leq 1, \forall (i,j) \in A, k = 1, ..., m \tag{8}$$

$$\sum_{j\in V} f_{ij}^k - \sum_{r\in V} (f_{ri}^k + c_{ri} x_{ri}^k) = 0, \forall i \in \overline{V}, k = 1, ..., m \tag{9}$$

$$c_{ij} x_{ij}^k \leq f_{ij}^k \leq (c_{max} - c_{ij}) x_{ij}^k, \forall (i,j) \in A, k = 1, ..., m \tag{10}$$

$$f_{0j}^k = 0, \forall j \in \overline{V}, k = 1, ..., m \tag{11}$$

$$f_{ij}^k \geq 0, \forall (i,j) \in A, k = 1, ..., m \tag{12}$$

$$x_{ij}^k \in \{0,1\}, \forall (i,j) \in A, k = 1, ..., m \tag{13}$$

Constraints (2)-(3) ensure that the total time and cost of a tour do not exceed resource capacity t_{max} and c_{max}. Constraints (4)-(5) state that a tour can only leave and return the hub once. Constraints (6) are classical flow balance equations on each node in $\overline{V}$. Constraints (7)-(8) enforce each arc and edge to be selected at most once. Constraints (9)-(11) are the single-commodity flow constraints used to ensure the flow conservation and eliminate any subtour without the hub. To incorporate the diversity restrictions, the TOP-DC formulation also includes the global constraints linking different tours k and k':

$$\sum_{j \in V} x_{ij}^{k} + \sum_{j \in V} x_{ij}^{k'} - z_i^{kk'} \leq 1, \forall i \in \overline{V}, k = 1, ..., m, k' > k \tag{14}$$

$$\sum_{i \in \overline{V}} z_i^{kk'} \leq s, k = 1, ..., m, k' > k \tag{15}$$

$$z_i^{kk'} \geq 0, \forall i \in \overline{V}, k = 1, ..., m, k' > k \tag{16}$$

Continuous Variables $z_i^{kk'}$ indicates if node i is visited by both tours k and k', with $z_i^{kk'} = 0$ if node i is not shared by the two tours. Constraints (14) will force $z_i^{kk'} = 1$ if node i is in both tours k and k', and constraints (15) guarantee that the number of common nodes between two tours k and k' does not exceed s.

The above formulation (1)-(16) includes at least $O(m|A|)$ variables and $O(m^2 n)$ constraints. This becomes intractable for commercial solvers for medium to large-size instances, as we see further in the numerical experiments. Thus, we explore other solution approaches in the following sections: a Branch-and-Price (BP) and a Branch-Cut-and-Price (BCP) algorithm.

4. Branch-and-Price method

Branch and Price is a method to solve mixed integer linear programs with many variables (Barnhart et al., 1998). It is a branch and bound method in which at each node of the search tree the LP-relaxation is computed by column generation. It begins by solving a restricted master problem (RMP) and a pricing subproblem. If the optimal solution of the RMP is not integer, then a branching strategy is applied by adding constraints to derive integral solutions.

4.1. Two-index extended formulation

Let $\mathcal{R}$ represent the set of feasible tours for a tourist, that is, a set of tours originating from the hub city 0 and ending at the hub city n, with total time and cost at most t_{max} and c_{max}, respectively. A city in $\overline{V}$ can be visited at most once in each tour $r \in \mathcal{R}$, but it can appear in several of the m tours. Let $a_i^r = 1$ if city $i \in \overline{V}$ is visited in tour $r \in \mathcal{R}$, $a_i^r = 0$ otherwise. One notes $p_r = \sum_{i \in V} a_i^r u_i$

the total utility collected by tour $r \in \mathcal{R}$. We introduce the following decision variables:

$$\lambda_{rk} = \begin{cases} 1 & \text{if column } r \text{ is assigned to tour index } k = 1, ..., m, \\ 0 & \text{otherwise.} \end{cases}$$

$$z_i^{kk'} = \text{if node } i \text{ is visited by both tours } k \text{ and } k', \forall k' > k, k = 1, ..., m.$$

Then the compact formulation (1)-(16) can be reformulated as the following Master Problem (MP) with a compact set of $O(m^2)$ constraints:

$$\max \sum_{k=1}^{m} \sum_{r \in \mathcal{R}} p_r \lambda_{rk} \tag{17}$$

subject to

$$\sum_{r \in \mathcal{R}} \lambda_{rk} \leq 1, \forall k = 1, ..., m \tag{18}$$

$$\sum_{r \in \mathcal{R}} a_i^r \lambda_{rk} \leq 1, \forall i \in \overline{V}, k = 1, ..., m \tag{19}$$

$$\sum_{r \in \mathcal{R}} a_i^r \lambda_{rk} + \sum_{r \in \mathcal{R}} a_i^r \lambda_{rk'} - z_i^{kk'} \leq 1, \forall i \in \overline{V}, k = 1, ..., m, k' > k \tag{20}$$

$$\sum_{i \in \overline{V}} z_i^{kk'} \leq s, k = 1, ..., m, k' > k \tag{21}$$

$$\lambda_{rk} \in \{0, 1\}, k = 1, ..., m, k' > k \tag{22}$$

$$z_i^{kk'} \geq 0, \forall i \in \overline{V}, k = 1, ..., m, k' > k \tag{23}$$

The objective function (17) maximizes the sum of the utilities of the selected tours (or equivalently, the average utility over the m tours). Constraints (19) states that a city cannot be visited more than once in a tour, which strengthen the formulation and generate less branching nodes. Constraints (20)-(21) ensure that no more than s cities are in common in any two tours.

4.2. Column generation scheme

The well-known iterative principle of column generation can be summarized as follows on the TOP-DC (Lusby et al., 2017). Let $\overline{\text{MP}}$ denote the linear relaxation of MP. The aim is to solve $\overline{\text{MP}}$ to optimality and get a lower bound. We start to solve a Restricted Master Problem (RMP), i.e., problem $\overline{\text{MP}}$ restricted to a small set of tours (columns) $\hat{\mathcal{R}} \subset \mathcal{R}$, by the Simplex method. This LP-solving of the RMP provides dual variables $\alpha_k, \sigma_i^k, \mu_i^{kk'}$ associated with the respective constraints (18)-(20). Then one checks whether there exists a tour $r \in \mathcal{R}$ with positive reduced utility that could be added to the RMP in order to improve the LP bound. If no such improving column exists then

the current solution of the last RMP is optimal for $\overline{\text{MP}}$, otherwise a subset of columns with positive reduced-utility is added to the RMP and one reiterates the process until no improving column is found. Finding a column with positive reduced utility at each iteration of column generation is called the Pricing problem. Finally, all columns with positive reduced utility are added to the RMP, and also no column generation stabilization technique is used.

4.3. Pricing problem

Given the current dual variables $\alpha_k, \sigma_i^k, \mu_i^{kk'}$ output by the LP-solving of the last RMP, the reduced utility of a column $r \in \mathcal{R}$ for an assigned tour k is:

$$\begin{aligned} \overline{c}_r^k &= p_r - \alpha_k - \sum_{i \in \overline{V}} a_i^r (\sigma_i^k + \sum_{k'>k} \mu_i^{kk'} + \sum_{k'<k} \mu_i^{k'k}) \\ &= \sum_{(i,j) \in A} b_{ij}^r \overline{u}_{ij}^k - \alpha_k \quad (\text{where } \overline{u}_{ij}^k = u_i - \sigma_i^k - \sum_{k'>k} \mu_i^{kk'} - \sum_{k'<k} \mu_i^{k'k}) \end{aligned} \tag{24}$$

The modified-utility of a node i is conveniently located on its outgoing arcs, hence the $\overline{u}_{ij}^k$ notation in (24). The parameter b_{ij}^r indicates whether arc (i, j) will be used in the generated column $r \in \mathcal{R}$ and obtained through the dynamic programming method introduced later. Then, solving the subproblems amounts to find a shortest path r on the modified support graph $G^k (k = 1, ..., m)$ with values only on arcs. Since the solution tours must respect the time and budget constraints, it is an NP-hard elementary shortest path problem with resource constraints (ESSPRC). In the following, we consider only one support graph G^k.

The ESSPRC on G^k can be solved by dynamic programming using a labeling algorithm (Feillet et al., 2004; Irnich & Desaulniers, 2005; Desaulniers et al., 2016), where labels are used to represent partial paths that start from the hub city 0. Starting from an initial label associated with 0, paths are constructed iteratively by extending this label and its descendents forwardly in G^k, using Resource Extension Functions (REFs). Each generated label is checked for feasibility with respect to the resource limitations and infeasible labels are discarded. Furthermore, since the time complexity of the dynamic programming is exponential because of the explosion of labels, in order to avoid enumerating all feasible $0 - n$ paths, a dominance check eliminates partial paths that are impossible to appear in an optimal solution, as shown by Feillet et al. (2004).

In a forward labeling algorithm for ESPPRC, a partial path p from hub 0 to a node $i \in V$ is represented by a label $L_i = (U_i, T_i, C_i, (N_i^v)_{v \in V})$, where the label components are as follows:

- U_i: reduced utility of path p;
- T_i: total time used along path p;

- C_i: total money spent along path p;
- N_i^v: binary value indicating whether city v has been visited or not along the partial path p. It is also set to 1 if city v is not visited but it is unreachable from p. A city v is said to be unreachable if $T_i + t_{iv} > t_{max}$ or $C_i + c_{iv} > c_{max}$, in which case it cannot be part of any feasible extension of path p.

The initialization of the label at node 0 is to set all components to 0. The extension of a label $L_i = (U_i, T_i, C_i, (N_i^v)_{v \in V})$ along an arc $(i, j) \in A$ is performed as the following REFs:

$$U_j = U_i + \overline{u}_{ij}^k, \tag{25}$$

$$T_j = T_i + t_{ij}, \tag{26}$$

$$C_j = C_i + c_{ij}, \tag{27}$$

$$N_j^v = \begin{cases} N_i^v + 1, & \text{if } j = v, \\ \max\{N_i^v, UR_v(T_j, C_j)\}, & \text{otherwise.} \end{cases} \tag{28}$$

$UR_v(T_j, C_j) = 1$ if city v is unreachable from label L_j, i.e., if at least one of the following conditions holds (assume triangle inequality holds for at least one of the travel times and costs): (i) City v has already been visited, (ii) $T_j + t_{jv} > t_{max}$, (iii) $C_j + c_{jv} > c_{max}$. Given conditions (i), (ii), (iii) for a city to be unreachable from a label, the infeasibility check to reach city v is addressed by $N_j^v = 1$ in the $(N_i^v)_{v \in V}$ vector of the label. Furthermore, we can obtain formula (28) from Gutiérrez-Jarpa et al. (2010) and explain it as follows:

- If $j = v$, since we extend the subpath ending at node i by adding arc (i, j), it means that $j = v$ was not in that path so we had $N_i^v = 0$, and hence $N_j^v = N_i^v + 1 = 1$.
- If $j \neq v$, there are two cases: (a) if v was not visited in the path to i and v is reachable. From j, $N_i^v = 0$ and $UR_v(T_j, C_j) = 0$, then $N_j^v = \max(N_i^v, UR_v(T_j, C_j)) = 0$. (b) if v was visited in the path to i or v is not reachable from j, then $N_i^v = 1$ or $UR_v(T_j, C_j) = 1$. Consequently, $N_j^v = \max(N_i^v, UR_v(T_j, C_j)) = 1$. In any case, we indeed find $N_j^v = \max(N_i^v, UR_v(T_j, C_j))$.

This path is feasible if all the following conditions hold:

$$T_j = T_i + t_{ij} \leq t_{max}, \tag{29}$$

$$C_j = C_i + c_{ij} \leq c_{max}, \tag{30}$$

$$N_j^v \leq 1, v \in V \tag{31}$$

In order to avoid enumerating all feasible paths/tours, given that all the REFs are nondecreasing functions, a dominance rule can apply to discard those unpromising labels.

Definition 2. (*Dominance rule*) *Let* $L_i^1 = (U_i^1, T_i^1, C_i^1, (N_i^{1,v})_{v\in V})$ *and* $L_i^2 = (U_i^2, T_i^2, C_i^2, (N_i^{2,v})_{v\in V})$ *represent two labels associated with different tours ending at the same city* i*. Then label* L_i^1 *is said to dominate* L_i^2 *if* $U_i^1 \geq U_i^2, T_i^1 \leq T_i^2, C_i^1 \leq C_i^2$ *and* $(N_i^{1,v})_{v\in V} \leq (N_i^{2,v})_{v\in V}$ *holds, and at least one of them is strict.*

4.4. Acceleration strategies for the pricing problem

We use two strategies to accelerate the pricing problem solution. The first strategy is to relax the subproblem by allowing paths containing cycles. Several relaxations relying on this principle have been developed in VRP studies (Desaulniers et al., 2014, 2016), and the *ng*-route relaxation proposed in Baldacci et al. (2011) was proven the most effective.

An *ng*-route is a route that may contain cycles if they satisfy some conditions. More precisely, we define a neighborhood $NG_i \subset V$ that contains node i and its ξ closest nodes in V, where ξ is a predefined parameter on the neighborhood size. An *ng*-route is allowed to visit a node i twice if it visits at least one node j in between two visits to i and $i \notin NG_j$. Using this route relaxation, the subproblem becomes a shortest *ng*-path problem with resource constraints (*ng*-SPPRC) which can be solved by the labeling algorithm. On the one hand, if ξ is small, the subproblem becomes easier to solve but with a weaker bound; on the other hand, a larger size ξ yields better bounds but with intensively computational efforts.

To incorporate the *ng*-route into the labeling algorithm, the resource function $(N_i^v)_{v\in V}$ in a label should be processed. For a subpath $r = (0, j_1, ..., j_q)$ and $\forall l = 0, 1, ..., q$, we know that $N_{j_l}^v = 1$ if v belongs to path r (that is, $v = j_l$ for $l \in \{1, ..., q\}$) and $v \in NG_{j_{l'}}$ for all $l' \in \{l, ..., q\}$, and 0 otherwise. On the contrary, for a node v, $N_j^v = 0$ holds if v has not been visited or if it does not belong to the neighborhood of a node visited after its last visit. To perform the extension along arc (i, j) through the REFs, for each node $v \in V$, we set

$$N_j^v = \begin{cases} 1, & \text{if } j = v \text{ or if } N_i^v = 1 \text{ and } v \in NG_j, \\ 0, & \text{otherwise.} \end{cases} \tag{32}$$

In the dynamic programming algorithm to solve the ESPPRC (Feillet et al., 2004), the state of a subpath associated with a label L on a node includes the triplet (U, T, C) and the full vector of unreachable nodes. Therefore, there exist at worst case $O(2^{|\overline{V}|})$ labels on a node. However, by replacing the unreachable vector with the forbidden set in the *ng*-route, the label space is significantly

reduced. As for each pair of (U, T, C), the number of N^v equal to 1 is less than or equal to ξ, the label space is reduced to less than $O(2^\xi)$ (Gianessi et al., 2016).

The second acceleration strategy is to rapidly generate columns with positive reduced utility using a heuristic-based labeling algorithm. More precisely, the various labels L_i at node i are sorted first by decreasing reduced utility, then by increasing time and cost consumption (Righini & Salani, 2009). Based on this efficient list structure, we propose a relaxed dominance condition which is to introduce a redundant label component $\phi_i = \sum_{v \in V} N_i^v$ in L_i. The dominance check can be sped up by modulating between weak and strong dominance levels. At the weak level, we keep the first three conditions but replace the fourth one with $\phi_i^1 \leq \phi_i^2$, until no column with positive reduced utility can be generated. Then we restore the fourth condition to its strong version $(N_1^v)_{v \in V} \leq (N_2^v)_{v \in V}$ and continue the labeling algorithm until no positive column can be found. The weak dominance check enables to solve the subproblem faster in the first iterations.

4.5. Warm-up and lower bound

The pool of initial columns is generated by a primal-heuristic (PH). We enumerate all feasible tours which visit one node and two nodes respectively, i.e., we generate the sets:

$$\begin{aligned} \mathcal{R}^1 &= \{(0, i, n) \mid c_{0i} + c_{in} \leq c_{max}, t_{0i} + t_{in} \leq t_{max}, \forall i \in \overline{V}\} \\ \mathcal{R}^2 &= \{(0, i, j, n) \mid c_{0i} + b_{ij} c_{ij} + c_{jn} \leq c_{max}, t_{0i} + b_{ij} t_{ij} + t_{jn} \leq t_{max}, \forall (i, j) \in A\} \end{aligned}$$

By incorporating $\mathcal{R}^1$ and $\mathcal{R}^2$ as the initial pool of columns, the dual variables help to find the most promising node pairs in the future columns.

At the end of the column generation phase, when no positive reduced utility column is found, the value of the RMP is the value of $\overline{\text{MP}}$. We run a MIP solver on the subset of columns of the last RMP. This MIP-based heuristic (MIP-H) provides a lower bound for integer solutions and helps pruning unpromising nodes in the branch-and-bound tree.

4.6. Branching scheme

If the solution of $\overline{\text{MP}}$ obtained from the column generation phase is fractional and the corresponding dual bound is not below any known lower bound, the associated node in the search tree cannot be pruned, and branching occurs. The branching strategy we choose for TOP-DC is to branch on nodes at first, then on arcs, as in Boussier et al. (2007). When the solution is fractional, we first branch on a node $i \in \overline{V}$ that is visited a fractional number of times in a tour k ($0 < \sum_{r \in \hat{\mathcal{R}}} a_i^r \lambda_{rk} < 1$), and has the largest reduced utility $\sum_j \overline{u}_{ij}^k b_{ij}^r \lambda_{rk}$. Two branches are created by updating constraints (19) in the MP as follows:

- In the enforcement branch, node i must be visited in tour k, that is, $\sum_{r\in\mathcal{R}} a_i^r \lambda_{rk} = 1$.
- In the forbidden branch, node i is not allowed to be visited in tour k, that is, $\sum_{r\in\mathcal{R}} a_i^r \lambda_{rk} = 0$. Moreover, we need to remove node i in G^k, such that node i will never be visited in the future columns.

When the flow traversing each node is integer, the branching is then applied to an arc (i, j) with a fractional value in selected tour k. If the arc flow $\sum_{r\in\hat{\mathcal{R}}} b_{ij}^r \lambda_{rk}$ is fractional for several arcs (i, j), we choose an arc whose fractional part of the flow is the closest to 0.5. For the selected arc (i, j) in tour k, we consider two cases as in Boussier et al. (2007), depending on whether i or j is served or not. For more details on selected arc (i, j) in G^k, we refer to Desrosiers & Lübbecke (2005).

The branch-and-bound search tree is explored with a best-first search strategy. A node is evaluated at each subtree, only the node whose dual bound is greater than the current lower bound is added to the search-tree for future exploration. The current lower bound can be obtained by either the MIP-H heuristic, or the integer solution found in the previous subtree exploration. This process is repeated until completing the exploration of the search tree and getting the final optimal integer solution.

The above branch-and-price method based on the two-index reformulation provides a tractable way to solve the TOP-DC problem. However, for most of the instances, we encounter a symmetry problem which induces redundant nodes in the search tree and increases the pruning burden. Indeed with the two-index variables λ_{rk}, given two columns $r = 1, 2$ and tour indices $k = 3, 4$, switching the pair of variables $(\lambda_{13}, \lambda_{24})$ to $(\lambda_{23}, \lambda_{14})$ provides exactly the same solution and same value of $(z_i^{34})_{i\in\overline{V}}$. To avoid the branching symmetry problem, we further provide a one-index reformulation and a Branch-Cut-and-Price method which is shown to outperform the above BP method a majority of cases.

5. Branch-Cut-and-Price method

Branch-Cut-and-Price (BCP) combines branch and bound, column generation and cutting plane algorithms for solving mixed integer programs. Compared to a pure BP, adding cut generation in BCP can yield a stronger LP relaxation. However, the pricing problem may become much harder because the new added rows can destroy the structure of the pricing problem (Barnhart et al., 1998).

5.1. One-index master reformulation

Given a subset of nodes $S \subseteq \overline{V}$, let $a_S^r = 1$ if all nodes of S are visited in tour $r \in \mathcal{R}$, and $a_S^r = 0$ otherwise. When $S = \{i\}$ for some $i \in \overline{V}$, it means as before whether node i is visited or

not. The binary variables associated to the selection of a feasible tour or column are not indexed by k anymore:

$$\theta_r = \begin{cases} 1, & \text{if column } r \text{ is selected in the solution,} \\ 0, & \text{otherwise.} \end{cases}$$

And the new one-index master reformulation is:

$$\max \sum_{r \in \mathcal{R}} p_r \theta_r \tag{33}$$

subject to

$$\sum_{r \in \mathcal{R}} \theta_r = m \tag{34}$$

$$\sum_{r \in \mathcal{R}} a_S^r \theta_r \leq 1, \forall S \subseteq \overline{V} : |S| \geq s+1 \tag{35}$$

$$\theta_r \in \{0,1\}, \forall r \in \mathcal{R} \tag{36}$$

The objective function (33) maximizes the sum of the utilities of the selected tours as before. Constraints (34) ensures that m tours will be selected in the final solution. Constraints (35) are used to cut-off those solutions violating the diversity conditions. Because all potential columns cannot be generated at once, constraints (35) are generated on the fly from incompatible columns in the RMP, and we denote by $\hat{\mathcal{S}}$ the collection of common-node sets S generated up to the current iteration of column generation.

The column generation procedure in the BCP is similar to that in the BP. However, due to the existence of cut generation constraints (35), the number of dual variables are increasing such that the structure of the pricing problem is changed at each iteration. As a result, the algorithm to solve the new pricing problem needs to be adapted in order to find columns with positive reduced utility.

5.2. Pricing problem

Let π and η_S be the dual variables associated to constraints (34) and (35). The reduced utility of a column $r \in \mathcal{R}$ for the new BCP pricing problem is:

$$\overline{c}_r = p_r - \pi - \sum_{S \in \hat{\mathcal{S}}} a_S^r \eta_S \tag{37}$$

The pricing problem for BCP is also solved by the ESPPRC algorithm but with a modified dominance rule due to constraints (35). Indeed, the previous dominance rule would delete some promising labels and prevent the generation of optimal columns, as shown in the following example.

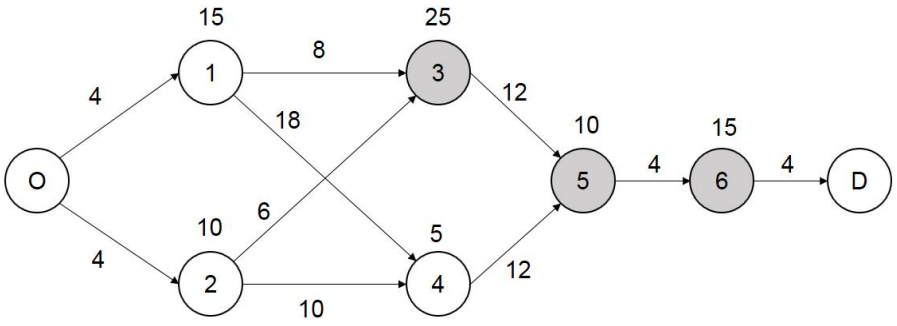


Figure 1: An example of delayed dominance check

Example. In Figure 1, the values on nodes and arcs represent their respective utility and travel time. Suppose $t_{max} = 50$ and $s = 2$. The ESPPRC algorithm on this graph, after extending labels to node 5, generates the following labels: $L_5^1 = (50, 24), L_5^2 = (30, 34), L_5^3 = (45, 22), L_5^4 = (25, 26)$, associated with subpaths $r_1 = (0, 1, 3, 5), r_2 = (0, 1, 4, 5), r_3 = (0, 2, 3, 5), r_4 = (0, 2, 4, 5)$. Obviously, $L_5^1(L_5^3)$ dominates $L_5^2(L_5^4)$. Finally, two complete paths $r_1 = (0, 1, 3, 5, 6, n)$ and $r_3 = (0, 2, 3, 5, 6, n)$ would be added to the column pool. Assume these two paths are selected in the final solution with values $\theta_1 = 1$ and $\theta_3 = 1$. However, this solution violates the diversity constraints since they share 3 common nodes $S = \{3, 5, 6\}$. According to our cut-generation constraints, a new constraint $\theta_1 + \theta_3 \leq 1$ would be added to the $\overline{\text{MP}}$ and the model is resolved immediately. Suppose the dual value to the new cut (constraint) is $\eta_{\{3,5,6\}} = -27$, which is given to the pricing problem for the next iteration of column generation. In the new iteration of ESSPRC, the previous dominance rule cannot be applied to node 5 any more, as it would delay the dominance check. That is, assuming the above four labels are kept at node 5 and extended to node 6 now, the new labels in node 6 are: $L_6^1 = (65 - 27, 28) = (38, 28), L_6^2 = (45, 38), L_6^3 = (60 - 27, 26) = (33, 26), L_6^4 = (40, 30)$ with associated subpaths $r'_1 = (0, 1, 3, 5, 6), r'_2 = (0, 1, 4, 5, 6), r'_3 = (0, 2, 3, 5, 6), r'_4 = (0, 2, 4, 5, 6)$. Thus, labels L_6^2 and L_6^4 are not dominated any more and will be preserved in the final column pool. However, if we apply the traditional dominance rule at node 5, we are not able to get these two labels at the end of the dynamic programming procedure because they are removed earlier. Thus, to implement the ESPPRC algorithm in BCP, we propose a new delayed dominance rule as follows. We add a new vector and scalar to a label: $L_i = (U_i, T_i, C_i, (N_i^v)_{v \in V}, (a_S^{r_i})_{S \in \hat{\mathcal{S}}}, H_i)$, where r_i is the related path to label L_i and the last two items in L_i indicate:

- $a_S^{r_i} = 1$ if subset $S \in \hat{\mathcal{S}}$ has already been visited in the path r_i, or cannot be visited because of the unreachable property; otherwise, $a_S^{r_i} = 0$.
- $H_i = 1$ if $a_S^{r_i} = 1, \forall S \in \hat{\mathcal{S}}$; otherwise, $H_i = 0$.

If the label L_i is extending from city i to city j, such that L_j is the new label and r_j is related path. Considering the two additional resources, the corresponding REFs are described as

$$U_j = U_i + \overline{u}_{ij}^k - \sum_{S\in\hat{\mathcal{S}}} a_S^{r_j}\eta_S, \tag{38}$$

$$T_j = T_i + t_{ij}, \tag{39}$$

$$C_j = C_i + c_{ij}, \tag{40}$$

$$N_j^v = \begin{cases} N_i^v + 1, & \text{if } j = v, \\ \max\{N_i^v, UR_v(T_j, C_j)\}, & \text{otherwise.} \end{cases} \tag{41}$$

$$a_S^{r_j} = \begin{cases} a_S^{r_i} + 1, & \text{if subset } S \in \hat{\mathcal{S}} \text{ is visited in the path } r_j, \\ \max\{a_S^{r_i}, UR_S(T_j, C_j)\}, & \text{otherwise.} \end{cases} \tag{42}$$

We redefine $UR_S(T_j, C_j) = 1$ if any city $v \in S$ is unreachable from label L_j, i.e., either $T_j + t_{jv} > t_{max}$ or $C_j + c_{jv} > c_{max}$ holds. Given vector $(a_S^{r_j})_{S\in\hat{\mathcal{S}}}$, we can check whether $H_j = 1$ holds during the label extension. This path L_j is feasible if all the REFs hold as (29), (30) and (31), and the new additional resources should satisfy $(a_S^{r_j})_{S\in\hat{\mathcal{S}}} \leq 1$ and $H_j \leq 1$ respectively.

Definition 3. (*Delayed dominance rule*) *Let* $L_i^1 = (U_i^1, T_i^1, C_i^1, (N_i^{1,v})_{v\in V}, (a_S^{r_i^1})_{S\in\hat{\mathcal{S}}}, H_i^1)$ *and* $L_i^2 = (U_i^2, T_i^2, C_i^2, (N_i^{2,v})_{v\in V}, (a_S^{r_i^2})_{S\in\hat{\mathcal{S}}}, H_i^2)$ *represent two labels associated with different tours (i.e.,* r_i^1 *and* r_i^2*) ending at the same city* i*. Label* L_i^1 *is said to dominate* L_i^2 *if and only if both* $H_i^1 = 1$ *and* $H_i^2 = 1$*, and* $U_i^1 \geq U_i^2, T_i^1 \leq T_i^2, C_i^1 \leq C_i^2$ *and* $(N_i^{1,v})_{v\in V} \leq (N_i^{2,v})_{v\in V}$ *holds, and at least one of them is strict.*

The acceleration strategy for the BCP pricing uses the *ng*-routes and the strong-level dominance check presented in Section 4 .

5.3. Cutting planes

To process a node of the search tree, the column and row generation are implemented in a loop. Each loop starts with column generation, solving a RMP with the Simplex method to optimality, characterized by a set of branching constraints and a set of valid inequalities. Then if the solution violates the diversity constraints, cutting plane occurs. When a cut (indeed, a row) is generated, it is added to the last RMP of the current node and the new RMP is solved with the Simplex method again. Otherwise, if all diversity violation constraints are added, the algorithm exits from the loop. If the solution is integer, then we have found an optimal integer solution, otherwise, either the node is pruned by the lower bound, or branching occurs.

Separation algorithm. Note that, since we work on an extended formulation, we do not need a complex separation algorithm on the space of the original arc variables. Cuts are separated as follows. From the last solution given by column generation, we identify the set $\hat{\mathcal{R}}^+ \subset \hat{\mathcal{R}}$ of variables with $\theta_r > 0$. For any two columns $r, r' \in \hat{\mathcal{R}}^+$, we compute $S^*_{r,r'}$ the largest set of common nodes between r and r'. Then we add the set of cuts:

$$\theta_r + \theta_{r'} \leq 1, \qquad \forall r, r' \in \hat{\mathcal{R}}^+ : S^*_{r,r'} \notin \hat{\mathcal{S}}, |S^*_{r,r'}| \geq s+1 \tag{43}$$

If $S^*_{r,r'} \in \hat{\mathcal{S}}$ for some pair $r, r' \in \hat{\mathcal{R}}^+$, i.e. the common-node set has already been generated before, and $|S^*_{r,r'}| \geq s+1$, then for this $S = S^*_{r,r'}$ we just need to modify the corresponding constraint as:

$$\sum_{r \in \hat{\mathcal{R}}} a^r_S \theta_r \leq 1 \tag{44}$$

The iterative row generation stops when the solution satisfies the diversity constraints.

5.4. Branching scheme

In the search tree, branching occurs when the master problem is solved at optimality and the corresponding solution of the arc-flow formulation is not integer. We have implemented a branching scheme consisting of three hierarchical levels of additional constraints in the master problem and the pricing problem.

1. For a node $i \in \overline{V}$, if $\sum_{r \in \mathcal{R}} a^r_i \theta_r = \Psi_i$ with Ψ_i fractional, then we create two child nodes: one with the constraint $\sum_{r \in \mathcal{R}} a^r_i \theta_r \geq \lceil \Psi_i \rceil$, the other one with $\sum_{r \in \mathcal{R}} a^r_i \theta_r \leq \lfloor \Psi_i \rfloor$. Note that incorporating the dual variable β_i associated with each additional constraint to the pricing problem (37) does not change its structure.

2. If $\forall i \in \overline{V}, \sum_{r \in \mathcal{R}} a^r_i \theta_r \in \mathbb{N}$ holds, but there exists an arc $(i,j) \in A$ such that $\sum_{r \in \mathcal{R}} b^r_{ij} \theta_r = \Phi_{ij}$ with Φ_{ij} fractional, then two child nodes are created: one with constraint $\sum_{r \in \mathcal{R}} b^r_{ij} \theta_r \geq \lceil \Phi_{ij} \rceil$, the other one with $\sum_{r \in \mathcal{R}} b^r_{ij} \theta_r \leq \lfloor \Phi_{ij} \rfloor$. Again, adding the dual values δ_{ij} associated with these arc-traversing constraints does not affect the structure of the subproblem (37).

3. Even if the above node-visiting and arc-flow variables are integer, it does not ensure optimal solutions to be integer. For example, assume $m = 2$ and $s = 2$, and we got a solution of optimal tours given by $r_1 = (0,1,3,4,n), r_2 = (0,2,3,4,n), r_3 = (0,1,3,5,n), r_4 = (0,2,3,5,n)$. If the path flow for each tour is 0.5, the arc-flow variables are all integer, however, the solution is still fractional. Therefore, if none of conditions 1. and 2. holds, it is necessary to check whether there are two arcs $(i,j) \in A$ and $(j,l) \in A$ traversed consecutively for a fractional

number of times: $\sum_{r\in\mathcal{R}} h^r_{ijl}\theta_r = \Lambda_{ijl}$, where $h^r_{ijl} = 1$ if arc (j,l) is visited immediately after arc (i,j) in tour r. If Λ_{ijl} is fractional, two branches are added in the master problem: one with $\sum_{r\in\mathcal{R}} h^r_{ijl}\theta_r \geq \lceil \Lambda_{ijl} \rceil$ on the first child node, the other one with $\sum_{r\in\mathcal{R}} h^r_{ijl}\theta_r \leq \lfloor \Lambda_{ijl} \rfloor$. We note ν_{ijl} the dual variables associated to these constraints, which are to be given again to the pricing subproblem. The labeling algorithm should be modified then by adding one additional resource to indicate the consecutive arcs' extension status. The REF for such additional resource is similar to the manipulation of generating cuts (i.e., $(a^{r_i}_S)_{S\in\hat{\mathcal{S}}}$ and H_i) as we mentioned before. That is, we introduce a vector and a scalar: the vector is used to check whether the consecutive two-arcs with positive dual values are visited in the extending label. If yes, the corresponding dual value is deducted from the collected utility. The scalar indicates whether all identified two-arcs have been checked such that this label is ready for dominance check. However, this triplet branching rules rarely happen. For more details, we refer to the work Salani & Vacca (2011).

With the above branchings, the reduced utility of a column becomes:

$$\overline{c}_r = p_r - \pi - \sum_{S\in\hat{\mathcal{S}}} a^r_S \eta_S - \sum_{i\in\overline{V}} a^r_i \beta_i - \sum_{(i,j)\in A} b^r_{ij}\delta_{ij} - \sum_{(i,j)\in A,(j,l)\in A} h^r_{ijl}\nu_{ijl} \tag{45}$$

The procedure to implement the whole branch-cut-and-price algorithm is summarized in the Figure 2. We now give the numerical results of the above solving strategies and managerial insights.

6. Computational experiments

We first use a benchmark dataset from the literature to test the performance of our BP and BCP methods. Then, we conduct experiments on a real-case study with data provided by a Chinese OTA (niding.net) based at Beijing who offers global travel products. Beyond the computational aspects, we analyze the obtained solutions and derive managerial insights. In particular, we analyze the trade-off between high diversity, which provides a richer offer for customers, and low diversity, which ensures to select less suppliers with better prices. Our implementation has been coded in C++ on Linux running on an Intel Core i7 with 64 GB RAM, with a time limit of 3600s. All algorithms use IBM ILOG Cplex 12.9 as a LP-solver in multi-threads (8 threads).

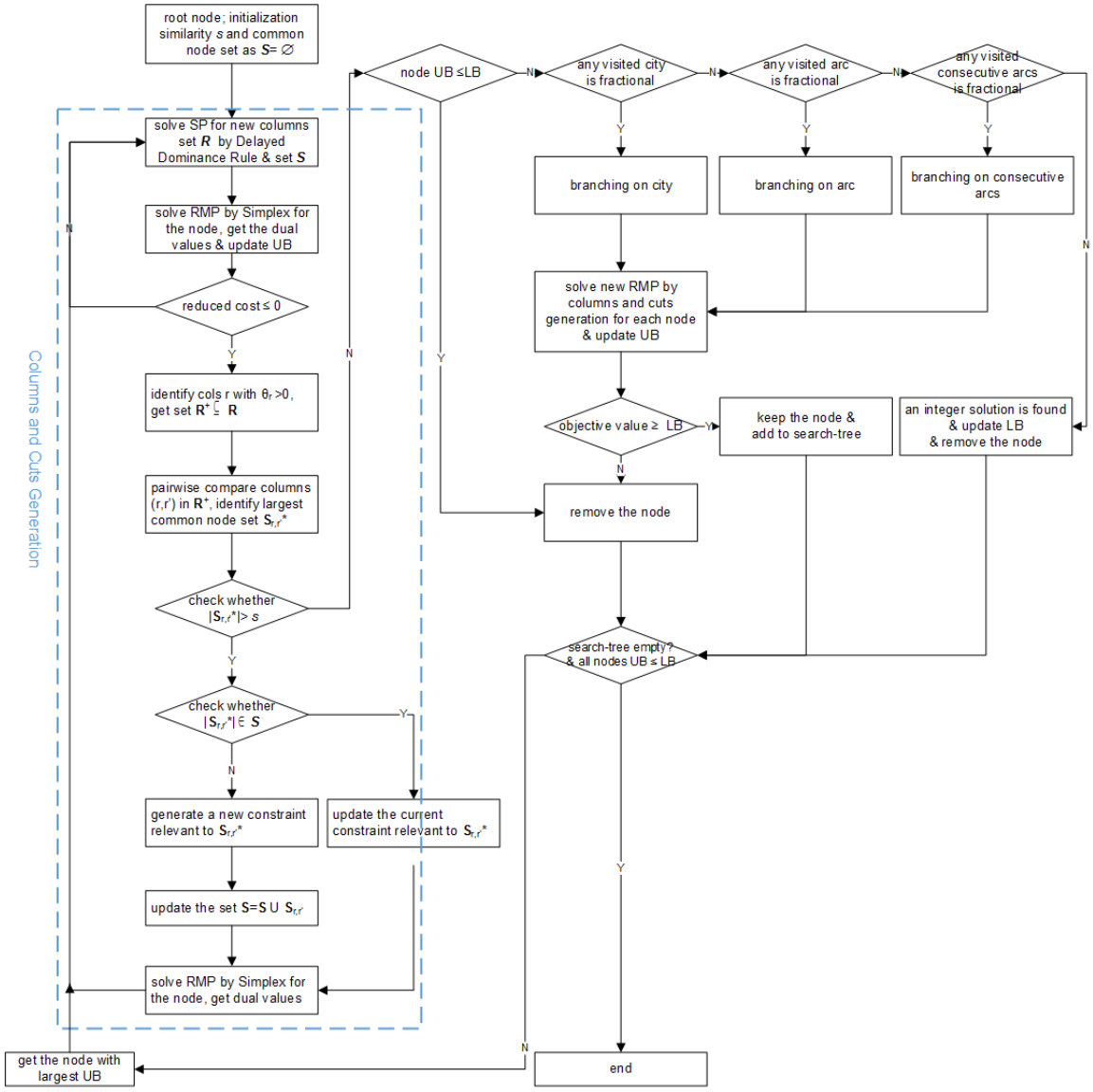


Figure 2: The procedure to implement a branch-cut-and-price algorithm

6.1. Benchmark dataset tests

Dataset description.

The benchmark dataset we used are the TOP instances tested in Chao et al. (1996b), Boussier et al. (2007) and Keshtkaran et al. (2016). When $s \geq 1$, there are no benchmark results as this problem has not been solved before. To assess the performance of our algorithms, we test instances with 21 nodes and 66 nodes from the dataset except those without feasible solutions (which gives 46 instances for 66 nodes out of the initial 50 instances). Each dataset with given number of nodes differs in the resource limitation t_{max} and the number of tours selected m. Each node has

known coordinates and score. We use only the time resource for the TOP-DC, and the travel times/costs are the Euclidean distances calculated from node coordinates. The experimental results are summarized in Tables 1 to 6. In the tables, columns *BP*, *BCP* and *CF* provide the results obtained from Branch-and-Price, Branch-Cut-and-Price, and the compact formulation solved by CPLEX; subcolumns *val* (i.e., optimal or best-known value), *optgap* (i.e., (upperbound - optimal or best-known value)/upperbound), *rootgap* (i.e., (root node LP relaxation value - optimal or best-known value)/root node LP relaxation value), *nodes* (i.e., number of branching nodes) and *time* (i.e., computational time in seconds) for each instance. We also provide the *Tour Details: #cities* in the optimal/best-known solutions, including the *total* (i.e., total number of cities visited over all tours), *average, max, min* (i.e., average, maximum and minimum number of cities per tour). Because of the pages limitation, we only report the *rootgap* and *Tour Details: #cities* for scenarios $s = 1$, but we observe similar phenomena for general value of s. The mark '-' indicates we cannot provide a gap since no upper bound was found.

Numerical results and analysis.

First of all, after comparison the *optgaps* and the *rootgaps* in the Tables 1 to 6, we can draw several interesting conclusions: the *optgaps* differences show that our BP and BCP algorithms outperform CPLEX solver for the compact formulation, and give optimal solutions for most of the instances within a reasonable time (also note that our methods could also solve the classical TOP ($s = 0$) with reasonable times, without competing with up-to-date TOP methods as our methods are tailored for the specific TOP-DC). Moreover, the comparison of *rootgaps* demonstrates that our two-index extended formulation and one-index reformulation provide tighter initial LP relaxation values than the (arc-flow) compact formulation, accelerating the convergence to optimality of the designed algorithms.

Secondly and not surprisingly, the smaller s, the harder the problem is to solve, as a stricter diversity requirement is to be satisfied. Our numerical experiments also showed that BCP clearly outperforms BP for the TOP-DC for the larger graphs. For example, among all 46 instances in the scenario $n = 66$ nodes, the average computing time of BCP is faster than the BP (46 vs. 129 seconds), and the former could solve more instances to optimal (99.2% vs. 97.5%). Also, the root gap is significantly better for BCP for $n = 66$. As we can observe, the larger s, the less pairs of tours violating the diversity constraints, and less cuts are generated, which favors BCP.

Thirdly, we notice that the computational time increases with m and n for a given s, as more iterations are needed to solve the subproblems by dynamic programming. We also observe that the subproblems in the two-index formulations are solved more fastly but with a less tight bound, while that in the one-index formulation are slower but with a tighter bound. Therefore, the two methods

are somewhat complementary.

Table 1: Computational results for TOP-DC instances with nodes $n = 21$ and similarity parameter $s = 1$

			BP					*BCP*					*CF*					*Tour Details:#cities*			
Instance	t_{max}	m	*val*	*optgap*	*rootgap*	*nodes*	*time*	*val*	*optgap*	*rootgap*	*nodes*	*time*	*val*	*optgap*	*rootgap*	*nodes*	*time*	*total*	*average*	*max*	*min*
p2.2.a	7.5	2	105	0	0	0	0	105	0	6.2%	5	0	105	0	14.6%	0	0	8	5	6	5
p2.2.b	10		130	0	0	0	0	130	0	13.3%	33	1	130	0	31.5%	0	0	10	6	7	6
p2.2.c	11.5		165	0	0	0	0	165	0	4.1%	11	1	165	0	24.3%	0	0	11	7	8	6
p2.2.d	12.5		175	0	0	0	0	175	0	5.4%	37	10	175	0	24.2%	84	0	12	7	8	7
p2.2.e	13.5		195	0	0	7	0	195	0	5.3%	63	16	195	0	20.4%	3562	2	13	8	9	7
p2.2.f	15		225	0	0	0	0	225	0	6.3%	263	157	225	0	25.5%	260	1	14	8	9	8
p2.2.g	16		225	0	0	0	0	225	-	18.2%	381	3600	225	0	32.0%	59	1	14	8	10	7
p2.2.h	17.5		250	0	0	0	7	230	-	28.6%	1432	3600	250	0	31.3%	3498	3	15	9	11	7
p2.2.i	19		255	0	0	0	1	230	-	37.0%	108	3600	255	0	35.8%	1351	2	15	9	12	6
p2.2.j	20		280	0	0	3	0	230	-	39.8%	102	3600	280	0	33.5%	5936	12	16	9	10	9
p2.2.k	22.5		290	0	3.3%	75	14	260	-	32.4%	43	3600	290	0	38.6%	12655	24	16	9	10	9
p2.3.a	5	3	95	0	9.5%	13	0	95	0	0	0	0	95	0	13.6%	0	0	6	4	5	3
p2.3.b	6.7		120	0	15.5%	37	0	120	0	7.7%	11	0	120	0	15.5%	0	0	7	4	5	4
p2.3.c	7.7		150	0	12.8%	61	0	150	0	6.3%	15	0	150	0	23.1%	0	0	8	5	7	4
p2.3.d	8.3		150	0	14.8%	179	0	150	0	8.5%	25	0	150	0	30.9%	0	0	8	5	6	4
p2.3.e	9		155	0	17.1%	543	1	155	0	17.6%	51	0	155	0	28.6%	1524	0	10	5	6	5
p2.3.f	10		170	0	12.8%	179	0	170	0	15.8%	81	3	170	0	40.4%	3506	2	10	5	7	4
p2.3.g	10.7		175	0	13.4%	1033	3	175	0	27.1%	297	33	175	0	42.6%	8592	7	11	5	8	4
p2.3.h	11.7		215	0	13.0%	109	0	215	0	12.6%	1007	506	215	0	35.1%	5236	8	12	6	8	5
p2.3.i	12.7		240	0	8.4%	129	1	240	0	14.9%	3265	2101	240	0	31.8%	8737	12	13	6	7	6
p2.3.j	13.3		255	0	12.7%	513	6	250	-	16.7%	2979	3600	255	0	30.0%	16070	38	14	7	7	7
p2.3.k	15		265	0	21.4%	3913	45	200	-	44.0%	711	3600	265	0	41.6%	12589	52	14	7	8	5
p2.4.a	3.8	4	40	0	0	0	0	40	0	0	0	0	40	0	0	0	0	3	3	3	3
p2.4.b	5		120	0	14.3%	61	0	120	0	0	0	0	120	0	18.4%	0	0	6	3	5	3
p2.4.c	5.8		135	0	0	105	0	135	0	0	0	0	135	0	26.6%	16	0	6	4	5	3
p2.4.d	6.2		155	0	8.8%	53	0	155	0	0	0	0	155	0	18.4%	65	0	7	4	6	4
p2.4.e	6.8		155	0	18.4%	225	0	155	0	3.7%	13	0	155	0	18.4%	56	0	7	4	5	4
p2.4.f	7.5		170	0	19.1%	1571	3	170	0	4.0%	41	0	170	0	30.9%	800	0	8	4	6	4
p2.4.g	8		185	0	20.3%	1357	3	185	0	5.1%	45	1	185	0	36.2%	950	0	8	4	6	4
p2.4.h	8.8		195	0	22.0%	3639	12	195	0	6.3%	65	0	195	0	32.7%	4594	2	9	5	6	5
p2.4.i	9.5		205	0	21.2%	4251	17	205	0	6.8%	343	4	205	0	43.4%	7928	10	10	5	6	5
p2.4.j	10		210	0	19.2%	3933	16	210	0	7.9%	183	5	210	0	44.7%	57460	41	10	5	7	5
p2.4.k	11.2		250	0	16.7%	1605	10	250	0	11.4%	79	33	250	0	41.3%	113906	147	12	6	7	5

Table 2: Computational results for TOP-DC instances with nodes $n = 21$ and similarity parameter $s = 2, 3$

			s=2												*s=3*											
			BP				*BCP*				*CF*				*BP*				*BCP*				*CF*			
Instance	t_{max}	m	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*
p2.2.a	7.5	2	110	0	0	0	110	0	5	0	110	0	0	0	120	0	0	0	120	0	0	0	120	0	0	0
p2.2.b	10		140	0	5	0	140	0	11	0	140	0	938	0	150	0	0	0	150	0	3	0	150	0	0	1
p2.2.c	11.5		170	0	0	0	170	0	7	0	170	0	679	1	170	0	23	0	170	0	7	0	170	0	0	1
p2.2.d	12.5		180	0	7	0	180	0	11	2	180	0	3969	3	190	0	0	0	190	0	0	0	190	0	2540	2
p2.2.e	13.5		200	0	9	0	200	0	53	6	200	0	793	2	210	0	0	0	210	0	0	0	210	0	2482	2
p2.2.f	15		240	0	0	0	240	0	5	1	240	0	2586	3	240	0	0	0	240	0	0	0	240	0	1215	2
p2.2.g	16		245	0	0	0	245	0	833	537	245	0	316	2	265	0	0	0	265	0	143	49	265	0	1953	3
p2.2.h	17.5		260	0	14	0	230	-	2042	3600	260	0	4840	14	270	0	49	3	265	-	4425	3600	270	0	8020	21
p2.2.i	19		275	0	0	0	230	-	184	3600	275	0	2862	5	290	0	0	0	230	-	580	3600	290	0	1644	3
p2.2.j	20		285	0	29	2	230	-	116	3600	285	0	8347	25	295	0	65	7	230	-	172	3600	295	0	2689	5
p2.2.k	22.5		310	0	27	5	260	-	47	3600	310	0	18106	58	325	0	9	4	260	-	42	3600	325	0	6108	9
p2.3.a	5	3	105	0	0	0	105	0	0	0	105	0	0	0	105	0	0	0	105	0	0	0	105	0	0	0
p2.3.b	6.7		145	0	47	0	145	0	5	0	145	0	0	0	160	0	23	0	160	0	0	0	160	0	0	0
p2.3.c	7.7		175	0	59	0	175	0	5	0	175	0	0	0	190	0	13	0	190	0	0	0	190	0	0	0
p2.3.d	8.3		185	0	31	0	185	0	7	0	185	0	249	0	210	0	5	0	210	0	0	0	210	0	0	0
p2.3.e	9		185	0	145	0	185	0	39	0	185	0	1017	1	210	0	0	0	210	0	0	0	210	0	0	0
p2.3.f	10		195	0	335	2	195	0	45	0	195	0	6070	8	220	0	17	0	220	0	5	0	220	0	7388	4
p2.3.g	10.7		210	0	49	1	210	0	123	6	210	0	19040	20	225	0	89	2	225	0	63	1	225	0	24138	22
p2.3.h	11.7		235	0	233	2	235	0	599	62	235	0	5299	14	250	0	131	3	250	0	85	0	250	0	12264	17
p2.3.i	12.7		260	0	341	2	260	0	1163	286	260	0	20235	35	285	0	17	1	285	0	0	0	285	0	4973	7
p2.3.j	13.3		275	0	833	3	275	0	1797	461	275	0	17103	45	300	0	59	1	300	0	209	31	300	0	12457	22
p2.3.k	15		315	0	1597	54	310		7535	3600	315	0	20387	111	345	0	327	22	345	0	741	294	345	0	15778	38
p2.4.a	3.8	4	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0
p2.4.b	5		140	0	0	0	140	0	0	0	140	0	0	0	140	0	0	0	140	0	0	0	140	0	0	0
p2.4.c	5.8		160	0	0	0	160	0	0	0	160	0	0	0	160	0	0	0	160	0	0	0	160	0	0	0
p2.4.d	6.2		190	0	21	0	190	0	0	0	190	0	0	0	200	0	0	0	200	0	0	0	200	0	0	0
p2.4.e	6.8		190	0	313	2	190	0	5	0	190	0	0	0	210	0	117	0	210	0	0	0	210	0	0	0
p2.4.f	7.5		205	0	199	2	205	0	0	0	205	0	1038	2	225	0	189	1	225	0	0	0	225	0	0	1
p2.4.g	8		240	0	67	0	240	0	0	0	240	0	161	0	250	0	71	0	250	0	0	0	250	0	223	1
p2.4.h	8.8		240	0	527	33	240	0	115	0	240	0	5755	7	270	0	133	8	270	0	0	0	270	0	1325	1
p2.4.i	9.5		245	0	2871	16	245	0	109	1	245	0	16634	21	280	0	245	2	280	0	5	0	280	0	5544	4
p2.4.j	10		250	0	3789	15	250	0	265	2	250	0	22868	33	285	0	371	3	285	0	39	0	285	0	10429	15
p2.4.k	11.2		285	0	1697	10	285	0	1435	58	285	0	255576	813	310	0	807	13	310	0	711	13	310	0	167708	306

Table 3: Computational results for TOP-DC instances with nodes $n = 21$ and similarity parameter $s = 4, 5$

			s=4												*s=5*									
			BP				*BCP*				*CF*				*BP*				*BCP*				*CF*			
Instance	t_{max}	m	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*
p2.2.a	2	7.5	120	0	0	0	120	0	0	0	120	0	0	0	120	0	0	0	120	0	0	0	120	0	0	0
p2.2.b		10	160	0	0	0	160	0	0	0	160	0	0	0	160	0	0	0	160	0	0	0	160	0	0	0
p2.2.c		11.5	180	0	0	0	180	0	0	0	180	0	0	0	180	0	0	0	180	0	0	0	180	0	341	1
p2.2.d		12.5	190	0	13	0	190	0	0	0	190	0	646	1	195	0	5	1	190	0	0	0	195	0	918	1
p2.2.e		13.5	210	0	5	0	210	0	0	0	210	0	1026	1	210	0	9	1	210	0	0	1	210	0	3809	2
p2.2.f		15	240	0	0	0	240	0	0	0	240	0	7246	4	240	0	0	0	240	0	0	0	240	0	3180	3
p2.2.g		16	270	0	9	1	270	0	9	1	270	0	1683	2	280	0	0	1	280	0	0	0	280	0	450	1
p2.2.h		17.5	285	0	39	5	285	0	2239	1726	285	0	1096	3	305	0	0	0	305	0	0	0	305	0	3840	4
p2.2.i		19	305	0	7	1	300	-	1329	3600	305	0	873	3	315	0	19	2	315	-	1869	3600	315	0	891	3
p2.2.j		20	315	0	11	1	260	-	85	3601	315	0	965	3	330	0	9	1	240	-	355	3600	330	0	287	2
p2.2.k		22.5	345	0	0	1	260	-	63	3600	345	0	1056	4	355	0	25	7	260	-	85	3601	355	0	16134	33
p2.3.a	3	5	105	0	0	0	105	0	0	0	105	0	0	0	105	0	0	0	105	0	0	0	105	0	0	0
p2.3.b		6.7	180	0	0	0	180	0	0	0	180	0	0	0	180	0	0	0	180	0	0	0	180	0	0	0
p2.3.c		7.7	190	0	55	0	190	0	0	0	190	0	0	0	210	0	0	0	210	0	0	0	210	0	0	0
p2.3.d		8.3	210	0	0	0	210	0	0	0	210	0	0	0	210	0	0	0	210	0	0	0	210	0	0	0
p2.3.e		9	210	0	0	0	210	0	0	0	210	0	0	0	210	0	0	0	210	0	0	0	210	0	0	0
p2.3.f		10	230	0	21	0	230	0	0	0	230	0	3428	2	240	0	0	0	240	0	0	0	240	0	1280	2
p2.3.g		10.7	245	0	19	1	245	0	0	0	245	0	2888	3	255	0	27	0	255	0	0	0	255	0	2147	2
p2.3.h		11.7	260	0	69	2	260	0	0	0	260	0	14941	16	265	0	63	1	265	0	0	0	265	0	31662	24
p2.3.i		12.7	295	0	9	0	295	0	0	0	295	0	5234	9	295	0	9	0	295	0	0	0	295	0	10366	10
p2.3.j		13.3	310	0	105	3	310	0	0	0	310	0	5487	10	310	0	45	1	310	0	0	0	310	0	8271	10
p2.3.k		15	355	0	119	7	355	0	39	8	355	0	22882	49	360	0	7	1	360	0	0	0	360	0	30460	42
p2.4.a	4	3.8	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0
p2.4.b		5	140	0	0	0	140	0	0	0	140	0	0	0	140	0	0	0	140	0	0	0	140	0	0	0
p2.4.c		5.8	160	0	0	0	160	0	0	0	160	0	0	0	160	0	0	0	160	0	0	0	160	0	0	0
p2.4.d		6.2	200	0	0	0	200	0	0	0	200	0	0	0	200	0	0	0	200	0	0	0	200	0	0	0
p2.4.e		6.8	240	0	0	0	240	0	0	0	240	0	0	0	240	0	0	0	240	0	0	0	240	0	0	0
p2.4.f		7.5	240	0	0	0	240	0	0	0	240	0	0	0	240	0	0	0	240	0	0	0	240	0	0	0
p2.4.g		8	250	0	369	1	250	0	0	0	250	0	0	0	280	0	0	0	280	0	0	0	280	0	0	0
p2.4.h		8.8	280	0	0	0	280	0	0	0	280	0	0	1	280	0	0	0	280	0	0	0	280	0	0	1
p2.4.i		9.5	300	0	205	1	300	0	0	0	300	0	12952	10	320	0	0	0	320	0	0	0	320	0	0	1
p2.4.j		10	300	0	251	2	300	0	0	0	300	0	188308	111	320	0	0	0	320	0	0	0	320	0	18797	13
p2.4.k		11.2	325	0	955	14	325	0	0	0	325	0	136811	125	340	0	283	5	340	0	0	0	340	0	207054	186

Table 4: Computational results for TOP-DC instances with nodes $n = 66$ and similarity parameter $s = 1$

			BP					*BCP*					*CF*					*Tour Details:#cities*			
Instance	t_{max}	m	*val*	*optgap*	*rootgap*	*nodes*	*time*	*val*	*optgap*	*rootgap*	*nodes*	*time*	*val*	*optgap*	*rootgap*	*nodes*	*time*	*total*	*average*	*max*	*min*
p5.2.b	5	2	20	0	0	0	0	20	0	0	0	0	20	0	15.5%	0	0	6	4	4	0
p5.2.c	7.5		50	0	0	0	0	50	0	0	0	0	50	0	39.5%	426	3	8	5	5	5
p5.2.d	10		80	0	0	0	0	80	0	0	0	0	80	0	47.2%	124764	1274	11	6	6	6
p5.2.e	12.5		180	0	0	0	0	180	0	0	0	0	180	0	21.1%	938	15	14	8	8	8
p5.2.f	15		240	0	0	0	0	240	0	0	0	0	240	0	22.4%	112954	854	14	8	8	8
p5.2.g	17.5		320	0	0	0	0	320	0	0	0	0	320	0	18.4%	244002	1576	18	10	10	10
p5.2.h	20		410	0	0	0	0	410	0	0	0	0	410	0	14.0%	78026	606	20	11	11	11
p5.2.i	22.5		480	0	0	0	2	480	0	0	0	1	480	4.5%	14.6%	179569	3600	22	12	12	12
p5.2.j	25		580	0	0	0	2	580	0	0	0	3	580	0	10.4%	154262	2305	22	12	12	12
p5.2.k	27.5		670	0	0	0	3	670	0	0	0	5	670	2.2%	8.7%	280060	3600	28	15	15	15
p5.3.b	3.3	3	15	0	0	0	0	15	0	0	0	0	15	0	0	0	0	3	3	3	3
p5.3.c	5		30	0	0	11	0	30	0	0	0	0	30	0	15.5%	0	0	6	4	4	4
p5.3.d	6.7		60	0	0	7	0	60	0	0	0	0	60	0	36.9%	1062	8	7	4	4	4
p5.3.e	8.3		105	0	12.5%	83	0	105	0	0	0	0	105	0	32.3%	25439	160	11	5	6	5
p5.3.f	10		120	0	0	9	0	120	0	0	0	0	120	46.8%	47.2%	90076	3600	12	6	6	6
p5.3.g	11.7		190	0	2.6%	49	0	190	0	0	0	0	190	24.2%	37.6%	160699	3600	15	6	7	6
p5.3.h	13.3		270	0	0	7	0	270	0	0	0	0	270	16.4%	29.1%	114450	3600	18	8	8	8
p5.3.i	15		340	0	5.6%	525	17	340	0	2.9%	9	2	340	15.6%	26.7%	117824	3600	19	8	8	8
p5.3.j	16.7		475	0	1.0%	17	2	475	0	0	0	0	475	0	13.4%	156636	2608	24	9	10	9
p5.3.k	18.3		495	0	0	9	2	495	0	0	0	0	490	17.4%	22.1%	88953	3600	24	9	11	9
p5.3.l	20		605	0	1.6%	199	47	605	0	0	2	5	605	9.9%	15.4%	85900	3600	27	11	11	11
p5.3.m	21.7		660	0	0	4	1	660	0	0	0	1	660	15.2%	17.7%	49580	3600	30	12	12	12
p5.3.n	23.3		755	0	1.3%	3423	2372	755	0	0.7%	69	1359	755	11.3%	14.6%	41571	3600	32	12	13	11
p5.3.o	25		870	0	0	0	2	870	0	0	0	3	870	4.8%	10.4%	54458	3600	32	12	12	12
p5.3.p	26.7		990	0	0	0	3	990	0	0	0	5	990	1.5%	6.5%	233758	3600	38	14	14	14
p5.4.c	3.8	4	20	0	0	0	0	20	0	0	0	0	20	0	0	0	0	3	3	3	3
p5.4.d	5		40	0	0	19	0	40	0	0	0	0	40	0	15.5%	0	0	6	4	4	4
p5.4.e	6.2		40	0	0	19	0	40	0	0	0	0	40	0	61.9%	2432	35	6	4	4	4
p5.4.f	7.5		90	0	10.0%	659	2	90	0	0	0	0	90	23.2%	45.5%	153157	3600	10	4	5	4
p5.4.g	8.8		150	0	6.3%	177	0	150	0	0	0	0	150	17.6%	35.9%	132553	3600	14	5	6	5
p5.4.h	10		160	0	0	35	1	160	0	0	0	0	160	54.1%	47.2%	70466	3600	14	6	6	6
p5.4.i	11.2		240	0	7.7%	1335	13	240	0	0	0	0	240	31.5%	36.0%	90799	3600	18	6	7	5
p5.4.j	12.5		350	0	2.8%	149	2	350	0	0	0	0	350	13.8%	23.3%	76102	3600	22	7	8	7
p5.4.k	13.8		360	0	0	9	1	360	0	0	0	0	360	35.9%	33.3%	43969	3600	22	8	8	8
p5.4.l	15		440	0	8.3%	17303	709	440	0	1.6%	21	6	440	29.7%	28.8%	31098	3600	24	8	8	8
p5.4.m	16.2		560	0	9.7%	6341	404	560	0	1.1%	99	61	550	14.2%	21.2%	67376	3600	28	9	9	9
p5.4.n	17.5		640	0	0	7	1	640	0	0	0	0	640	15.0%	18.4%	40823	3600	30	10	10	10
p5.4.o	18.8		700	0	2.8%	4039	746	700	0	0	3	3	675	22.4%	22.6%	27690	3600	32	10	10	10
p5.4.p	20		800	0	2.4%	5575	1717	800	0	0	2	2	800	13.6%	16.1%	24760	3600	34	11	11	11
p5.4.q	21.2		880	0	0	9	3	880	0	0	0	1	875	12.6%	15.5%	23425	3600	38	12	12	12
p5.4.r	22.5		960	0	0	19	10	960	0	0	0	1	875	23.3%	22.2%	14687	3600	42	12	12	12
p5.4.s	23.8		1060	0	1.9%	3063	3600	1060	0	0	2	102	1030	13.5%	15.1%	13354	3600	40	12	12	12
p5.4.t	25		1160	0	0	19	19	1160	0	0	0	3	1150	8.4%	11.2%	15058	3600	42	12	12	12
p5.4.u	26.2		1300	0	0	19	24	1300	0	0	0	4	1300	1.8%	5.6%	17527	3600	46	13	13	13
p5.4.v	27.5		1330	0.4%	0.8%	1167	3600	1325	-	0.5%	65	3600	1330	7.7%	9.4%	15318	3600	50	14	15	14
p5.4.w	28.8		1430	2.7%	3.4%	877	3600	1365	-	7.1%	39	3600	1440	6.0%	7.5%	35398	3600	50	15	15	15

Table 5: Computational results for TOP-DC instances with nodes $n = 66$ and similarity parameter $s = 2, 3$

			s=2												*s=3*											
			BP				*BCP*				*CF*				*BP*				*BCP*				*CF*			
Instance	t_{max}	m	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*
p5.2.b	5	2	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0
p5.2.c	7.5		50	0	0	0	50	0	0	0	50	0	707	5	50	0	0	0	50	0	0	0	50	0	620	3
p5.2.d	10		80	0	0	0	80	0	0	0	80	0	197299	1621	80	0	0	0	80	0	0	0	80	0	115221	479
p5.2.e	12.5		180	0	0	0	180	0	0	0	180	0	2602	16	180	0	0	0	180	0	0	0	180	0	923	10
p5.2.f	15		240	0	0	0	240	0	0	0	240	0	38565	375	240	0	0	0	240	0	0	0	240	0	82859	600
p5.2.g	17.5		320	0	0	0	320	0	0	0	320	0	257382	2305	320	0	0	0	320	0	0	0	320	0	314991	2987
p5.2.h	20		410	0	0	0	410	0	0	0	410	0	172183	1192	410	0	0	0	410	0	0	1	410	0	179132	1382
p5.2.i	22.5		480	0	0	3	480	0	0	2	480	7.6%	153833	3600	480	0	0	0	480	0	0	1	480	6.7%	169463	3600
p5.2.j	25		580	0	0	2	580	0	0	3	580	0	282868	2058	580	0	0	0	580	0	0	3	580	0	202340	1289
p5.2.k	27.5		670	0	0	3	670	0	0	6	670	3.3%	283466	3600	670	0	0	3	670	0	0	5	670	2.8%	291199	3600
p5.3.b	3.3	3	15	0	0	0	15	0	0	0	15	0	0	0	15	0	0	0	15	0	0	0	15	0	0	0
p5.3.c	5		30	0	0	0	30	0	0	0	30	0	0	0	30	0	0	0	30	0	0	0	30	0	0	0
p5.3.d	6.7		60	0	0	0	60	0	0	0	60	0	1986	10	60	0	0	0	60	0	0	0	60	0	1185	8
p5.3.e	8.3		110	0	23	0	110	0	0	0	110	0	14449	65	115	0	0	0	115	0	0	0	115	0	1883	16
p5.3.f	10		120	0	9	0	120	0	0	0	120	44.8%	111976	3600	120	0	9	0	120	0	0	0	120	46.7%	106583	3600
p5.3.g	11.7		190	0	89	0	190	0	0	0	190	21.4%	187139	3600	190	0	59	0	190	0	0	0	190	33.8%	111447	3600
p5.3.h	13.3		270	0	9	0	270	0	0	0	270	16.5%	117954	3600	270	0	19	1	270	0	0	0	270	19.6%	146728	3600
p5.3.i	15		340	0	715	27	340	0	0	3	340	18.3%	102437	3600	350	0	119	6	350	0	0	0	350	12.3%	159817	3600
p5.3.j	16.7		475	0	31	2	475	0	0	1	475	0	170706	1328	475	0	35	3	475	0	0	0	475	0	179641	2260
p5.3.k	18.3		495	0	9	1	495	0	0	0	495	17.7%	119686	3600	495	0	9	1	495	0	0	0	495	16.8%	111741	3600
p5.3.l	20		615	0	7	2	615	0	0	0	615	8.2%	88041	3600	615	0	9	3	615	0	0	0	615	7.7%	101447	3600
p5.3.m	21.7		660	0	7	3	660	0	0	1	660	14.9%	72845	3600	660	0	9	4	660	0	0	1	660	15.0%	78673	3600
p5.3.n	23.3		755	0.1%	4317	3600	755	0	53	571	755	11.7%	66135	3600	760	0	959	735	760	0	1	12	760	9.5%	83680	3600
p5.3.o	25		870	0	0	3	870	0	0	3	870	4.9%	66015	3600	870	0	0	3	870	0	0	3	870	6.5%	84314	3600
p5.3.p	26.7		990	0	0	2	990	0	0	4	990	2.3%	135899	3600	990	0	0	3	990	0	0	5	990	1.3%	254776	3600
p5.4.c	3.8	4	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0
p5.4.d	5		40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0
p5.4.e	6.2		40	0	0	0	40	0	0	0	40	0	4710	32	40	0	0	0	40	0	0	0	40	0	5223	40
p5.4.f	7.5		100	0	27	0	100	0	0	0	100	0	139564	1255	100	0	0	0	100	0	0	0	100	0	314314	2136
p5.4.g	8.8		150	0	147	1	150	0	0	0	150	19.2%	127068	3600	150	0	295	3	150	0	0	0	150	13.4%	175178	3600
p5.4.h	10		160	0	39	1	160	0	0	0	160	53.3%	78823	3600	160	0	9	0	160	0	0	0	160	52.7%	93433	3600
p5.4.i	11.2		240	0	2193	39	240	0	0	0	240	29.8%	81495	3600	250	0	255	7	250	0	0	0	250	21.2%	116101	3600
p5.4.j	12.5		350	0	177	4	350	0	0	0	350	12.1%	90867	3600	350	0	147	5	350	0	0	0	350	10.7%	107523	3600
p5.4.k	13.8		360	0	25	1	360	0	0	0	360	32.8%	68930	3600	360	0	49	3	360	0	0	0	360	32.6%	62711	3600
p5.4.l	15		440	0	44685	2274	440	0	55	9	440	28.1%	53302	3600	460	0	1123	562	460	0	0	0	460	20.3%	55297	3600
p5.4.m	16.2		560	0	12455	946	560	0	99	27	560	13.8%	89844	3600	560	0	15537	1208	560	0	0	13	560	13.2%	72679	3600
p5.4.n	17.5		640	0	9	1	640	0	0	0	640	14.4%	64178	3600	640	0	29	4	640	0	0	0	640	14.5%	64515	3600
p5.4.o	18.8		700	0	5315	1072	700	0	0	0	700	17.8%	48070	3600	710	0	719	138	710	0	0	1	700	18.2%	42092	3600
p5.4.p	20		820	0	13	3	820	0	0	0	810	11.7%	33668	3600	820	0	29	8	820	0	0	0	800	12.1%	21170	3600
p5.4.q	21.2		880	0	19	8	880	0	0	2	880	12.2%	31255	3600	880	0	19	9	880	0	0	0	875	12.7%	38286	3600
p5.4.r	22.5		960	0	19	12	960	0	0	1	940	14.6%	22982	3600	960	0	19	13	960	0	0	2	920	16.5%	24167	3600
p5.4.s	23.8		1080	0	19	16	1080	0	0	2	1050	10.9%	17112	3600	1080	0	19	16	1080	0	0	3	1050	11.6%	33539	3600
p5.4.t	25		1160	0	9	9	1160	0	0	3	1160	7.2%	22197	3600	1160	0	39	36	1160	0	0	3	1160	7.5%	29299	3600
p5.4.u	26.2		1300	0	9	14	1300	0	0	4	1300	2.3%	19086	3600	1300	0	27	41	1300	0	0	5	1300	2.0%	23510	3600
p5.4.v	27.5		1340	0	47	87	1330	0	63	1807	1330	7.2%	26042	3600	1340	0	5	9	1340	0	0	26	1335	6.8%	34124	3600
p5.4.w	28.8		1470	0.1%	1071	3600	1465	0	25	1301	1470	2.9%	45733	3600	1470	0.3%	1313	3600	1470	0	0	14	1470	3.8%	123114	3600

Table 6: Computational results for TOP-DC instances with nodes $n = 66$ and similarity parameter $s = 4, 5$

			s=4												*s=5*											
			BP				*BCP*				*CF*				*BP*				*BCP*				*CF*			
Instance	t_{max}	m	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*	*val*	*optgap*	*nodes*	*time*
p5.2.b	5	2	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0
p5.2.c	7.5		50	0	0	0	50	0	0	0	50	0	1284	6	50	0	0	0	50	0	0	0	50	0	0	3
p5.2.d	10		80	0	0	0	80	0	0	0	80	0	126747	816	80	0	0	0	80	0	0	0	80	0	180390	658
p5.2.e	12.5		180	0	0	0	180	0	0	0	180	0	2630	16	180	0	0	0	180	0	0	0	180	0	2493	9
p5.2.f	15		240	0	0	0	240	0	0	0	240	0	76630	657	240	0	0	0	240	0	0	0	240	0	121287	461
p5.2.g	17.5		320	0	0	0	320	0	0	1	320	0	226052	1542	320	0	0	0	320	0	0	0	320	0	170443	840
p5.2.h	20		410	0	0	0	410	0	0	1	410	0	155374	663	410	0	0	0	410	0	0	1	410	0	188317	526
p5.2.i	22.5		480	0	0	1	480	0	0	1	480	5.3%	191868	3600	480	0	0	2	480	0	0	2	480	3.8%	246879	3600
p5.2.j	25		580	0	0	2	580	0	0	3	580	0	273288	2022	580	0	0	2	580	0	0	3	580	0	149652	485
p5.2.k	27.5		670	0	0	3	670	0	0	6	670	1.9%	400063	3600	670	0	0	4	670	0	0	6	670	0	697241	3031
p5.3.b	3.3	3	15	0	0	0	15	0	0	0	15	0	0	0	15	0	0	0	15	0	0	0	15	0	0	0
p5.3.c	5		30	0	0	0	30	0	0	0	30	0	0	0	30	0	0	0	30	0	0	0	30	0	0	0
p5.3.d	6.7		60	0	0	0	60	0	0	0	60	0	1047	10	60	0	0	0	60	0	0	0	60	0	1483	5
p5.3.e	8.3		120	0	0	0	120	0	0	0	120	0	188	10	120	0	0	0	120	0	0	0	120	0	253	6
p5.3.f	10		120	0	0	0	120	0	0	0	120	47.0%	121687	3600	120	0	0	0	120	0	0	0	120	43.5%	203246	3600
p5.3.g	11.7		195	0	9	0	195	0	0	0	195	21.0%	168609	3600	195	0	0	1	195	0	0	0	195	22.4%	194641	3600
p5.3.h	13.3		270	0	9	0	270	0	0	0	270	16.5%	124716	3600	270	0	19	1	270	0	0	0	270	16.3%	181429	3600
p5.3.i	15		360	0	7	0	360	0	0	0	360	11.2%	131226	3600	360	0	9	1	360	0	0	0	360	10.9%	165316	3600
p5.3.j	16.7		475	0	35	3	475	0	0	0	475	0	179774	2141	475	0	35	3	475	0	0	0	475	0	149512	792
p5.3.k	18.3		495	0	17	3	495	0	0	0	495	17.4%	101592	3600	495	0	17	3	495	0	0	0	495	15.8%	182101	3600
p5.3.l	20		615	0	9	3	615	0	0	0	615	7.9%	100710	3600	615	0	9	3	615	0	0	0	615	7.5%	126563	3600
p5.3.m	21.7		660	0	9	5	660	0	0	1	660	14.7%	94375	3600	660	0	19	11	660	0	0	1	660	13.7%	137257	360
p5.3.n	23.3		765	0	29	18	765	0	0	0	765	9.4%	77076	3600	765	0	9	9	765	0	0	4	765	9.4%	128675	3600
p5.3.o	25		870	0	0	3	870	0	0	3	870	5.7%	90890	3600	870	0	0	3	870	0	0	0	870	5.6%	163549	3600
p5.3.p	26.7		990	0	7	10	990	0	0	5	990	1.3%	213965	3600	990	0	5	11	990	0	0	5	990	1.7%	226524	3600
p5.4.c	3.8	4	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0	20	0	0	0
p5.4.d	5		40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0	40	0	0	0
p5.4.e	6.2		40	0	0	0	40	0	0	0	40	0	26559	134	40	0	0	0	40	0	0	0	40	0	23059	78
p5.4.f	7.5		100	0	0	0	100	0	0	0	100	0	293665	1611	100	0	0	0	100	0	0	0	100	0	131072	556
p5.4.g	8.8		160	0	0	6	160	0	0	0	160	0	228992	3347	160	0	0	0	160	0	0	0	160	0	104641	750
p5.4.h	10		160	0	0	0	160	0	0	0	160	54.2%	87427	3600	160	0	0	0	160	0	0	0	160	51.8%	144135	3600
p5.4.i	11.2		260	0	27	0	260	0	0	0	260	19.3%	95751	3600	260	0	0	0	260	0	0	0	260	16.3%	178557	3600
p5.4.j	12.5		350	0	175	3	350	0	0	0	350	12.3%	99120	3600	350	0	331	11	350	0	0	0	350	8.8%	158193	3600
p5.4.k	13.8		360	0	39	1	360	0	0	0	360	32.1%	80378	3600	360	0	39	2	360	0	0	0	360	31.0%	98499	3600
p5.4.l	15		480	0	25	3	480	0	0	0	480	15.9%	80981	3600	480	0	27	2	480	0	0	0	480	16.4%	95923	3600
p5.4.m	16.2		560	0	35207	3010	560	0	55	9	560	12.8%	79605	3600	590	0	4715	479	590	0	0	0	590	7.7%	90738	3600
p5.4.n	17.5		640	0	19	2	640	0	0	0	640	15.0%	67292	3600	640	0	39	6	640	0	0	0	640	14.1%	108516	3600
p5.4.o	18.8		720	0	29	5	720	0	0	1	710	16.2%	48136	3600	720	0	37	10	720	0	0	1	710	16.3%	85323	3600
p5.4.p	20		820	0	9	4	820	0	0	0	820	9.9%	69294	3600	820	0	29	11	820	0	0	0	810	10.1%	91337	3600
p5.4.q	21.2		880	0	37	10	880	0	0	2	880	11.8%	47105	3600	880	0	31	18	880	0	0	2	880	11.4%	89259	3600
p5.4.r	22.5		960	0	77	43	960	0	0	1	935	14.4%	26341	3600	960	0	49	91	960	0	0	2	960	11.8%	81063	3600
p5.4.s	23.8		1080	0	27	21	1080	0	0	2	1050	10.9%	31850	3600	1080	0	29	38	1080	0	0	3	1070	7.6%	78796	3600
p5.4.t	25		1160	0	9	10	1160	0	0	3	1160	7.2%	28329	3600	1160	0	39	57	1160	0	0	3	1160	6.2%	72292	3600
p5.4.u	26.2		1300	0	25	41	1300	0	0	4	1300	2.1%	80058	3600	1300	0	29	72	1300	0	0	5	1300	1.0%	76979	3600
p5.4.v	27.5		1340	0	9	12	1340	0	0	25	1340	6.8%	69760	3600	1340	0	19	47	1340	0	0	24	1340	6.8%	159074	3600
p5.4.w	28.8		1480	0	11	23	1480	0	0	7	1480	3.0%	124195	3600	1480	0	9	25	1480	0	0	8	1480	2.9%	184502	3600

6.2. Tests on a Chinese OTA dataset

Dataset description.

The Chinese OTA *niding.net* provided us with 1379 orders information of tour products sold in the last three years. These orders cover famous tourism destinations distributed among Europe, Asia and North/South America. To construct our dataset, the tour products were decomposed into several elementary components, including the visited cities and the relevant stay times and costs. In particular, 40 European cities (see Figure 3.a) among them were selected to represent the nodes in our Europe dataset (EuroData). For the sake of triangular inequality requirement on travel times/costs in solving subproblems, only the direct flights between cities were considered in our EuroData, even though some close cities without flights may be connected by railways. Moreover, the attractiveness (score) of destinations are measured by the "nights spent in tourist accommodation" in all NUTS-2 regions (EuroStat, 2016). The basic information of the selected European 40 cities are summarized in Table 7.

Table 7: Basic information of the selected 40 cities in EuroData

city	*node #*	*stay days*	*daily cost** $	*value***	*score/day*	*city*	*node*	*stay days*	*daily cost* $	*value*	*score/day*
Paris (Hub)	0	3	143	71.2	19	Ljubljana	20	1	91	6.8	3
Amsterdam	1	2	97	27.4	13	London	21	3	204	123.9	20
Athens	2	2	92	57.3	17	Lyon	22	2	152	49.1	16
Barcelona	3	3	131	79.8	20	Madrid	23	2	171	66.2	18
Belgrade	4	1	109	6.6	3	Malaga	24	1	117	9.0	4
Berlin	5	2	117	30.9	13	Malta	25	1	142	25.1	12
Bordeaux	6	2	118	32.9	14	Milan	26	2	170	37.2	15
Brussels	7	1	119	21.1	11	Munich	27	2	113	35.1	15
Budapest	8	1	74	10.5	4	Nice	28	2	78	51.8	17
Copenhagen	9	1	91	10.7	5	Oslo	29	1	83	7.0	3
Dublin	10	1	111	23.8	12	Prague	30	1	96	16.8	10
Dusseldorf	11	1	112	11.3	5	Reykjavik	31	1	165	7.8	3
Florence	12	2	106	44.2	16	Rome	32	2	118	32.1	14
Frankfurt	13	1	194	18.3	11	Split	33	3	126	74.2	19
Geneva	14	1	175	14.1	8	Stockholm	34	1	66	13.4	7
Hamburg	15	1	99	13.3	7	Stuttgart	35	1	227	11.7	6
Helsinki	16	1	110	5.5	3	Venice	36	2	74	65.4	18
Istanbul	17	1	179	17.6	10	Vienna	37	1	73	14.8	8
Koln	18	1	102	12.8	6	Warsaw	38	1	75	8.1	3
Lisbon	19	1	109	16.3	9	Zurich	39	1	180	15.7	9

*Daily cost is the hotel price from *booking.com*
**This value is the real number of "nights spent in tourist accommodation" (in millions) (EuroStat, 2016).

After analyzing the orders information in detail, we found that a tourist spent on average around USD 2300 for a trip of around 10 days in Europe. In our test instances, the travel budget is restricted by $c_{max} \in \{1500, 2300\}$ USD and the travel time is limited by $t_{max} \in \{10, 14\}$ days. By varying m and s, we get the results in Table 8, where the composition of each tour is shown (with its collected score in brackets).

Table 8: Computational results for TOP-DC instances with EuroData from a Chinese OTA company

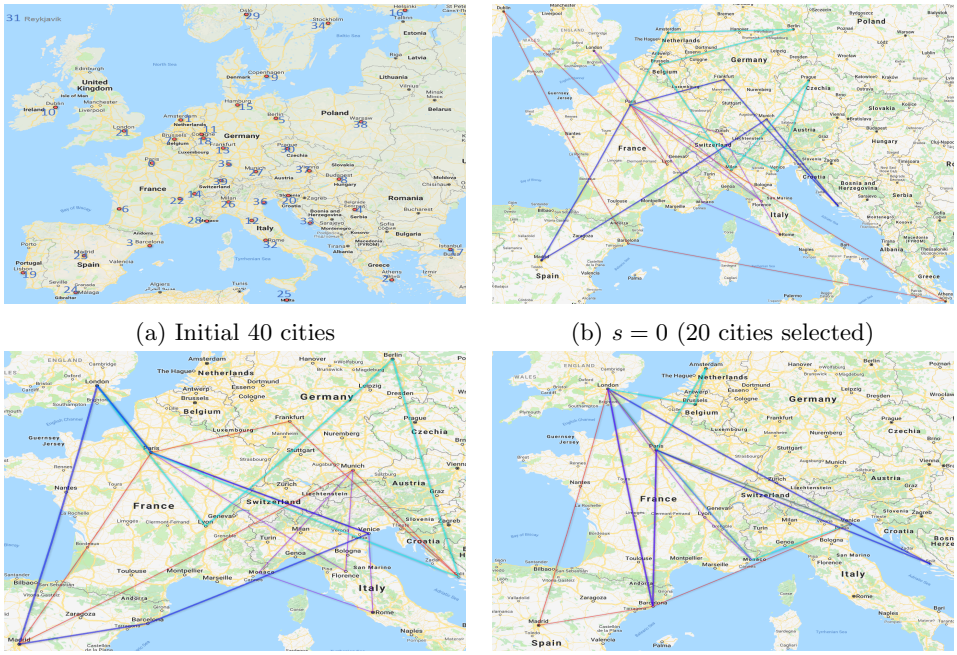
(t_{max}, c_{max})	m	s	$score$	$avgscore/day$	gap	t	$Tour1$	$Tour2$	$Tour3$	$Tour4$	actual uniformity
(10,1500)	3	0	479	16.4	0	0	0-10-23-3-0 (165)	0-21-30-36-0 (163)	0-22-32-2-0 (151)		0.33
(10,2300)	3	0	497	17.1	0	1	0-3-21-0 (177)	0-22-23-36-0 (161)	0-33-13-28-0 (159)		0.33
(14,1500)	3	0	526	16.0	0	0	0-2-23-12-0 (159)	0-27-33-5-0 (170)	0-28-3-30-36-0 (197)		0.33
(14,2300)	3	0	671	16.1	0	16	0-1-6-22-28-32-0 (205)	0-12-27-33-13-26-0 (217)	0-21-23-36-3-0 (249)		0.33
(10,1500)	4	0	638	16.4	0	4	0-10-23-3-0 (165)	0-21-30-36-0 (163)	0-22-32-2-0 (151)	0-28-7-33-0 (159)	0.25
(10,2300)	4	0	647	16.7	0	10	0-3-21-0 (177)	0-10-26-33-0 (156)	0-22-27-12-0 (151)	0-23-28-36-0 (163)	0.25
(14,1500)	4	0	671	16.3	0	4	0-2-23-12-0 (159)	0-27-33-5-0 (170)	0-28-3-36-30-0 (197)	0-6-21-0 (145)	0.25
(14,2300)	4	0	830	15.3	2.4	3600	0-1-5-7-26-30-36-0 (196)	0-2-3-10-22-0 (195)	0-6-23-27-33-13-0 (219)	0-12-21-28-32-39-0 (220)	0.25
(10,1500)	3	1	492	16.9	0	0	0-10-23-3-0 (165)	0-21-30-36-0 (163)	0-36-3-13-0 (164)		0.43
(10,2300)	3	1	504	17.5	0	0	0-3-21-0 (177)	0-10-21-28-0 (163)	0-36-7-21-0 (164)		0.44
(14,1500)	3	1	535	17.3	0	1	0-3-22-28-0 (183)	0-21-30-36-0 (163)	0-36-3-23-0 (189)		0.43
(14,2300)	3	1	694	16.6	0	45	0-21-23-3-36-0 (249)	0-3-12-13-27-28-0 (224)	0-26-33-7-28-22-0 (221)		0.39
(10,1500)	4	1	651	16.9	0	1	0-10-23-3-0 (165)	0-21-30-36-0 (163)	0-36-3-13-0 (164)	0-22-28-36-0 (159)	0.33
(10,2300)	4	1	669	17.3	0	2	0-3-21-0 (177)	0-10-21-28-0 (163)	0-36-7-21-0 (164)	0-10-23-3-0 (165)	0.39
(14,1500)	4	1	718	16.7	0	14	0-3-22-28-0 (183)	0-2-3-36-0 (187)	0-6-28-32-36-0 (183)	0-10-23-3-0 (165)	0.36
(14,2300)	4	1	917	16.5	0	374	0-21-23-3-36-0 (249)	0-6-23-27-33-13-0 (219)	0-12-27-28-32-36-0 (217)	0-21-22-5-33-0 (232)	0.35
(10,1500)	3	2	492	16.9	0	1	0-10-23-3-0 (165)	0-21-30-36-0 (163)	0-36-3-13-0 (164)		0.43
(14,1500)	3	2	575	17.1	0	0	0-2-3-36-0 (187)	0-36-3-23-0 (189)	0-28-36-3-25-0 (199)		0.56
(14,2300)	3	2	726	17.8	0	0	0-3-21-22-28-0 (243)	0-3-21-33-0 (234)	0-21-23-3-36-0 (249)		0.52
(10,1500)	4	2	656	16.9	0	4	0-10-23-3-0 (165)	0-21-30-36-0 (163)	0-36-3-13-0 (164)	0-3-7-36-0 (164)	0.38
(14,1500)	4	2	760	17.2	0	1	0-2-3-36-0 (187)	0-36-3-23-0 (189)	0-36-28-3-25-0 (199)	0-36-3-22-0 (185)	0.46
(14,2300)	4	2	950	17.4	0	77	0-3-21-22-28-0 (243)	0-3-21-33-0 (234)	0-21-23-3-36-0 (249)	0-1-7-21-28-36-0(224)	0.53

Numerical results and analysis.

In Table 8, we first observe that with the same resource capacity, the total number of distinct cities visited by the m tours decreases with s, possibly in large amounts (e.g. for $(14, 2300)$ and $m = 4$, the selected tours visit $20, 13$ and 9 cities in total for $s = 0, 1, 2$, respectively). As expected, this demonstrates the opportunity to condense potential destinations into a small range of attractive cities, in order to bargain better prices with suppliers. Moreover, the total score collected in a tour package naturally increases with capacities (t_{max}, c_{max}), while the average score per day does not necessarily follow the same trend, unless a certain level of similarity is allowed and each tour is long enough. The reason is that for a lower s and a few short tours planning, it forces the tour to discard some most attractive cities and select some with lower scores. Also, we note that for a given setting, increasing m by one may completely reshuffle the composition of the tours (e.g., there is only one common tour in the solutions of rows $(14, 1500)|3|1$ and $(14, 1500)|4|1$, and in the solutions of rows $(14, 2300)|3|1$ and $(14, 2300)|4|1$), whereas in some other cases the m existing tours are kept when increasing m (see the rows for $s = 2$).

Insights for digital travel.

As mentioned earlier, the OTA is recommended to offer several potential package tours to customers in a given range of time and budget, rather than a single one. This enables to increase the opportunities to satisfy the customers' individual preference, and thus successful sales transactions. By increasing the number of common cities between tours without sacrificing diversity of the offer, the company confirmed they obtained better discounts from suppliers with economies of scale, which finally brings customer value in a win-win logic. As shown in Figure 3, the occurrence of popular cities and thus the average score of tours increase with the similarity parameter, indicating that the company has more chance to aggregate their customers into these attractive destinations. This is an



(a) Initial 40 cities

(b) $s = 0$ (20 cities selected)

(c) $s = 1$ (13 cities selected)

(d) $s = 2$ (9 cities selected)

Figure 3: The generated 4 tours under capacity $(14, 2300)$ for different similarity parameter s

argument for not being too restrictive in the similarity parameter s when managing the trade-off between diversification of products, leading to more flexible choice for customers, and standardization, meaning higher discounts from suppliers by economies of scale.

The last column "actual uniformity" in Table 8 indicates the percentage of tours where a city is present in average. The value 0.33 in the first row means that in average, a city is present in 33% of the 3 tours proposed in the optimal solution, hence in 1 tour over 3. A low uniformity means high diversity. Naturally, the uniformity increases with a higher s for a given setting of other parameters. Only rows $(10, 1500)|3|1$ and $(10, 1500)|3|2$ compare the same because the selected tours for $s = 2$ have similarity one in the diversity constraints, which is due to tight time capacity limiting the tour length and feasible combinations of cities. However, we observe that the increase of uniformity with s is generally lower for the highest time and cost capacities (see rows $(14, 2300)|3|0$ vs $(14, 2300)|3|1$). The reason is that less tight capacities on the tours provides more flexibility to find multiple tours with less common nodes. Higher customer resources favor the diversity of the products offered, which makes sense.

A related issue is the integration of the few scenarios of different profiles (t_{max}, c_{max}) typically asked by customers into a single decision model, to concentrate the whole demand on fewer selected cities in a global view, thus linking all scenarios.

Last but not least, regarding real-time changes on flights and hotels prices, it is convenient for the company to implement our TOP-DC tool to update their travel products dynamically and automatically, thus saving labor force for collecting data and re-building package products.

7. Conclusion

In this paper, we studied a problem of travel product design for online travel agencies. Our research is motivated by the tour design operations of a Chinese company. We formulated the problem as a Team Orienteering Problem with additional Diversity Constraints (TOP-DC). To manage the trade-off between customers' freedom of choice and suppliers discount, we design a collection of tours with controlled diversity between tours. The conclusions are twofold. On a tractability point of view, the two extended formulations were demonstrated to perform well on our numerical experiments, with most of instances solved to optimality. The first extended formulation is based on two-index assignment variables and a compact set of constraints, whereas the second one is based on one-index tour selection variables and an exponential-size set of diversity constraints to be generated on the fly. The Branch-Cut-and-Price method (column-and-row generation) associated with the one-index formulation generally outperformed the Branch-and-Price method based on the two-index formulation for larger graphs.

On managerial insights, we conclude that the similarity parameter between tours is a key decision for online travel agencies to drive their sales performance. We found that one can align the goals of satisfying individual preferences of customers and gaining discounting benefits from suppliers by economies of scale. A reasonable setting is to limit the similarity between tours to less than 50% of the cities (typically, two cities in common for package tours with 4 or 5 cities), in order to give enough freedom to a customer for selecting one tour among the m tours proposed in her budget and time limit, while concentrating the whole demand on much fewer cities than in the classical TOP, which only allows completely disjoint tours. Lastly, as the decision model applies to a digital business where flights, hotels and costs data can be dynamically updated, the decision tool can help to save labor force for collecting data, in a very competitive environment, and re-design automatically the offer of online-products based on new data. An interesting avenue of future research would also be to control the number of cities selected overall, if several combinations of budget and time limit were put inside the same model.

References

Abbaspour, R. A., & Samadzadegan, F. (2011). Time-dependent personal tour planning and scheduling in metropolises. *Expert Systems with Applications*, *38*, 12439–12452.

Archetti, C., Corberán, Á., Plana, I., Sanchis, J. M., & Speranza, M. G. (2015). A matheuristic for the team orienteering arc routing problem. *European Journal of Operational Research*, *245*, 392–401.

Archetti, C., Hertz, A., & Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. *Journal of Heuristics*, *13*, 49–76.

Archetti, C., Speranza, M. G., Corberán, Á., Sanchis, J. M., & Plana, I. (2013). The team orienteering arc routing problem. *Transportation Science*, *48*, 442–457.

Archetti, C., Speranza, M. G., & Hertz, A. (2006). A tabu search algorithm for the split delivery vehicle routing problem. *Transportation Science*, *40*, 64–73.

Baldacci, R., Mingozzi, A., & Roberti, R. (2011). New route relaxation and pricing strategies for the vehicle routing problem. *Operations Research*, *59*, 1269–1283.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. *Operations Research*, *46*, 316–329.

Boussier, S., Feillet, D., & Gendreau, M. (2007). An exact algorithm for team orienteering problems. *4or*, *5*, 211–230.

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

Butt, S. E., & Ryan, D. M. (1999). An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, *26*, 427–441.

Chao, I. M., Golden, B. L., & Wasil, E. A. (1996a). A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, *88*, 475–489.

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

Chiappa, G. D. (2013). Internet versus travel agencies: The perception of different groups of italian online buyers. *Journal of Vacation Marketing*, *19*, 55–66.

Clemons, E. K., Hann, I. H., & Hitt, L. M. (2002). Price dispersion and differentiation in online travel: An empirical investigation. *Management Science*, *48*, 534–549.

Costa, L., Contardo, C., & Desaulniers, G. (2018). Exact branch-price-and-cut algorithms for vehicle routing. *Transportation Science*, (p. to appear).

Dang, D.-C., Guibadj, R. N., & Moukrim, A. (2013). An effective pso-inspired algorithm for the team orienteering problem. *European Journal of Operational Research*, *229*, 332–344.

Desaulniers, G., Errico, F., Irnich, S., & Schneider, M. (2016). Exact algorithms for electric vehicle-routing problems with time windows. *Operations Research*, *64*, 1388–1405.

Desaulniers, G., Madsen, O. B. G., & Ropke, S. (2014). Chapter 5: The vehicle routing problem with time windows. In *Vehicle Routing: Problems, Methods, and Applications, Second Edition* (pp. 119–159). SIAM.

Desrosiers, J., & Lübbecke, M. E. (2005). A primer in column generation. In *Column Generation* (pp. 1–32). Springer.

Divsalar, A., Vansteenwegen, P., Sörensen, K., & Cattrysse, D. (2014). A memetic algorithm for the orienteering problem with hotel selection. *European Journal of Operational Research*, *237*, 29–49.

Enoch, Y. (1996). Contents of tour packages: A cross-cultural comparison. *Annals of Tourism Research*, *23*, 599–616.

EuroStat (2016). Nights spent in tourist accommodation, by NUTS 2 regions, 2016. `https://ec.europa.eu/eurostat/statistics-explained/index.php?title=File:Nights_spent_in_tourist_accommodation,_by_NUTS_2_regions,_2016_(million_nights_spent_by_residents_and_non-residents)-RYB18.png`.

Feillet, D., Dejax, P., Gendreau, M., & Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. *Networks*, *44*, 216–229.

Fischetti, M., Gonzalez, J. J. S., & Toth, P. (1998). Solving the orienteering problem through branch-and-cut. *INFORMS Journal on Computing*, *10*, 133–148.

Gavalas, D., Konstantopoulos, C., Mastakas, K., & Pantziou, G. (2014). A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, *20*, 291–328.

Gianessi, P., Alfandari, L., Létocart, L., & Calvo, R. W. (2016). A column generation based heuristic for the multicommodity-ring vehicle routing problem. *Transportation Research Procedia*, *12*, 227–238.

Godart, J.-M. (1999). Combinatorial optimisation based decision support system for trip planning. In *Information and Communication Technologies in Tourism 1999* (pp. 318–327). Springer.

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*, 315–332.

Gutiérrez-Jarpa, G., Desaulniers, G., Laporte, G., & Marianov, V. (2010). A branch-and-price algorithm for the vehicle routing problem with deliveries, selective pickups and time windows. *European Journal of Operational Research*, *206*, 341–349.

Herzog, D., & Wörndl, W. (2014). A travel recommender system for combining multiple travel regions to a composite trip. In *CBRecSys@ RecSys* (pp. 42–48).

Heung, V. C., & Chu, R. (2000). Important factors affecting hong kong consumers choice of a travel agency for all-inclusive package tours. *Journal of Travel Research*, *39*, 52–59.

Hu, Q., & Lim, A. (2014). An iterative three-component heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, *232*, 276–286.

Irnich, S., & Desaulniers, G. (2005). Shortest path problems with resource constraints. In *Column Generation* (pp. 33–65). New York: Springer.

Ke, L., Xu, Z., Feng, Z., Shang, K., & Qian, X. (2013). Proportion-based robust optimization and team orienteering problem with interval data. *European Journal of Operational Research*, *226*, 19–31.

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

Labadie, N., Mansini, R., Wolfler Calvo, R., & Melechovskỳ, J. (2012). The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, *220*, 15–27.

Laporte, G., & Martello, S. (1990). The selective travelling salesman problem. *Discrete Applied Mathematics*, *26*, 193–207.

Law, R., Leung, K., & Wong, R. (2004). The impact of the internet on travel agencies. *International Journal of Contemporary Hospitality Management*, *16*, 100–107.

Lee, A., Basak, D. G., & Law, R. (2013). An examination of the relationship between online travel agents and hotels: A case study of choice hotels international and expedia. com. *Cornell Hospitality Quarterly*, *54*, 95–107.

Lewis, I., Janjaap, S., & Talalayevsky, A. (1998). The impact of information technology on travel agents. *Transportation Journal*, (pp. 20–25).

Lin, S.-W., & Vincent, F. Y. (2012). A simulated annealing heuristic for the team orienteering problem with time windows. *European Journal of Operational Research*, *217*, 94–107.

Luo, Z., Cheang, B., Lim, A., & Zhu, W. (2013). An adaptive ejection pool with toggle-rule diversification approach for the capacitated team orienteering problem. *European Journal of Operational Research*, *229*, 673–682.

Lusby, R. M., Haahr, J. T., Larsen, J., & Pisinger, D. (2017). A branch-and-price algorithm for railway rolling stock rescheduling. *Transportation Research Part B: Methodological*, *99*, 228–250.

Malucelli, F., Giovannini, A., & Nonato, M. (2015). Designing single origin-destination itineraries for several classes of cycle-tourists. *Transportation Research Procedia*, *10*, 413–422.

Mei, Y., Salim, F. D., & Li, X. (2016). Efficient meta-heuristics for the multi-objective time-dependent orienteering problem. *European Journal of Operational Research*, *254*, 443–457.

Morrison, A. M. (1989). Hospitality and tourism marketing. *Albany, NY: Delmar*, .

Poggi, M., Viana, H., & Uchoa, E. (2010). The team orienteering problem: Formulations and branch-cut and price. In *OASIcs-OpenAccess Series in Informatics* (pp. 142–155). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik volume 14.

Prieto, M. (2017). 10 Online Travel Public Companies. `https://medium.com/traveltechmedia/10-online-travel-public-companies-dee6df73f768`.

Riera-Ledesma, J., & Salazar-González, J. J. (2017). Solving the team orienteering arc routing problem with a column generation approach. *European Journal of Operational Research*, *262*, 14–27.

Righini, G., & Salani, M. (2006). *Dynamic programming for the orienteering problem with time windows*. Technical Report Dipartimento di Tecnologie dell'Informazione, Universita degli Studi Milano, Crema, Italy.

Righini, G., & Salani, M. (2009). Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. *Computers & Operations Research*, *36*, 1191–1203.

Salani, M., & Vacca, I. (2011). Branch and price for the vehicle routing problem with discrete split deliveries and time windows. *European Journal of Operational Research*, *213*, 470–477.

Sheldon, P. J. (1986). The tour operator industry: an analysis. *Annals of Tourism Research*, *13*, 349–365.

Sheldon, P. J., & Mak, J. (1987). The demand for package tours: A mode choice model. *Journal of travel research*, *25*, 13–17.

Song, Y., Ulmer, M. W., Thomas, B. W., & Wallace, S. W. (2018). Building trust in home services-stochastic team-orienteering with consistency constraints. *working paper*, .

Souffriau, W., Vansteenwegen, P., Vertommen, J., Vanden Berghe, G., & Van Oudheusden, D. (2008). A personalized tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, *22*, 964–985.

Statista (2015). Share of U.S. shoppers on online travel agency (OTA) sites, by where they complete their booking, in 2015. `https://www.statista.com/statistics/455895/ota-us-shopper-booking-completion/`.

Statista (2016). Digital travel sales worldwide from 2014 to 2020. `http://https://www.statista.com/statistics/499694/forecast-of-online-travel-sales-worldwide/`.

Sylejmani, K., Muhaxhiri, A., Dika, A., & Ahmedi, L. (2014). Solving tourist trip planning problem via a simulated annealing algorithm. In *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on* (pp. 1124–1129). IEEE.

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

Tarantilis, C. D., Stavropoulou, F., & Repoussis, P. P. (2013). The capacitated team orienteering problem: a bi-level filter-and-fan method. *European Journal of Operational Research*, *224*, 65–78.

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

Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Van Oudheusden, D. (2009a). A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, *196*, 118–127.

Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Van Oudheusden, D. (2009b). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, *36*, 3281–3290.

Verbeeck, C., Sörensen, K., Aghezzaf, E.-H., & Vansteenwegen, P. (2014a). A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, *236*, 419–432.

Verbeeck, C., Vansteenwegen, P., & Aghezzaf, E.-H. (2014b). An extension of the arc orienteering problem and its application to cycle trip planning. *Transportation Research Part E: Logistics and Transportation Review*, *68*, 64–78.

Werthner, H. (2002). Intelligent systems in travel and tourism. IJCAI 2003: 18th International Joint Conference on Artificial Intelligence.

Wong, C. S., & Kwong, W. Y. (2004). Outbound tourists selection criteria for choosing all-inclusive package tours. *Tourism Management*, *25*, 581–592.