



Discrete Optimization

The single vehicle routing problem with toll-by-weight scheme: A branch-and-bound approach

Zizhen Zhang^a, Hu Qin^{b,*}, Wenbin Zhu^c, Andrew Lim^a^a Department of Management Sciences, City University of Hong Kong, Tat Chee Ave, Kowloon Tong, Kowloon, Hong Kong^b School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China^c Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong

ARTICLE INFO

Article history:

Received 16 June 2011

Accepted 17 January 2012

Available online 25 January 2012

Keywords:

Branch and bound

Vehicle routing problem

Toll-by-weight

ABSTRACT

Expressways in China make use of the toll-by-weight scheme, in which expressway tolls are collected based on the weight and traveling distance of the vehicle. Most vehicle routing models assume that the cost of traversing each edge is equivalent to edge length or some constant; as a result, such models cannot be practically applied to the Chinese expressway transportation system. This study addresses a new single vehicle routing problem that takes the vehicle's (laden and unladen) weight into account. To solve this problem exactly, we provide a branch-and-bound algorithm with a provably valid lower bound measure, along with five dominance checkers for additional pruning. We analyze our algorithm using instances generated from standard TSP test cases, as well as two new sets of test cases based on real expressway information from the Gansu and Jiangxi provinces in China. The algorithm can be applied to any toll scheme in which the toll per unit distance monotonically increases with weight, even if the toll function is non-linear.

© 2012 Elsevier B.V. All rights reserved.

1. Introduction

The construction of modern expressway systems has significantly accelerated the economic and social development of China by facilitating the establishment of a unified market system, increasing the efficiency of modern logistics and narrowing the development gaps among different regions. At the end of 2010, China's expressway network has a total length of over 74,000 km (Wikipedia, 2011), second only to the United States, and its overall construction cost exceeds 240 billion US dollars.

In some countries like Australia and the United States, the majority of expressways are state-owned and toll-free. In contrast, almost all expressways in China are currently owned by for-profit corporations that raise construction funds from securities markets or banks, and recoup investments through tolls. Previously, and for a long time, expressway tolls in China have been levied based on the type of the vehicle (e.g., tonnage or seating capacity) and traveling distance. Under this toll scheme, a vehicle is charged the same toll regardless of whether it is empty, normally loaded or overloaded. This toll scheme violates the principle of equity where a greater load should incur greater cost, and encourages transpor-

tation service providers to overload their vehicles for economic benefits.

The overloading of transport vehicles brings about several serious issues. Firstly, overloaded vehicles damage the expressways, and the damage caused grows exponentially as the load increases; this leads to higher maintenance cost and shortens the service life-time of the expressway. Secondly, an overloaded vehicle puts the driver and other expressway users at risk. Braking and steering are significantly more difficult when the maximum weight limit of a vehicle is exceeded. Furthermore, many vehicles were illegally modified to enable them to carry greater loads, and the resultant instability of the vehicles is the cause of a large number of fatal traffic accidents. Thirdly, overloading undercuts the transportation service providers that do comply with the transportation regulations, which gives rise to cut-throat competition. Since providers that overload vehicles are able to quote lower transportation rates and gain greater market share, transportation service providers are forced to overload their vehicles simply to compete on an equal footing.

Since 2003, over twenty Chinese provinces have implemented a new toll scheme, known as the *toll-by-weight* scheme, in which expressway tolls are collected based on the weight and traveling distance of the vehicle. The primary aim of the toll-by-weight scheme is not to increase toll revenue, but to introduce a fairer approach of collecting tolls and to effectively discourage overloading. For ease of collection, the government stipulates that the toll-by-weight scheme used in all expressways within a province must

* Corresponding author. Tel.: +852 64117909/+86 13349921096; fax: +852 34420189.

E-mail addresses: zizhang@cityu.edu.hk (Z. Zhang), tigerqin1980@gmail.com (H. Qin), izhuwb.com (W. Zhu), lim.andrew@cityu.edu.hk (A. Lim).

be identical. Weighing machines and electronic displays are installed at the exit gates of the toll stations; when a vehicle passes through a gate, its weight and toll are immediately shown on the electronic display. Commonly, a very high surcharge is imposed on the overloaded vehicles.

Most existing vehicle routing models assume that the cost of traversing each edge in the graph representation of the problem is equivalent to edge length or some constant (Toth and Vigo, 2002); a common objective for such models is to minimize the overall traveling distance of one or more vehicles. However, in practice transportation service providers wish to minimize their total transportation costs rather than the total traveling distance. When the toll-by-weight scheme is imposed, minimizing cost and distance are two different objectives, which makes classical vehicle routing models inapplicable.

This study addresses a new vehicle routing problem that takes a single vehicle and its weight into account. We call this problem the single vehicle routing problem with toll-by-weight scheme (SVRPTBW); this problem is directly applicable to the increasing number of Chinese provinces that are imposing toll-by-weight schemes on their expressway systems. Additionally, in regions with toll-free expressways, this model is also applicable when minimizing fuel consumption is the objective, because the fuel consumption of a vehicle is approximately proportional to its weight.

The remainder of the paper is organized as follows. Section 2 precisely defines the SVRPTBW, and also shows how it relates to some of the problems in existing literature. We then describe our branch-and-bound approach in Section 3, including details on how we generate a solution for the initial upper bound, the valid lower bound as well as a proof of its validity, and our dominance checkers. Our computational experiments are reported in Section 4, where we use our algorithm to solve both existing (adapted) TSP test instances and new instances based on actual expressway information in China. We conclude our study in Section 5 with some remarks on further directions of research.

2. Problem description and literature

The SVRPTBW is defined on a complete and undirected graph $G=(V,E)$, where $V=\{0,1,\dots,n,n+1\}$ is the vertex set and $E=\{e_{ij}=(i,j): i,j \in V, i \neq j\}$ is the edge set. Vertices 0 and $n+1$ represent the exit from and the entrance to the depot, respectively. We denote the vertices representing the set of n customers by $V_C=\{1,\dots,n\}$. Each customer i has a non-negative weight demand q_i ($q_0=q_{n+1}=0$) to be delivered from the depot, and each edge e_{ij} has a non-negative distance d_{ij} , where the distance matrix $[d_{ij}]$ satisfies the triangle inequality rule.

A vehicle with unladen weight \bar{Q} and unlimited capacity is loaded with goods that satisfy all customer demands weighing $\sum_{i=1}^n q_i$ at the depot, and then successively visits each vertex exactly once. Upon arriving at vertex $i \in V_C$, the vehicle's weight is decreased by q_i to fulfill customer demand. If the vehicle travels from vertex i to j , its weight during this traversal is denoted by w_{ij} ; otherwise, $w_{ij}=0$. The toll function $f(w)$, where w is the vehicle's weight, is used to calculate the transportation cost per unit distance and is applied to all graph edges. The traversal cost from vertex i to j paid by the vehicle with weight w_{ij} is given by $d_{ij} \times f(w_{ij})$. The objective of the problem is to find a Hamiltonian path on G , starting from vertex 0 and ending at vertex $n+1$, that minimizes the total transportation cost of the vehicle.

Let $R=(r_0, r_1, \dots, r_n, r_{n+1})$ be an SVRPTBW route, where $r_0=0$ and $r_{n+1}=n+1$. We denote by d_{ij}^R and w_{ij}^R the distance and vehicle weight associated with the movement from vertex r_i to r_j , respectively; we also use the shorthands d_i^R and w_i^R to represent $d_{i,i+1}^R$ and $w_{i,i+1}^R$ respectively. The weight demanded at vertex r_i is given

by q_i^R . The cost of traversing route R incurred by the vehicle with initial weight $Q_0 = \bar{Q} + \sum_{i=1}^n q_i$ can be calculated by:

$$z(R) = \sum_{i=0}^n d_i^R \cdot f(w_i^R) = \sum_{i=0}^n d_i^R \cdot f\left(\bar{Q} + \sum_{j=i+1}^{n+1} q_j^R\right) \quad (1)$$

For example, if $f(w) = 0.08w$, the total transportation cost of route R shown in Fig. 1 is:

$$z(R) = 2f(15) + 6f(11) + 2f(9) + 2f(8) + 2f(5) = 11.2$$

Fig. 2 gives two typical toll functions (or schemes) in which the unit of $f(w)$ is Chinese Yuan (RMB) per kilometer and the weight unit is tonne. As shown in Fig. 2(a), the tolls in the Gansu province are charged directly proportionally to the vehicle's weight; the toll per tonne per kilometer is fixed at 0.08 RMB. The Jiangxi province uses a relatively more complex toll-by-weight scheme (see Fig. 2(b)), which is given by the following non-linear piecewise function:

$$f(w) = \begin{cases} 0 & \text{if } w = 0 \\ 0.4 & \text{if } 0 < w \leq 5 \\ 0.08w & \text{if } 5 < w \leq 10 \\ -0.0005w^2 + 0.07w + 0.15 & \text{if } 10 < w \leq 40 \\ 2.15 & \text{if } w > 40 \end{cases}$$

Consider the case where the toll function $f(w)$ is linear, i.e., $f(w) = \alpha w + \beta$, where α and β are non-negative constants. If $\alpha = 0$ and $\beta > 0$, then the SVRPTBW reduces to the well-studied traveling salesman problem (TSP) (Gutin and Punnen, 2002). If $\alpha > 0$ and $\beta \geq 0$, we show that the SVRPTBW is equivalent to the minimum latency problem (MLP) (Blum et al., 1994; Goemans and Kleinberg, 1998; Arora and Karakostas, 2003; Wu et al., 2004; Archer et al., 2008), which is also termed the traveling repairman problem (García et al., 2002; Salehipour et al., 2008), the traveling deliveryman problem (Minieka, 1989; Fischetti et al., 1993; Méndez-Díaz et al., 2008) and the cumulative traveling salesman problem (Bianco et al., 1993). The MLP is a variant of the TSP in which the objective is to minimize the sum of the first arrival times at all vertices; the first arrival time at a vertex, also called the latency, is the total distance covered prior to reaching the vertex.

The transformation from the SVRPTBW to the MLP is as follows. By expression (1), for any route R , we have:

$$\begin{aligned} z(R) &= \sum_{i=0}^n d_i^R \cdot \left(\alpha \cdot \left(\bar{Q} + \sum_{j=i+1}^{n+1} q_j^R \right) + \beta \right) \\ &= (\alpha \bar{Q} + \beta) \sum_{i=0}^n d_i^R + \alpha \sum_{i=0}^n d_i^R \sum_{j=i+1}^{n+1} q_j^R \\ &= (\alpha \bar{Q} + \beta) \sum_{i=0}^n d_i^R + \alpha \sum_{j=1}^n q_j^R \sum_{i=0}^{j-1} d_i^R \\ &= \left(\alpha \bar{Q} + \beta + \alpha q_{n+1}^R \right) \sum_{i=0}^n d_i^R + \sum_{j=1}^n \alpha q_j^R \sum_{i=0}^{j-1} d_i^R \end{aligned} \quad (2)$$

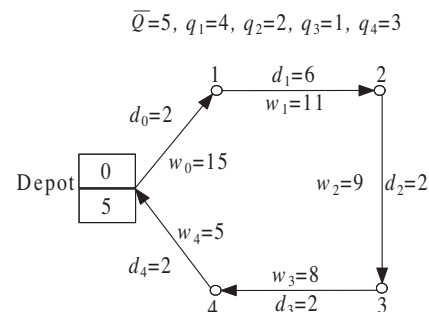


Fig. 1. An example of calculating total transportation cost.

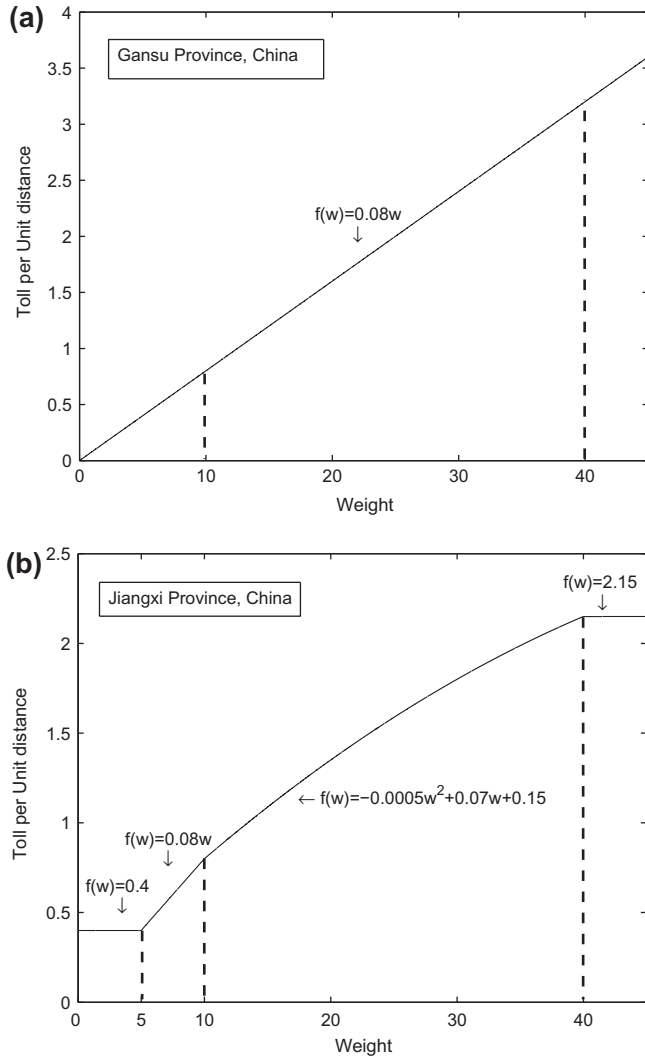


Fig. 2. (a) Linear toll-by-weight function. (b) Nonlinear piecewise toll-by-weight function.

Let $q_i^R = \alpha q_i^R$ for $0 \leq i \leq n$ and $q_{n+1}^R = \alpha \bar{Q} + \beta + \alpha q_{n+1}^R$. We can rewrite $z(R)$ as:

$$z(R) = \sum_{j=1}^{n+1} q_j^R \sum_{i=0}^{j-1} d_i^R \quad (3)$$

Hence, $z(R)$ can be viewed as the total weighted latency of route R , where q_j^R is the weight of vertex r_j ; the SVRPTBW with such a toll function is equivalent to identifying a route with the minimum weighted latency. As stated by Archer et al. (2008), the weighted MLP can be transformed into the unweighted MLP by substituting λ vertices joined by zero-length edges for each vertex with weight λ .

To date, all Chinese provinces that have implemented toll-by-weight schemes have adopted monotonically increasing toll functions; our paper focuses on developing a branch-and-bound (B&B) algorithm that is able to produce optimal solutions for SVRPTBW instances with any monotonically increasing toll function $f(w)$. At present, there are only two articles that study the toll-by-weight scheme in the vehicle routing problem in existing literature (Chen et al., 2007; Shen et al., 2009); both apply heuristics to solve their models.

When solving SVRPTBW cases where the toll function is linear, we can transform them into the TSP or the MLP. As reported by

Applegate et al. (2006), the largest solved TSP instance consists of a tour of 85,900 cities. Three types of solution approaches have been applied to the MLP, namely exact algorithms (Minieka, 1989; Bianco et al., 1993; Fischetti et al., 1993; García et al., 2002; Wu et al., 2004; Méndez-Díaz et al., 2008), approximation algorithms (Blum et al., 1994; Goemans and Kleinberg, 1998; Arora and Karakostas, 2003; Archer et al., 2008) and meta-heuristics (Salehipour et al., 2008). Compared with the TSP, the size of the largest MLP instance that has been solved exactly is much smaller, containing only 60 randomly generated vertices (Fischetti et al., 1993).

To the best of our knowledge, there are no existing papers that address the SVRPTBW with non-linear toll functions at the time of writing.

3. Branch-and-bound algorithm

Branch-and-bound (B&B) is a common technique for finding optimal solutions of various combinatorial optimization problems. For the rest of this discussion, we distinguish between the terms *node* and *vertex*, which are usually considered the same and are used interchangeably in standard graph terminology; we specify that *node* refers to the B&B search tree node, and *vertex* refers to the vertex in a graph.

Our B&B technique first finds an initial upper bound by generating an SVRPTBW route using a simulated annealing algorithm; this upper bound is updated over the course of the B&B search process. We then explore the search space of possible routes according to a depth-first policy. At each node, we calculate a feasible lower bound value, and the node is pruned if it is greater than or equal to the current upper bound. We also make use of a dominance rule, and the node is similarly pruned if it is found to be strictly dominated by another node.

In this section, we describe the algorithm used to generate our initial upper bound, our feasible lower bound as well as a proof of its correctness and our dominance rule along with the set of five subroutines that make use of this rule. For ease of exposition, we introduce some additional notations as follows. For a complete graph $G = (V, E)$ and toll function $f(x)$, we denote an instance of the SVRPTBW by the quadruple (x, y, V, Q) , where $x \in V$ and $y \in V$ are the starting and ending vertices respectively; $V \subseteq V$ is the vertex set of the instance; and Q is the initial weight of the vehicle. Hence, this quadruple describes an SVRPTBW instance on a complete subgraph of G . Correspondingly, the cost of the optimal solution to this instance is denoted by $z(x, y, V, Q)$. In order to fulfill all customer demands, Q must be greater than or equal to $\bar{Q} + \sum_{i \in V'} q_i$; in the case of $Q > \bar{Q} + \sum_{i \in V'} q_i$, we can transform the problem into the standard form described in Section 2 by setting $\bar{Q} = Q - \sum_{i \in V'} q_i$.

We also define $F(R, Q)$ as the traversal cost of route R (which may be a partial route) incurred by a vehicle with initial weight Q . Note that if $Q - \bar{Q}$ cannot satisfy the demands of customers along R , then $F(R, Q) = +\infty$; if R is a complete route on $(0, n+1, V, Q)$ where $Q = \bar{Q} + \sum_{i=1}^n q_i$, then $F(R, Q) = z(R)$.

3.1. Upper bound

The initial upper bound to the problem is generated by a standard simulated annealing (SA) algorithm; this upper bound is updated during the B&B search process. In our SA algorithm, three conventional local search operations are employed (see Fig. 3):

- 2-opt** remove two edges (r_{i-1}, r_i) and (r_j, r_{j+1}) in the route, replace them with (r_{i-1}, r_{j+1}) and (r_i, r_j) , and reverse the path (r_i, \dots, r_j)
- shift** move one non-null route segment to another position in the route

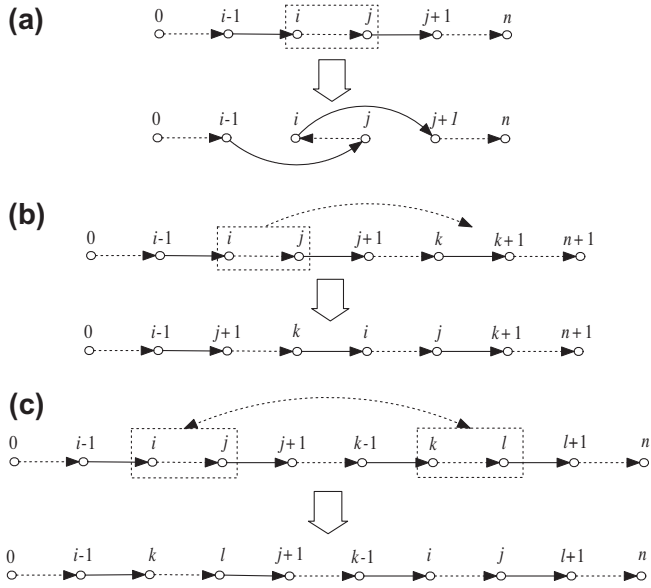


Fig. 3. (a) The 2-opt operation. (b) The shift operation. (c) The exchange operation.

exchange exchange the positions of two non-null and non-overlapping route segments

The initial temperature is set to T_0 . In each iteration of the SA algorithm, we randomly perform one of the three search operations to generate a new route. If N consecutive non-improving iterations occur, we replace the current temperature T with $T \times c$, where c ($0 < c < 1$) is the cooling ratio. Our values for the parameters are as follows: $T_0 = 600$, $T_c = 0.001$, $c = 0.99$ and $N = 2000$.

3.2. Lower bound

The quality of the lower bound has a significant effect on the performance of a B&B algorithm. In our approach, we employ a lower bound measure for the SVRPTBW that relies on the monotonically increasing nature of the toll function, as well as certain properties of minimum spanning trees (MST). In this section, we first prove two lemmas. Then, we describe our lower bound measure and show its validity using these lemmas.

Lemma 1. Let $(\hat{d}_0^R, \dots, \hat{d}_n^R)$ be non-decreasingly sorted edge lengths of any SVRPTBW route R , i.e., $\hat{d}_i^R \leq \hat{d}_{i+1}^R$ for $0 \leq i \leq n-1$. If $f(w)$ is a monotonically increasing toll function, then:

$$z(R) = \sum_{i=0}^n \hat{d}_i^R f(w_i^R) \geq \sum_{i=0}^n \hat{d}_i^R f(w_i^R) \quad (4)$$

Proof. For any route R , $w_i^R \geq w_{i+1}^R$. Since the toll function $f(w)$ is monotonically increasing, therefore $f(w_i^R) \geq f(w_{i+1}^R)$ for $1 \leq i \leq n$. The rearrangement inequality (Hardy et al., 1988) states that:

$$x_1 y_1 + \dots + x_n y_n \geq x_{\pi(1)} y_n + \dots + x_{\pi(n)} y_1 \geq x_1 y_n + \dots + x_n y_1$$

for every choice of real numbers $x_1 \leq \dots \leq x_n$ and $y_1 \leq \dots \leq y_n$ and every permutation $x_{\pi(1)}, \dots, x_{\pi(n)}$ of x_1, \dots, x_n . The direct application of the rearrangement inequality proves this lemma. \square

Lemma 2. Let T be a minimum spanning tree (MST) on a complete and undirected graph $G_M = (V_M, E_M)$, where V_M contains n vertices, and let T' be an arbitrary spanning tree on G_M . Let the sequences $(\hat{e}_1, \dots, \hat{e}_{n-1})$ and $(\hat{e}'_1, \dots, \hat{e}'_{n-1})$ be the edges of T and T' , respectively, sorted in non-decreasing order of length. Then, $|\hat{e}_i| \leq |\hat{e}'_i|$ for $1 \leq i \leq n-1$.

Proof. Let $E(T)$ and $E(T')$ denote the sets of edges of T and T' , respectively. Since T is an MST, the addition of any edge $e'_i \in E(T')$ into T must generate a single cycle $C_i = (e'_i, e_{k_1}, \dots, e_{k_m})$, where $e_{k_j} \in E(T)$, $1 \leq j \leq m$. It can be shown that $|e_{k_j}| \leq |e'_i|$ for $1 \leq j \leq m$, as follows. Assume to the contrary that there exists $e_{k_j} \in C_i - \{e'_i\}$ such that $|e'_i| < |e_{k_j}|$. Then the tree $T - \{e_{k_j}\} + \{e'_i\}$ will have a smaller total length than T , which contradicts the fact that T is an MST.

Construct a bipartite graph $G_B = (X, Y, E_B)$, where each vertex in X corresponds to one edge in $E(T)$, i.e., $X = \{v'_i : e'_i \in E(T')\}$; similarly, each vertex in Y corresponds to one edge in $E(T)$, i.e., $Y = \{v_j : e_j \in E(T)\}$. If $e_j \in E(T)$ is in the cycle associated with $e'_i \in E(T')$, i.e., $e_j \in C_i$, then E_B contains an edge connecting the vertices corresponding to e'_i and e_j . To illustrate this construction process, we consider an example shown in Fig. 4, where $e'_1 = e'_2 = \sqrt{2}/2$, $e'_3 = e'_4 = 1$ and $e_1 = e_2 = e_3 = e_4 = \sqrt{2}/2$. In this example, since (e'_4, e_1, e_4) is a cycle, so e'_4 is connected to e_1 and e_4 in G_B .

Next, we show that there exists a perfect bipartite matching on G_B . By Hall's Theorem (Hall, 1935), it suffices to show that for all $S \subseteq X$, $|S| \leq |N(S)|$, where $N(S)$ is the set of vertices adjacent to the vertices of S in G_B .

Let $V(S)$ and $V(N(S))$ be the vertices of G_M that are associated with S and $N(S)$ respectively; in our example, if $S = \{e'_3\}$, then $V(S) = \{2, 3\}$. We construct a graph $G' = (V', S)$, where G' is a subgraph of G_M and $V' = V(S) \cup V(N(S))$. Continuing the example, given $S = \{e'_3\}$, we have $N(S) = \{e_1, e_2\}$, $V(S) = \{2, 3\}$, $V(N(S)) = \{2, 3, 4\}$, and $G' = (\{2, 3, 4\}, \{(2, 3)\})$ (see Fig. 4(e)).

G' is a forest with $c(G')$ components, each of which is either a subset of S or an isolated vertex. Since S does not contain cycles, we can deduce that $|V'| = |S| + c(G')$ (by Corollary 3.1.7 of Gross and Yellen (2005)). In the same way, we can also construct $G'' = (V', N(S))$ and get $|V'| = |N(S)| + c(G'')$. By the definition of $N(S)$, we find that any pair of vertices in G_M connected by one edge in S must be connected by one or several edges in $N(S)$, and accordingly $c(G') \geq c(G'')$. This implies $|S| \leq |N(S)|$, and therefore there is a perfect bipartite matching on G_B by Hall's Theorem. As a result, each e'_i can be exclusively matched to one edge $e_j \in C_i - \{e'_i\} \subseteq E(T)$, where $|e'_i| \geq |e_j|$.

Finally, we show that $|\hat{e}_i| \leq |\hat{e}'_i|$ for $1 \leq i \leq n-1$. Recall that \hat{e}_i is the i th smallest element in $E(T)$, i.e., $|\hat{e}_i| \geq |\hat{e}_{i-1}| \geq \dots \geq |\hat{e}_1|$. Since each edge \hat{e}'_k ($1 \leq k \leq i$) exclusively corresponds to one edge

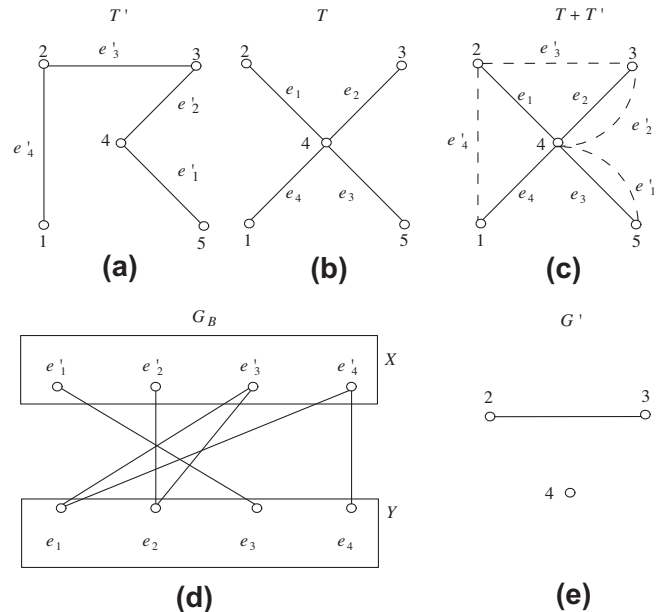


Fig. 4. (a) Arbitrary spanning tree T' . (b) Minimum spanning tree T . (c) $T + T'$, where T is denoted by dashed lines. (d) Bipartite graph G_B . (e) $G' = (\{2, 3, 4\}, \{(2, 3)\})$.

$\hat{e}_j \in E(T)$ whose length is less than or equal to $|\hat{e}'_k|$, we can deduce that \hat{e}'_i is at least larger than or equal to i elements of $E(T)$, which guarantees that $|\hat{e}_i| \leq |\hat{e}'_i|$. \square

We can now derive a lower bound for the SVRPTBW. For a problem instance denoted by the graph G , find an MST T of $V_c = \{1, \dots, n\}$. Then, construct a spanning tree T of G by connecting vertices 0 and $n+1$ to their nearest neighbors in T . Both vertices 0 and $n+1$ have a degree of one; we denote the lengths of the edges incident to vertices 0 and $n+1$ by d_0 and d_n respectively. Let $(\hat{d}_1, \dots, \hat{d}_{n-1})$ be the lengths of the edges of T sorted in ascending order, and let $(\hat{q}_1, \dots, \hat{q}_n)$ be the weights of the goods demanded by the customers sorted in descending order.

Theorem 1. A polynomially computable lower bound for the SVRPTBW is:

$$LB = d_0 f\left(\sum_{i=1}^n q_i + \bar{Q}\right) + \sum_{i=1}^{n-1} \hat{d}_i f\left(\sum_{j=i+1}^n \hat{q}_j + \bar{Q}\right) + d_n f(\bar{Q}) \quad (5)$$

Proof. Suppose $R = (r_0, r_1, \dots, r_n, r_{n+1})$ is an optimal SVRPTBW route on G . With the exception of d_0^R and d_n^R , sort all edge lengths of R in ascending order, generating a sequence $(\hat{d}_1^R, \hat{d}_2^R, \dots, \hat{d}_{n-1}^R)$. By Lemma 1, we have:

$$z(R) = \sum_{i=0}^n d_i^R f(w_i^R) \geq d_0^R f(w_0^R) + \sum_{i=1}^{n-1} \hat{d}_i^R f(w_i^R) + d_n^R f(w_n^R)$$

By the construction process of T , we know that $d_0^R \geq d_0$ and $d_n^R \geq d_n$. Since the subroute (r_1, r_2, \dots, r_n) is a spanning tree covering vertices $\{1, 2, \dots, n\}$, by Lemma 2 we have $\hat{d}_i^R \geq \hat{d}_i$ for $1 \leq i \leq n-1$. Subsequently, we can deduce that:

$$z(R) \geq d_0 f(w_0^R) + \sum_{i=1}^{n-1} \hat{d}_i f(w_i^R) + d_n f(w_n^R) \quad (6)$$

It is apparent that $w_0^R = \sum_{i=1}^n q_i + \bar{Q}$, $w_n^R = \bar{Q}$ and $w_i^R = \sum_{j=i+1}^n q_j + \bar{Q} \geq \sum_{j=i+1}^n \hat{q}_j + \bar{Q}$. Since $f(w)$ is monotonically increasing, $f(\sum_{j=i+1}^n q_j + \bar{Q}) \geq f(\sum_{j=i+1}^n \hat{q}_j + \bar{Q})$. By substituting these equations into (6), we can derive the lower bound. \square

Given a non-leaf node u at level k of the tree, the path from the root to node u is a partial route of the SVRPTBW, represented by $R_u = (r_0^u, \dots, r_{k-1}^u, r_k^u)$, where $r_0^u = 0$ and $r_k^u = u$. Let P_u be the subproblem starting from vertex r_k^u and ending at $n+1$, having already traveled along subroute R_u , i.e., $P_u = (r_k^u, n+1, V - \{i : i \in R_u - \{r_k^u\}\}, Q_0 - \sum_{i \in R_u} q_i)$. The lower bound LB_u associated with node u is the sum of $F(R_u, Q_0)$ and the lower bound of P_u calculated by expression (5).

The branching rule of our algorithm works as follows. Assume that node u is the selected node for further branching or pruning. If LB_u is greater than or equal to the current upper bound, then node u is pruned and the search process selects the unexplored sibling of node u or the unexplored sibling of the parent of node u (if all siblings of node u have been explored) that has the smallest lower bound. Otherwise, the search process constructs $|V - \{i : i \in R_u - \{r_k^u\}\}|$ branches originating from node u , each corresponding to one child of node u , and then selects the child node with the smallest lower bound.

3.3. Dominance rule

Dominance rules are widely applied in B&B algorithms (Fischetti and Toth, 1988; Bianco et al., 1993; Flidner and Boysen,

2008; Azi et al., 2010) and dynamic programming (Dumas et al., 1995; Mingozzi et al., 1997) for reducing search space; the purpose of dominance rules is to determine when the partial solution represented by a node in the search tree must produce a poorer solution than another node; if so, the node need not be further explored and can be safely pruned.

The dominance rules used in our B&B algorithm are derived on the basis of the following property.

Lemma 3. For two problems $P_1 = (x, y, V_1, Q_1)$ and $P_2 = (x, y, V_2, Q_2)$, if $V_1 \subseteq V_2$, $Q_1 = \sum_{i \in V_1} q_i$ and $Q_2 = \sum_{i \in V_2} q_i$, then $z(x, y, V_1, Q_1) \leq z(x, y, V_2, Q_2)$.

Proof. Suppose by way of contradiction that $z(x, y, V_1, Q_1) > z(x, y, V_2, Q_2)$, and $R' = (x, r'_1, r'_2, \dots, y)$ is an optimal route of P_2 . If the vehicle with initial weight Q_1 travels along R' and does not fulfill the demands of customers in $V_2 - V_1$, then the traversal cost incurred must be less than or equal to $z(x, y, V_2, Q_2)$ because $Q_1 \leq Q_2$ and the toll function is monotonically increasing. This implies that we can find a route with the transportation cost less than $z(x, y, V_1, Q_1)$ for P_1 , which is a contradiction. \square

By Lemma 3, we can easily derive the dominance rule as the following theorem:

Theorem 2 (Dominance rule). Let R_u and R_v be two partial routes that start from vertex 0, visit vertex sets V_u and V_v , and end at the same vertex k . If $V_u \subseteq V_v$ and $F(R_u, Q_0) > F(R_v, Q_0)$, then R_v strictly dominates R_u .

In addition to comparing the lower bound of the node to the upper bound, a tree node can also be eliminated from consideration by the dominance relation. The B&B search process does not prune node u if its lower bound LB_u is less than the current upper bound. At this point, before branching to the next tree level, our algorithm checks if node u is strictly dominated by other nodes; this requires the identification of nodes v with the property that R_u and R_v has the same ending vertex, and the vertex sets covered by these two paths have relationship $V_u \subseteq V_v$.

Various search heuristics can be applied to identify such nodes v ; however, there is a trade-off between the potential amount of pruning brought about by more advanced search heuristics and the added computation time required for the search. After some preliminary experimentation, we chose five simple subroutines that search for node v , which we call *dominance checkers*; node u will be ignored if R_u is strictly dominated by an R_v found by these dominance checkers. For ease of discussion and without loss of generality, we assume $R_u = (0, 1, \dots, k)$ and $V_u = \{0, 1, \dots, k\}$ for the remainder of this section.

3.3.1. Dominance checker 1 (forward shift)

For this checker, partial routes R_v are produced by shifting each vertex i ($1 \leq i \leq k-2$) to the position between vertex $k-1$ and k . Note that it is not necessary to consider shifting i to a position between vertices $i+1$ and $k-1$ because if such a shift can produce a route R_v that strictly dominates R_u , then an ancestor of node u associated with the path from vertex 0 to that position must have already been pruned, and the search process would not have been able to reach node u in the first place.

After the forward shift of one vertex, the calculation of $F(R_v, Q_0)$ requires the recalculation of the traversal cost of each edge between vertices $i-1$ and k , which takes $O(n)$ time; since this operation is performed on $k-2$ vertices, this checker runs in $O(n^2)$ time.

3.3.2. Dominance checker 2 (backward shift)

This checker produces partial routes R_v by shifting vertex $k-1$ backwards to the position between j and $j+1$ for $0 \leq j \leq k-3$. For much the same reason as the forward shift checker, it is unnecessary to consider the shifting of other vertices. For example, given the initial partial route R_u in Fig. 5(a), shifting vertex $k-1$ to the position between $k-3$ and $k-2$ would result in the route R_1 shown in Fig. 5(b).

After satisfying the demand at vertex k , the weight of the vehicle is $Q' = Q_0 - \sum_{i \in V_u} q_i$. It can be seen that the initial vehicle weight of subroute R'_1 is $Q'_1 = Q' + q_k$, and the traversal cost of the subroute $F(R'_1, Q'_1) = d_{k-2,k}f(Q'_1)$. Moreover, $F(R'_1, Q_0)$ has been calculated during the previous level of the B&B tree. We can compute the traversal cost of R_1 by:

$$F(R_1, Q_0) = F(R''_1, Q_0) + F(R'_1, Q'_1) + d_{k-1,k-2}f(Q'_1 + q_{k-2}) + d_{k-3,k-1}f(Q'_1 + q_{k-2} + q_{k-1}) \quad (7)$$

Because $Q'_2 = Q'_1 + q_{k-2}$ and $F(R'_2, Q'_2) = F(R'_1, Q'_1) + d_{k-3,k-2}f(Q'_2)$, the traversal cost of R_2 (see Fig. 5(c)) can be computed using an expression similar to Eq. (7). Observe that with the previously computed cost of route R_i , the traversal cost of route R_{i+1} can be computed in $O(1)$ time. As there are $k-2$ positions for the backward shift of vertex $k-1$, the overall time complexity of this checker is $O(n)$.

3.3.3. Dominance checker 3 (vertex swap)

This checker swaps vertex $k-1$ with each vertex j in $\{1, 2, \dots, k-2\}$. It requires $O(n)$ time to compute the traversal cost of the resultant partial route because the traversal cost of all edges between vertices j and $k-1$ are affected. Hence, the time complexity of this checker is $O(n^2)$.

3.3.4. Dominance checker 4 (2-opt)

For each vertex $i \in \{1, \dots, k-2\}$, this checker works like a 2-opt exchange. First, edges $(i-1, i)$ and $(k-1, k)$ are broken. Then, edges $(i-1, k-1)$ and (i, k) are inserted. Finally, the route segment $(i, i+1, \dots, k-2, k-1)$ is reversed to create the route R_v . After these operations, it is necessary to calculate the traversal cost of edges $(i-1, k-1)$ and (i, k) , as well as the route segment $(k-1, k-2, \dots, i+1, i)$, which requires $O(n)$ time. Therefore, this checker runs in $O(n^2)$ time.

3.3.5. Dominance checker 5 (vertex insert)

This checker inserts each of the vertices $j \in V - V_u - \{n+1\}$ into the partial route R_u . After inserting j at any point in R_u , the weight of the vehicle at k becomes $Q' = Q_0 - \sum_{i \in V_u} q_i - q_j$. Consider the case where j is inserted between vertices $k-1$ and k (see Fig. 6(b)). The traversal cost of R_1 can be calculated as:

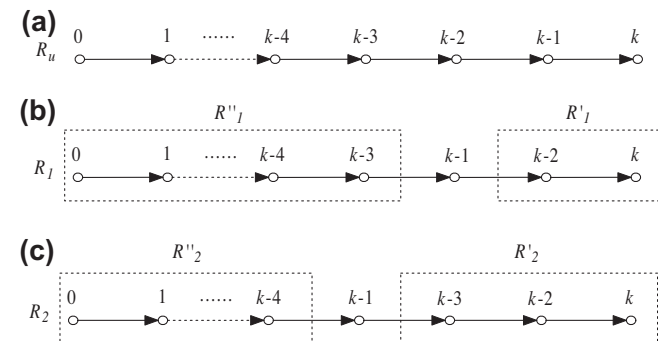


Fig. 5. (a) R_u . (b) R_1 created by backward shift of $k-1$ to the position between $k-3$ and $k-2$. (c) R_2 created by backward shift of $k-1$ to the position between $k-4$ and $k-3$.

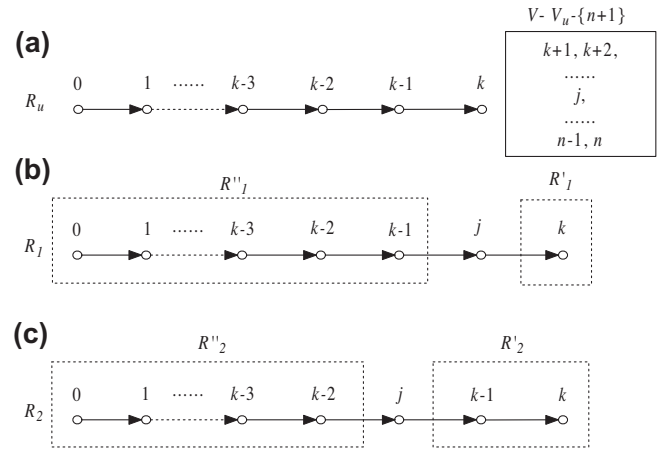


Fig. 6. (a) R_u . (b) R_1 created by inserting j into the position between $k-1$ and k . (c) R_2 created by inserting j into the position between $k-2$ and $k-1$.

$$F(R_1, Q_0) = F(R''_1, Q_0) + d_{k-1,j}f(Q' + q_k + q_j) + d_{j,k}f(Q' + q_k) \quad (8)$$

where $F(R''_1, Q_0)$ has been computed in a previous level. Like for dominance checker 2 (Backward Shift), when the cost of R_i has been precomputed, the traversal cost of R_{i+1} can also be computed in $O(1)$ time; with $k-1$ positions where j can be inserted into R_u , it takes $O(n)$ time to check all positions. Since there are $n-k$ vertices in $V - V_u - \{n+1\}$, the overall time complexity of this checker is $O(n^2)$.

It is theoretically possible to insert more than one vertex into R_u , although the number of combinations of possible insertions increases exponentially as a result. We performed some experiments where more than one vertex is inserted, and found that the amount of computation time increased enormously but the amount of increased pruning was not significant. Therefore, we limited this checker to consider only the insertion of a single vertex.

This dominance checker is invalid for instances that do not satisfy the triangle inequality rule. The minimum cost of all complete SVRPTBW routes containing R_u is $F(R_u, Q_0) + z(k, n+1, \{k, k+1, \dots, n, n+1\}, Q_0 - \sum_{i=0}^{k-1} q_i)$. When vertex j ($j > k$) is inserted into R_u , the minimum cost related to the resultant $R_v = \{0, 1, \dots, j, \dots, k-1, k\}$ is calculated by $F(R_v, Q_0) + z(k, n+1, \{k, k+1, \dots, j-1, j+1, \dots, n, n+1\}, Q_0 - \sum_{i=0}^{k-1} q_i - q_j)$. For an instance that violates the triangle inequality rule, we cannot prove $z(k, n+1, \{k, k+1, \dots, j, \dots, n, n+1\}, Q_0 - \sum_{i=0}^{k-1} q_i) \geq z(k, n+1, \{k, k+1, \dots, j-1, j+1, \dots, n, n+1\}, Q_0 - \sum_{i=0}^{k-1} q_i - q_j)$. Therefore, even when $F(R_v, Q_0) < F(R_u, Q_0)$, we still cannot judge whether R_v strictly dominates R_u . When this dominance checker is disabled, our B&B algorithm is capable of solving instances that do not necessarily satisfy the triangle inequality rule.

4. Experiments and analysis

Our computational experiments are divided into two distinct test sets. The first data set was derived from the TSPLIB library of TSP instances; the second data set was generated based on the real expressway information of the Gansu and Jiangxi provinces in China. For each problem instance, the first vertex was designated as the depot. All instances and results are available in the online supplement to this paper at: www.computational-logistics.org/orlib/tbw.

The algorithm described in this paper was coded in C++, and all experiments were run on a Linux server with 8 GB memory and Intel Xeon (R) 2.66 GHz processor. Computation times reported here are in CPU seconds on this machine.

During preliminary experimentation, it was found that as $|V_u|$ (the cardinality of V_u) grows, the effectiveness of the dominance checkers decreases. Based on our analysis, we modified our algorithm so that when $|V_u| \geq 0.7 \times |V|$, we disable three of the dominance checkers, namely dominance checkers 1 (forward shift), 3 (vertex swap) and 4 (2-opt). When $|V_u| \geq 0.75 \times |V|$, we disable dominance checker 5 (vertex insert). Finally, dominance checker 2 (backward shift) is disabled when $|V_u| \geq 0.8 \times |V|$.

4.1. Modified instances from TSPLIB

Our first data set consists of symmetric TSP instances taken from the standard test suite for the traveling salesman problem TSPLIB (Reinelt, 1991) (available at <http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/tsp/>). Some TSP instances do not satisfy the triangle inequality rule, so we disabled dominance checker 5 when solving the SVRPTW instances modified from such TSP instances.

We modified the TSP instances to become SVRPTBW problems by setting the weight demanded at each vertex to 1 tonne. When $\bar{Q} = 0$ and $f(w) = w$, these instances are equivalent to unweighted MLP instances. Since the MLP is a special case of the SVRPTBW, our algorithm can be used to solve MLP instances. The two best exact algorithms for the unweighted MLP in existing literature were proposed by Wu et al. (2004) and Méndez-Díaz et al. (2008). We compared our B&B algorithm with these techniques using the TSPLIB instances employed by Méndez-Díaz et al. (2008), and present the running times required to find the optimal solutions in Table 1. The instances that violate the triangle inequality rule are marked with an asterisk (*) in this and the following two tables. Since Wu et al. (2004) only reported computational results for *ulysses16*, *ulysses22*, *gr24* and *fri26*, the remaining cells in column 2 are marked “N/A”.

The results show that our algorithm found the optimal solution for the first seven instances in not more than 0.2 s, and for the rest of the instances in less than 4 s. In contrast, the technique by Wu et al. (2004) requires up to 30 s for some instances, while the technique by Méndez-Díaz et al. (2008) requires computing times larger than 1000 seconds for some instances. Although our machine is more powerful, this cannot account for the dramatic difference in speed compared to the two existing techniques; it is reasonable to conclude that our approach is more efficient than the existing best MLP approaches.

We then attempted to solve as many of these unweighted MLP instances from TSPLIB as possible using our B&B approach. Our algorithm was able to solve 15 of the instances within 1 h. The statistics for these instances are given in Table 2. The column *Nodes* gives the total number of explored nodes in the B&B search tree; *LB/Cost* gives the percentage of the initial lower bound (at the root node) calculated by Eq. (5) from the optimal solution value; and

Table 2

Unweighted MLP instances solved by B&B algorithm.

Instance	Cost	Time	Nodes	LB/Cost (%)	UB/Cost (%)
burma14	151.5	0.00	284	76.2	100.0
ulysses16	338.9	0.00	678	65.4	100.0
gr17*	10,845.0	0.07	21,434	75.8	100.0
gr21*	21,096.0	0.02	903	77.1	100.0
ulysses22	452.6	0.03	3,436	62.2	100.0
gr24*	12,292.0	0.16	6,082	82.0	100.0
fri26*	9,664.0	0.17	6,207	84.0	100.0
bayg29*	20,439.0	3.24	87,561	81.8	100.1
bays29*	24,408.0	3.43	86,223	79.0	100.0
dantzig42	11,277.6	7.98	85,298	81.0	100.9
swiss42*	20,905.0	16.83	134,963	82.7	100.0
gr48*	96,744.0	993.21	5,400,258	79.0	104.0
hk48*	234,588.0	1,423.14	9,710,210	80.0	101.2
eil51	9,712.0	1,944.15	12,710,046	87.4	100.4
berlin52	134,852.0	1,433.89	16,703,646	74.0	101.8

Table 3

Weighted MLP instances solved by B&B algorithm.

Instance	Cost	Time	Nodes	LB/Cost (%)	UB/Cost (%)
burma14	138.6	0.00	288	74.1	100.0
ulysses16	331.3	0.00	752	61.3	100.0
gr17*	9,931.0	0.07	20,862	72.3	100.0
gr21*	20,054.5	0.06	3,953	72.2	100.0
ulysses22	467.1	0.05	5,211	55.6	100.0
gr24*	11,748.5	0.31	10,611	77.5	101.2
fri26*	9,421.7	0.36	12,445	80.8	101.9
bayg29*	20,252.9	11.95	321,084	75.4	100.2
bays29*	24,704.5	16.21	434,095	72.6	100.6
dantzig42	11,507.2	34.54	389,123	75.1	101.4
swiss42*	21,709.9	101.64	833,534	76.5	100.7
gr48*	98,714.9	26,239.08	166,542,140	72.7	102.0
hk48*	240,594.2	26,078.89	201,824,403	73.1	103.0
eil51	9,498.2	7,992.44	51,323,043	82.9	102.4
berlin52	136,183.2	5,496.05	67,130,237	66.7	101.9

UB/Cost gives the percentage of the initial upper bound obtained using the SA algorithm from the optimal solution value.

The average initial lower bound over all instances is about 78% of the optimal solution value, which shows that our lower bound measure provides a reasonable approximation of the cost of the optimal solution for this data set. It is interesting to note that the initial upper bound found by our standard SA algorithm is very close to the optimal value (the maximum absolute error is less than 2%); this suggests that SA is an effective technique for the SVRPTBW and can produce a near-optimal solution quickly. As expected, problem difficulty increases sharply with n with a corresponding increase in the time taken and the nodes searched.

Finally, we again modified the TSP instances such that the weight demanded at each vertex is a uniformly randomly gener-

Table 1

Performance comparison on unweighted MLP instances.

Instance	Wu et al. (2004) P4 2.4 GHz + 256 M RAM	Méndez-Díaz et al. (2008) Sun UltraSparc III 1 GHz + 2 GB RAM	Branch-and-bound algorithm Intel Xeon (R) 2.66 GHz + 8 GB RAM
	Time	Time	Time
burma14	N/A	0.61	0.00
ulysses16	0.09	64.11	0.00
gr17*	N/A	22.44	0.07
gr21*	N/A	22.57	0.02
ulysses22	3.40	1,190.91	0.03
gr24*	30.23	18.06	0.16
fri26*	26.09	293.74	0.17
bayg29*	N/A	5,334.55	3.24
bays29*	N/A	1,440.34	3.43

Table 4
Detailed results for Gansu and Jiangxi generated data.

Instance	Gansu Expressway toll scheme (Fig. 2(a))					Jiangxi expressway toll scheme (Fig. 2(b))				
	Cost	Time	Nodes	LB/Cost (%)	UB/Cost (%)	Cost	Time	Nodes	LB/Cost (%)	UB/Cost (%)
gansu25-1	5264.0	1.55	136,255	61.0	101.3	4718.2	2.89	226,122	61.2	100.0
gansu25-2	5353.2	0.47	36,702	54.9	102.2	4704.0	0.63	45,606	56.8	102.1
gansu25-3	4616.8	1.00	70,791	61.9	100.0	4092.3	1.86	122,247	63.1	100.0
gansu25-4	6265.4	1.09	68,781	65.6	100.0	5436.1	1.48	85,494	67.5	100.0
gansu25-5	5126.5	0.12	9069	62.6	100.0	4588.5	0.20	13,262	64.1	100.0
gansu30-1	7224.5	8.97	567,507	61.8	102.2	6105.8	17.76	954,998	63.0	100.2
gansu30-2	7326.2	6.99	357,089	63.7	100.6	6303.6	13.98	680,430	64.9	100.4
gansu30-3	6818.1	4.90	230,000	61.7	100.4	5784.1	8.58	358,615	63.5	100.0
gansu30-4	7179.6	3.85	195,393	61.4	101.1	6210.9	9.36	417,356	62.3	100.4
gansu30-5	6242.8	3.32	169,554	57.4	100.9	5322.5	5.39	245,255	59.4	100.2
gansu35-1	7906.0	15.41	467,532	65.6	103.0	6411.4	26.82	740,155	68.7	100.6
gansu35-2	8478.8	49.50	1,561,205	59.5	102.1	6943.8	127.43	3,646,451	61.8	101.8
gansu35-3	7711.1	70.85	2,299,375	64.7	101.9	6536.0	209.76	6,676,884	65.6	101.3
gansu35-4	7339.0	60.91	1,817,959	59.7	101.8	6067.9	127.72	3,561,418	61.6	102.4
gansu35-5	6689.7	79.75	2,847,785	60.0	103.0	5569.2	180.47	5,918,448	61.8	101.8
gansu40-1	7838.3	71.01	1,719,561	63.0	104.9	6406.5	369.10	8,573,415	63.6	102.2
gansu40-2	9651.2	383.13	8,220,591	63.1	102.1	7725.5	1020.19	20,747,015	65.0	102.3
gansu40-3	8607.8	282.55	6,686,751	61.9	102.2	7005.2	737.91	16,380,111	63.0	101.5
gansu40-4	9257.7	2854.33	62,116,466	62.7	101.8	7198.3	5024.06	107,446,538	65.9	100.5
gansu40-5	10,176.0	1267.53	30,434,344	68.1	103.0	8033.0	4549.15	108,846,017	69.1	101.3
gansu45-1	10,593.4	21,649.52	395,857,260	61.0	103.0	N/A	> 40,000.00	N/A	N/A	N/A
gansu45-2	10,575.1	1341.82	23,110,153	60.7	102.6	8055.0	30,842.27	469,436,556	64.1	102.9
gansu45-3	10,567.7	4079.46	65,983,410	59.5	100.6	N/A	> 40,000.00	N/A	N/A	N/A
gansu45-4	10,453.1	942.79	16,328,648	61.7	105.1	8059.0	24,570.87	380,517,909	63.9	103.4
gansu45-5	9187.4	3633.14	51,204,824	66.1	101.6	7224.5	34,239.86	533,482,431	67.9	101.5
jiangxi25-1	3474.5	0.57	32,768	69.2	100.5	3013.5	0.85	44,010	71.8	100.4
jiangxi25-2	3414.9	0.23	12,189	69.1	100.0	2971.8	0.26	9706	71.5	100.0
jiangxi25-3	2960.5	0.11	5262	73.8	100.0	2623.8	0.21	9057	75.1	100.0
jiangxi25-4	2114.4	0.18	11,387	63.3	100.0	1868.7	0.45	26,328	66.0	100.0
jiangxi25-5	3337.1	0.19	9620	65.6	100.0	2935.6	0.29	13,012	67.4	100.1
jiangxi30-1	4229.3	1.41	45,074	70.0	100.7	3604.4	2.35	70,443	72.4	100.5
jiangxi30-2	4160.6	1.91	45,860	69.6	100.7	3523.1	4.19	111,444	71.1	100.3
jiangxi30-3	4128.3	0.81	21,253	73.3	100.1	3490.3	1.08	26,654	76.0	100.0
jiangxi30-4	4154.7	7.54	237,599	62.5	102.0	3540.2	16.06	450,011	64.6	101.9
jiangxi30-5	4161.7	2.49	77,330	64.2	100.3	3489.8	3.84	106,744	67.0	100.3
jiangxi35-1	4612.3	1.57	35,835	70.2	101.0	3793.2	4.06	87,970	72.7	100.2
jiangxi35-2	4788.4	9.83	183,620	70.3	100.0	3933.8	17.86	304,084	72.3	100.2
jiangxi35-3	4468.6	10.99	223,477	68.3	101.1	3623.1	20.24	392,798	71.3	100.9
jiangxi35-4	4952.6	56.21	1,155,540	66.6	100.9	4039.6	127.86	2,484,916	69.0	100.5
jiangxi35-5	4724.5	2.62	43,830	74.0	100.9	3882.7	5.78	88,149	76.3	100.4
jiangxi40-1	5852.4	181.04	3,028,776	68.0	102.6	4629.6	555.22	8,494,720	70.9	101.5
jiangxi40-2	5150.4	24.95	349,536	68.9	103.1	4123.4	59.49	896,078	71.2	102.3
jiangxi40-3	5385.1	87.07	1,496,751	69.2	103.1	4307.8	222.58	3,628,112	71.8	101.8
jiangxi40-4	4767.9	68.38	1,141,590	62.1	100.3	3828.3	228.62	4,104,835	64.8	100.1
jiangxi40-5	5330.0	232.47	4,605,134	61.9	103.1	4285.6	745.22	17,519,049	64.4	102.5
jiangxi45-1	6021.5	1255.52	15,035,163	67.3	102.7	4558.6	17,375.94	183,187,608	70.4	102.7
jiangxi45-2	6385.5	1304.99	15,949,805	69.9	101.7	4843.9	7565.08	78,092,802	73.7	101.1
jiangxi45-3	5884.6	492.92	6,428,149	69.0	103.5	4594.5	10,984.19	147,540,859	71.3	103.5
jiangxi45-4	6160.4	513.07	7,296,658	65.5	102.5	4709.7	8938.38	103,251,920	69.2	103.0
jiangxi45-5	6480.9	590.35	6,872,015	71.2	104.6	5003.1	4993.45	52,972,654	73.6	103.0

ated value in the range [0.8,1.2]; this converts the data set into weighted MLP instances. There are no existing reported results on algorithms that can find optimal solutions for the weighted MLP on a general graph, although García et al. (2002) do propose an exact algorithm for the weighted MLP when the underlying graph is a path. Once again, we applied our algorithm to these weighted MLP instances and attempted to solve as many instances as possible; the results are reported in Table 3.

For almost all instances, our B&B approach searched more nodes and took more time for the weighted version compared to the unweighted version. Similarly, both the initial upper and lower bound estimates were less accurate for the weighted MLP, even though the underlying data is otherwise identical. This suggests that having different weight demands for each customer significantly increases the difficulty of the problem when using a

branch-and-bound approach. Nonetheless, our approach was still able to solve 15 instances with up to 52 vertices within 8 h.

4.2. Generated instances based on Gansu and Jiangxi

Our second data set is constructed based on two Chinese provinces, Gansu (52 cities) and Jiangxi (99 cities), and the real information of the Chinese expressway network. Recently, Google launched an online service that allows users to retrieve the real traffic information between any two cities of China using the Google Map API (available at: <http://code.google.com/apis/maps/>). By using a JavaScript program that integrates the Google Map API, we collected the shortest expressway distance between every pair of cities within each province, and generated two complete graphs representing the two provinces based on this information.

Table 5

Summarized results for Gansu and Jiangxi generated data.

Instance	Gansu expressway toll scheme				Jiangxi expressway toll scheme			
	Time	Nodes	LB/Cost (%)	UB/Cost (%)	Time	Nodes	LB/Cost (%)	UB/Cost (%)
gansu-25	0.85	64,319.6	61.2	100.7	1.41	98,546.2	62.5	100.4
gansu-30	5.61	303,908.6	61.2	101.0	11.01	531,330.8	62.6	100.2
gansu-35	55.28	1,798,771.2	61.9	102.4	134.44	4,108,671.2	63.9	101.6
gansu-40	971.71	21,835,542.6	63.8	102.8	2340.08	52,398,619.2	65.3	101.6
gansu-45	6329.35	110,496,859.0	61.8	102.6	29,884.33	461,145,632.0	65.3	102.6
jiangxi-25	0.26	14,245.2	68.2	100.1	0.41	20,422.6	70.3	100.1
jiangxi-30	2.83	85,423.2	67.9	100.7	5.50	153,059.2	70.2	100.6
jiangxi-35	16.24	328,460.4	69.9	100.8	35.16	671,583.4	72.3	100.4
jiangxi-40	118.78	2,124,357.4	66.0	102.5	362.23	6,928,558.8	68.6	101.6
jiangxi-45	831.37	10,316,358.0	68.6	103.0	9971.41	113,009,168.6	71.6	102.7

Table 6

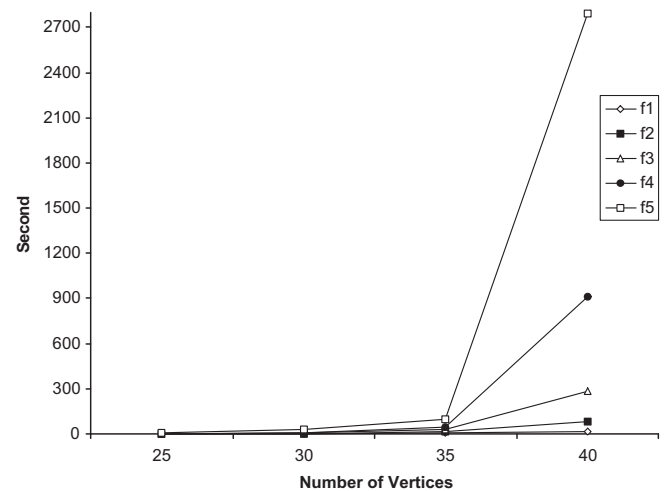
The contributions of the dominance checkers.

Instance	LB	LB + DC1	LB + DC2	LB + DC3	LB + DC4	LB + DC5	LB + DC1-5
Gansu-25-1	1455.17	58.48	62.23	54.35	25.72	90.59	2.89
Gansu-25-2	1039.93	24.21	39.80	27.33	15.88	16.42	0.63
Gansu-25-3	669.87	30.84	33.12	32.08	18.49	31.63	1.86
Gansu-25-4	178.40	11.99	11.21	11.45	4.64	22.34	1.48
Gansu-25-5	113.03	4.27	5.56	4.28	2.25	3.88	0.20
Average	691.28	25.96	30.38	25.90	13.40	32.97	1.41
Jiangxi-25-1	22.41	3.29	3.32	3.49	2.58	4.53	0.85
Jiangxi-25-2	6.63	0.99	1.04	0.95	0.68	1.66	0.26
Jiangxi-25-3	6.84	0.77	0.82	0.75	0.60	1.91	0.21
Jiangxi-25-4	153.28	5.31	6.59	5.36	3.63	16.79	0.45
Jiangxi-25-5	13.64	1.85	2.17	1.93	1.54	1.88	0.29
Average	40.56	2.44	2.79	2.50	1.81	5.35	0.41

From the city set of each province, five subsets are randomly selected for each value of $n + 1 = 25, 30, 35, 40$ and 45 , for a total of 50 test instances; all these instances satisfy the triangle inequality rule. The weight demanded at each city is also uniformly and randomly chosen from the interval $[0.8, 1.2]$ tonnes, and the unladen weight of the vehicle \bar{Q} is set to 5 tonnes. We selected these values because most provinces in China (including Gansu and Jiangxi) impose a heavy monetary penalty on vehicles that exceed a laden weight of 50 tonnes, and as a result it is not worthwhile for freight companies to load their vehicles beyond this limit; our chosen values will generate instances where the optimal solution will be unlikely to exceed this 50-tonne limit by a large margin. We ran our algorithm on these instances under both toll schemes shown in Fig. 2.

The results of these experiments are given in Table 4, where each row gives the statistics for an individual instance, and every set of 5 instances contains the same number of vertices. Our B&B technique was able to solve all but two instances within 40,000 seconds; both unsolved instances have 45 vertices and were generated from the Gansu map under the Jiangxi scheme.

For better analysis, we grouped these results by province and number of vertices as given in Table 5. Each row represents a set of five instances with the same number of vertices for the given toll scheme, and each value is the average over these five instances; the exception is the row *Gansu-45* under the Jiangxi scheme, which averages the values of only the three solved instances. The results show that the amount of time required to solve the same instance under the Gansu toll function (which is linear) is significantly less than under the Jiangxi toll function (which is piecewise non-linear), with a corresponding increase in the number of nodes searched; this implies that a piecewise non-linear toll function increases the difficulty of the problem under our B&B algorithm by reducing the opportunities for pruning. Our initial lower bound estimate is about 60–70% of the optimal solution. Finally, the average initial

**Fig. 7.** The effect of toll functions.

upper bound found by the SA algorithm is again very close to the optimal value, with a maximum absolute error of less than 3%.

Table 6 shows the marginal contribution of each dominance checker over the ten 25-city instances where the Gansu and Jiangxi instances are under the Gansu and Jiangxi toll schemes, respectively; all values in this table are computation times. Column *LB* gives the computation times of our B&B algorithm without any dominance checker (i.e., a basic branch-and-bound algorithm); the subsequent columns show the computation times obtained by adding one dominance checker at a time to the algorithm; the final column repeats the information given in Table 4 for the complete algorithm for easy comparison. It can be seen that the individual application of every dominance checker significantly

decreases the running time compared to the basic branch-and-bound algorithm, but the best performance is achieved by the inclusion of all five checkers.

From the above experiments, it can be seen that the nature of the toll function has an appreciable effect on the performance of our B&B algorithm. To gain a better understanding of the effects of the toll functions, we solved the instances under the following five toll functions: $f_1(w) = w^2$, $f_2(w) = w$, $f_3(w) = \sqrt{w}$, $f_4(w) = \log(w)$ and $f_5(w) = 1$. For each toll function, we averaged the computation times of all instances with the same number of vertices; the results are graphically depicted in Fig. 7. It is apparent that as the steepness of the toll function increases, so does the performance of our B&B algorithm. This is logical since a steeper toll function implies that the optimal solution would be closer to a “greedy” solution where the greatest weight is unloaded as early as possible in the route; as a result, routes that delay the unloading of goods are unlikely to be optimal and are quickly pruned. Furthermore, under a steeper function, the average computation time increases less quickly in relation to the number of vertices. For example, under $f_1(w) = w^2$ the ratio of the average computation time for 40-vertex instances compared to the 35-vertex instances is 2.2, while under $f_5(w) = 1$ the ratio is 28.9. This is a likely explanation of the drop in performance of our algorithm under the Jiangxi toll scheme compared to the Gansu scheme.

5. Conclusions

Motivated by the toll-by-weight schemes implemented in over twenty Chinese provinces, we propose a single vehicle routing problem in which the transportation cost per unit distance monotonically increases with the vehicle's weight. This problem is a new variant of the traditional traveling salesman problem, and is a generalization of both the unweighted and weighted minimum latency problem. The branch-and-bound algorithm described in this paper is the first exact algorithm for the problem and can be implemented on instances with any monotonically increasing toll function. Experiments show that our algorithm outperforms the currently best known exact algorithms for the unweighted MLP and was able to find the optimal solution for an instance with 52 vertices; is also capable of solving weighted MLP instances of up to 52 vertices; and can find the optimal solutions for several SVRPTBW instances with up to 45 vertices.

The SVRPTBW is a good starting point for the investigation of the toll-by-weight scheme that is being used on Chinese expressways, and will continue to be used in the foreseeable future. The model does simplify certain aspects of the problem, and more complex models that consider additional factors can be fruitful avenues for future research. Possibilities include the multiple vehicle routing version of the problem, consideration of vehicle capacity, pick-up as well as delivery, and time windows for delivery.

References

- Applegate, D.L., Bixby, R.E., Chvátal, V., Cook, W.J., 2006. *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, Princeton, New Jersey.
- Archer, A., Levin, A., Williamson, D.P., 2008. A faster, better approximation algorithm for the minimum latency problem. *SIAM Journal on Computing* 37 (5), 1472–1498.
- Arora, S., Karakostas, G., 2003. Approximation schemes for minimum latency problems. *SIAM Journal on Computing* 32 (5), 1317–1337.
- Azi, N., Gendreau, M., Potvin, J.-Y., 2010. An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles. *European Journal of Operational Research* 202 (3), 756–763.
- Bianco, L., Mingozzi, A., Ricciardelli, S., 1993. The traveling salesman problem with cumulative costs. *Networks* 23 (2), 81–91.
- Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan, P., Sudan, M., 1994. The minimum latency problem. In: *Proceedings of the 26th Annual ACM Symposium on Theory of Computing (STOC'94)*, Montréal, Québec, Canada, pp. 163–171.
- Chen, X., Li, J., Yang, F., Wu, Y.H., Ding, L.B., 2007. Design and implementation of vehicle routing optimization system based on toll-by-weight. *Manufacture Information Engineering of China* 17, 69–72.
- Dumas, Y., Desrosiers, J., Gelinas, E., Solomon, M.M., 1995. An optimal algorithm for the traveling salesman problem with time windows. *Operations Research* 43 (2), 367–371.
- Fischetti, M., Laporte, G., Martello, S., 1993. The delivery man problem and cumulative matroids. *Operations Research* 41 (6), 1055–1064.
- Fischetti, M., Toth, P., 1988. A new dominance procedure for combinatorial optimization problems. *Operations Research Letters* 7 (4), 181–187.
- Fliedner, M., Boysen, N., 2008. Solving the car sequencing problem via branch & bound. *European Journal of Operational Research* 191 (3), 1023–1042.
- García, A., Jodrá, P., Tejel, J., 2002. A note on the traveling repairman problem. *Networks* 40 (1), 27–31.
- Goemans, M., Kleinberg, J., 1998. An improved approximation ratio for the minimum latency problem. *Mathematical Programming* 82 (1–2), 111–124.
- Gross, J.L., Yellen, J. (Eds.), 2005. *Graph Theory and Its Applications*. Chapman and Hall/CRC.
- Gutin, G., Punnen, A.P. (Eds.), 2002. *The Traveling Salesman Problem and Its Variations*. Kluwer Academic Publishers, Dordrecht, Netherlands.
- Hall, P., 1935. On representatives of subsets. *Journal of the London Mathematical Society* 10, 26–30.
- Hardy, G., Littlewood, J., Pólya, G., 1988. *Inequalities*. Cambridge University Press, The Edinburgh Building, Cambridge, UK.
- Méndez-Díaz, I., Zabala, P., Lucena, A., 2008. A new formulation for the traveling deliveryman problem. *Discrete Applied Mathematics* 156 (17), 3223–3237.
- Mingozzi, A., Bianco, L., Ricciardelli, S., 1997. Dynamic programming strategies for the traveling salesman problem with time window and precedence constraints. *Operations Research* 45 (3), 365–377.
- Minieka, E., 1989. The delivery man problem on a tree network. *Annals of Operations Research* 18 (1–4), 261–266.
- Reinelt, G., 1991. TSPLIB — traveling salesman problem library. *ORSA Journal of Computing* 3 (4), 376–384.
- Salehipour, A., Sörensen, K., Goos, P., Bräysy, O., 2008. An efficient GRASP+VND metaheuristic for the traveling repairman problem. Working Papers, University of Antwerp, Faculty of Applied Economics. URL <<http://econpapers.repec.org/RePEc:ant:wpaper:2008008>>
- Shen, C.H., Qin, H., Lim, A., 2009. A capacitated vehicle routing problem with toll-by-weight rule. In: Chien, B.C., Hong, T.P. (Eds.), *Opportunities and Challenges for Next-Generation Applied Intelligence, Studies in Computational Intelligence*, vol. 214. Springer, Berlin, Heidelberg, pp. 311–316.
- Toth, P., Vigo, D. (Eds.), 2002. *The Vehicle Routing Problem*. SIAM, Philadelphia, PA.
- Wikipedia, 2011. Expressways of China. <http://en.wikipedia.org/wiki/Expressways_of_China>
- Wu, B.Y., Huang, Z.N., Zhan, F.J., 2004. Exact algorithms for the minimum latency problem. *Information Processing Letters* 92 (6), 303–309.