Discrete Applied Mathematics 160 (2012) 306–320

Contents lists available at SciVerse ScienceDirect

Discrete Applied Mathematics

journal homepage: www.elsevier.com/locate/dam

Approximation results for a min–max location-routing problem

Zhou Xu [a,1], Dongsheng Xu [b,*], Wenbin Zhu [a]

[a] Department of Logistics and Maritime Studies, Faculty of Business, Hong Kong Polytechnic University, Hong Kong

[b] Department of Management Science, Business School, Sun Yat-sen University, Guangzhou 510275, China

ARTICLE INFO

Article history:
Received 3 May 2009
Received in revised form 16 July 2011
Accepted 23 September 2011
Available online 28 October 2011

Keywords:
Approximation algorithm
Approximation hardness
Min–max location-routing

ABSTRACT

This paper studies a min–max location-routing problem, which aims to determine both the home depots and the tours for a set of vehicles to service all the customers in a given weighted graph, so that the maximum working time of the vehicles is minimized. The min–max objective is motivated by the needs of balancing or fairness in vehicle routing applications. We have proved that unless $\mathbb{NP} = \mathbb{P}$, it is impossible for the problem to have an approximation algorithm that achieves an approximation ratio of less than 4/3. Thus, we have developed the first constant ratio approximation algorithm for the problem. Moreover, we have developed new approximation algorithms for several variants, which improve the existing best approximation ratios in the previous literature.

1. Introduction

Consider a complete undirected graph $G = (V, E)$ with a vertex set V and an edge set E. Each $e \in E$ has a symmetric edge weight $w(e)$, which is a non-negative integer, to indicate the traveling time. Consider a set of customer locations $J \subseteq V$, and p identical vehicles. Given the home depots of the vehicles, the classical min–max p-traveling salesmen problem (p-TSP) [7,3,2] is to determine the tours for the p vehicles to service all the customer locations in J, such that each vehicle starts from and returns to its home depot, with the maximum traveling time of the vehicles being kept to a minimum. This min–max objective can be motivated by the need to minimize the latest completion time of the travels of the vehicles [7,6], by the desire to balance the traveling time of the vehicles in a fair way [4], and by restrictions such as limited working days to the vehicles [4].

In this paper, we study an extension of the min–max p-TSP, named as the min–max Location-Routing Problem (LRP), which, in addition to the tours, needs to simultaneously determine the home depots of the p vehicles by choosing from a given set $D \subseteq V$ of available depots. In the min–max LRP, we also take the capacities of the available depots into consideration, such that for each $d \in D$, the number of vehicles that start from d cannot exceed its capacity, which is denoted by a positive integer $b(d)$. Moreover, for each customer location $v \in J$, a vertex weight $h(v)$ is given to indicate the time for a vehicle to service v, where $h(v)$ is a non-negative integer. Thus, the min–max LRP is to determine the home depots and the tours for the p vehicles to service all the customer locations, so as to minimize the maximum working time of the vehicles, including both the travel and service time.

Our study of the min–max LRP follows a growing body of research on location-routing problems [17,5,13], which need to decide on the home depots and travel routes of the vehicles simultaneously. Most of the location-routing problems in the previous literature have a min–sum objective, which is to minimize the sum of the travel and service costs of the vehicles, and the cost of establishing the home depots. In the min–max LRP, we assume that the available depots in D have already been established, and only the maximum working time of the vehicles needs to be minimized. We do not include the cost

* Corresponding author. Tel.: +86 20 84113192; fax: +86 20 84036924.
E-mail addresses: lgtzx@polyu.edu.hk (Z. Xu), xudsh@mail.sysu.edu.cn (D. Xu), i@zhuwb.com (W. Zhu).
1 Tel.: +852 34004624; fax: +852 23302704.

0166-218X/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.dam.2011.09.014

of establishing the home depots in the min–max LRP, partially because it cannot be directly combined with the travel and service costs of the vehicles in the min–max objective.

We denote the above mentioned instance of the min–max LRP as $(\mathcal{N}, p)$, where $\mathcal{N} = (G, D, J, w, h, b)$ indicates a weighted network. We assume that G is a complete graph (i.e., $E = V \times V$), and that the edge weights form a metric (i.e., they are symmetric and satisfy the triangle inequality), because the vehicles can always travel along the shortest path between any two vertices. We also assume that for each $v \in V$, either $v \in D$ or $v \in J$, because otherwise, if there exists $v \in V$ that is in neither D nor J, then the vehicles can always skip v due to the triangle inequality of the edge weights, which implies that we can always remove v from G. Thus, we can assume that $D \cup J = V$. Moreover, we assume that $p \leq |J|$, because otherwise, if $p > |J|$, then there are always at least $(p - |J|)$ vehicles that do not service any customer locations, and therefore we can consider only $|J|$ vehicles. We also assume that $b(d) \leq p$ for each available depot $d \in D$, because otherwise, if there exists $d \in D$ with $b(d) > p$, we can always revise $b(d)$ to be equal to p. Finally, we assume that $\sum_{d \in D} b(d) \geq p$ to avoid trivial infeasible instances that lack the capacities of the available depots.

Since the classical traveling salesman problem (TSP) is a special case of the min–max LRP with $p = 1$, and is known to be strongly $\mathbb{NP}$-hard [9], the min–max LRP is also strongly $\mathbb{NP}$-hard. Thus, in this paper, we present approximation algorithms that achieve constant approximation ratios for the min–max LRP and its special cases. Here, a ρ-approximation algorithm is defined as a polynomial time algorithm that produces a feasible solution of objective value no more than ρ times the objective value of an optimal solution to a minimization problem. The value of ρ is called an approximation ratio of the algorithm.

1.1. Previous results

We use $(\alpha; \beta; \gamma)$ to classify the settings for different special cases of the min–max LRP, where α, β, and γ specify the restrictions on h, D, and b of a problem instance, respectively. For example, the setting $(h = 0; D = J; b = \infty)$ indicates that the problem instance satisfies $h(v) = 0$ for each $v \in J$, $D = J$, and $b(d) = \infty$ for each $d \in D$. In this paper, the min–max LRP with a setting $(\alpha; \beta; \gamma)$ is referred to as the min–max $(\alpha; \beta; \gamma)$-LRP. As a convention, we set α, β, and γ to be h, D, and b, respectively, when no restrictions are specified on h, D, and b. Thus, the min–max LRP can also be referred to as the min–max $(h; D; b)$-LRP.

In the literature to date, constant ratio approximation algorithms have been developed only for some special cases of the min–max LRP. Firstly, the min–max p-TSP is a special case of the min–max LRP with a setting $(h = 0; |D| = p; b = 1)$, where each available depot in D must be assigned as the home depot of a unique and distinct vehicle. The best approximation algorithm for the min–max p-TSP was developed by Even et al. [7], achieving an approximation ratio of $8 + \epsilon$ for any $\epsilon > 0$. Moreover, when G is a path, there exists a $(2 + \epsilon)$-approximation algorithm for the min–max p-TSP and the min–max $(h; |D| = p; b = 1)$-LRP [11].

Secondly, Nagamochi [14] studied a min–max rooted-cycle cover problem, which is equivalent to the min–max $(h; |D| = 1; b = \infty)$-LRP, where only one available depot, with unlimited capacity, is included in D, and this must be assigned as the home depot of all the vehicles. Nagamochi [14] developed a $[6 - 4/(p + 1)]$-approximation algorithm for this problem. When $h(v) = 0$ for $v \in J$, the approximation ratio can be reduced to $(5/2 - 1/p)$ [8].

Moreover, certain special cases of the min–max LRP with $D = J$ have also been studied in the previous literature, where each customer location can be assigned as the home depot of a vehicle. Even et al. [7] developed an $(8 + \epsilon)$-approximation algorithm for any $\epsilon > 0$ for a nurse location routing problem, which is equivalent to the min–max $(h = 0; D = J; b = 1)$-LRP. Assuming that G is a tree and p is a constant, Averbakh and Berman [3] and Nagamochi and Okada [15] studied a problem equivalent to the min–max $(h; D = J; b = \infty)$-LRP, and developed two algorithms that achieve approximation ratios of $2p/(p + 1)$ and $(p + 1)/(p - 1)$, respectively. However, the time complexities of the two algorithms are exponential in p. Furthermore, when $p = 2$ and $p = 3$, although the min–max $(h; D = J; b = \infty)$-LRP is still $\mathbb{NP}$-hard [3], Averbakh and Berman [4] proved that one can find optimal home depots of the p vehicles in polynomial time, without the corresponding tours.

It is widely known that from a set of p trees, one can obtain a set of p tours by first duplicating the edges of the trees to obtain p Eulerian graphs, and then computing a Eulerian cycle in each Eulerian graph [7]. Therefore, most of the approximation algorithms in the literature for the special cases of the min–max LRP mentioned above are based on approximation solutions to some min–max tree-cover problems, which aim to find a set of trees that cover all the vertices of a given graph so as to minimize the maximum of the weights of the trees [16,14,7]. In addition to the tree covers, approximation algorithms have also been developed in the literature for min–max graph cover problems that use other covering objectives, such as stars and paths [1,19].

Recently, inapproximability bounds have been derived for some min–max graph cover problems [18,19]. Among them, the only inapproximability bound known for the min–max LRP is 20/17 [18], which was derived for a tour cover problem equivalent to the min–max $(h = 0; |D| = 1; b = \infty)$-LRP.

1.2. Our results

We first define a gap-preserving reduction as follows. An algorithm Φ, is a gap-preserving reduction from a minimization problem $\mathcal{A}$ to a minimization problem $\mathcal{B}$, if algorithm Φ satisfies that given any instance of $\mathcal{A}$, algorithm Φ can compute in polynomial time an instance of $\mathcal{B}$, such that given any solution to $\mathcal{B}$ with the objective value at most being $\rho \geq 1$ times

the minimum of $\mathcal{B}$, algorithm Φ can further compute in polynomial time a solution to $\mathcal{A}$ with the objective value at most ρ times the minimum of $\mathcal{A}$. Therefore, for any $\rho \geq 1$, the existence of a ρ-approximation algorithm for $\mathcal{B}$ implies the existence of a ρ-approximation algorithm for $\mathcal{A}$.

The main results and the organization of this paper can be summarized as follows:

- After introducing the notation in Section 2, we present some preliminary results in Section 3, including the proofs of the existence of a gap-preserving reduction from the min–max $(\alpha; \beta; \gamma)$-LRP to the min–max $(h = 0; \beta; \gamma)$-LRP for any setting $(\alpha; \beta; \gamma)$, and the existence of a gap-preserving reduction from the min–max $(\alpha; \beta; \gamma)$-LRP to the min–max $(\alpha; D; b = 1)$-LRP for any setting $(\alpha; \beta; \gamma)$.
- We develop in Section 4 a tree decomposition procedure, which is used extensively in the development of approximation algorithms for the min–max LRP and its special cases.
- We develop a 7-approximation algorithm for the min–max $(h; |D| = p; b = 1)$-LRP (in Section 5), which improves the existing best approximation ratio from $(8 + \epsilon)$ for the min–max $(h = 0; |D| = p; b = 1)$-LRP [7].
- We develop the first constant ratio approximation algorithm for the min–max $(h; D; b)$-LRP (in Section 6), achieving an approximation ratio of 13.
- We develop a 6-approximation algorithm for the min–max $(h; D = J; b)$-LRP (in Section 7), which improves the existing best approximation ratio from $(8 + \epsilon)$ for the min–max $(h = 0; D = J; b = 1)$-LRP [7].
- We develop the first constant ratio approximation algorithm for the min–max $(h; D; b = \infty)$-LRP (in Section 8), achieving an approximation ratio of 7.
- For every setting of $(\alpha; \beta; \gamma)$ of the min–max LRP studied in this paper, we derive an inapproximability bound of $4/3$ (in Section 9), by showing that there exist no polynomial time approximation algorithms that can achieve an approximation ratio of $(4/3 - \epsilon)$ for any $\epsilon > 0$ unless $\mathbb{NP} = \mathbb{P}$, which improves the existing best bound from $20/17$ for the min–max $(h; D; b)$-LRP and the min–max $(h; D; b = \infty)$-LRP [18].

2. Notation

Consider an instance $(\mathcal{N}, p)$ of the min–max LRP, where $\mathcal{N} = (G, D, J, w, h, b)$ and $G = (V, E)$. For any subgraph H of G, let $J(H)$, $V(H)$, and $E(H)$ denote the customer location set, vertex set, and edge set of H, respectively. For any edge subset $Q \subseteq E$, let $w(Q)$ and $w_{\max}(Q)$ denote the total edge weight and maximum edge weight of Q, respectively. For any vertex subset $U \subseteq J$, let $h(U)$ and $h_{\max}(U)$ denote the total vertex weight and maximum vertex weight of U, respectively, and let $G(U)$ denote the subgraph of G induced by U. Moreover, for any $\delta \geq 0$, let $E[\delta] = \{e : w(e) \leq \delta,\ \forall e \in E\}$ denote the set of edges of G with weights less than or equal to δ. We use $G[\delta]$ to denote the subgraph $(V, E[\delta])$ of G.

We define a *p-schedule* of $\mathcal{N}$ as a collection $\mathcal{S} = \{S_i : 1 \leq i \leq p\}$ with $S_i = (r_i, Q_i, A_i)$, where $r_i \in D$ indicates the home depot of vehicle i, Q_i indicates a connected subgraph of G for the construction of a tour of vehicle i, and $A_i \subseteq J$ indicates the set of customer locations that are allocated to be served by vehicle i, such that the following conditions are satisfied:

- the customer allocation sets A_i for $1 \leq i \leq p$ must form a partition of J, i.e., $A_i \cap A_j = \emptyset$ for $i \neq j$ and $J = \bigcup_{i=1}^{p} A_i$;
- for each vehicle i, where $1 \leq i \leq p$, customer locations in A_i must be covered by the subgraph Q_i, i.e., $A_i \subseteq J(Q_i)$;
- for each vehicle i, where $1 \leq i \leq p$, its home depot r_i must be covered by Q_i, i.e., $r_i \in V(Q_i)$;
- for each $d \in D$, its capacity should not be exceeded, i.e., there are at most $b(d)$ indices i for $1 \leq i \leq p$ with $r_i = d$.

For $1 \leq i \leq p$, the total weight of $S_i = (r_i, Q_i, A_i)$ is defined as equal to $w(E(Q_i)) + h(A_i)$. The cost of the p-schedule $\mathcal{S}$, denoted by $\text{cost}(\mathcal{S})$, is defined as equal to $\max_{1 \leq i \leq p}[w(E(Q_i)) + h(A_i)]$. We use $E(\mathcal{S})$ to denote the set of edges of Q_i for all $(r_i, Q_i, A_i) \in \mathcal{S}$. Moreover, we define $\mathcal{S}$ as a *p-tour schedule* if $\mathcal{S}$ is a p-schedule with each Q_i for $1 \leq i \leq p$ being a tour, and define $\mathcal{S}$ as a *p-tree schedule* if $\mathcal{S}$ is a p-schedule with each Q_i for $1 \leq i \leq p$ being a tree rooted at r_i.

Thus, given any instance $(\mathcal{N}, p)$, the min–max LRP can be defined as to find a p-tour schedule $\mathcal{S}$ of $\mathcal{N}$ with $\text{cost}(\mathcal{S})$ minimized. Accordingly, a feasible solution to $(\mathcal{N}, p)$ is a p-tour schedule of $\mathcal{N}$, and an optimal solution to $(\mathcal{N}, p)$ is a feasible solution with a minimum cost. We use $\text{OPT}(\mathcal{N}, p)$ to denote the cost of an optimal solution to $(\mathcal{N}, p)$. Due to the assumption of $\sum_{d \in D} b(d) \geq p$ in Section 1, it can be seen that $(\mathcal{N}, p)$ always has a feasible solution.

3. Preliminary results

Firstly, we present Proposition 1, which is proved in Appendix A.

Proposition 1. *Consider any instance $(\mathcal{N}, p)$. Given any p-tree schedule $\mathcal{S}$ of $\mathcal{N}$, one can obtain a p-tour schedule of $\mathcal{N}$ with a cost not greater than $2\text{cost}(\mathcal{S})$ in polynomial time.*

Proposition 1 implies that for any setting $(\alpha; \beta; \gamma)$, if there is a polynomial time algorithm that can produce a p-tree schedule of $(\mathcal{N}, p)$ with a cost not greater than $\rho \cdot \text{OPT}(\mathcal{N}, p)$ for any instance $(\mathcal{N}, p)$ of the min–max (α, β, γ)-LRP, then there exists a (2ρ)-approximation algorithm for the min–max (α, β, γ)-LRP. Thus, like most approximation algorithms in the literature for various special cases of the min–max LRP [16,14,7], approximation algorithms in this paper are also based on constructions of tree schedules.

Secondly, we establish Lemma 1, which implies that for any setting $(\alpha; \beta; \gamma)$, if there exists a ρ-approximation algorithm for the min–max $(h = 0; \beta; \gamma)$-LRP or for the min–max $(\alpha; D; b = 1)$-LRP, then there exists a ρ-approximation algorithm for the min–max $(\alpha; \beta; \gamma)$-LRP.

Lemma 1. *For any setting* $(\alpha; \beta; \gamma)$, (i)*there exists a gap-preserving reduction from the min–max* $(\alpha; \beta; \gamma)$*-LRP to the min–max* $(h = 0; \beta; \gamma)$*-LRP;* (ii) *there exists a gap-preserving reduction from the min–max* $(\alpha; \beta; \gamma)$*-LRP to the min–max* $(\alpha; D; b = 1)$*-LRP.*

Proof. Consider any setting $(\alpha; \beta; \gamma)$, and any instance $(\mathcal{N}, p)$ of the min–max $(\alpha; \beta; \gamma)$-LRP, where $\mathcal{N} = (G, D, J, w, h, b)$ and $G = (V, E)$.

To prove (i), we define $h''(u) = 0$ for each $u \in J$, and define a revised edge weight for each $(u, v) \in E$ as $w''(u, v) = h(u) + 2w(u, v) + h(v)$, so that the revised edge weights w'' are all integers and form a metric. Thus, $(\mathcal{N}'', p)$, where $\mathcal{N}'' = (G, D, J, w'', h'', b)$, is an instance of the min–max $(h = 0; \beta; \gamma)$-LRP. Moreover, on the one hand, each p-tour schedule of $\mathcal{N}''$ is a p-tour schedule of $\mathcal{N}$ with its cost in $\mathcal{N}''$ larger than or equal to twice its cost in $\mathcal{N}$. On the other hand, from any p-tour schedule $\mathcal{S}$ of $\mathcal{N}$, one can obtain a p-tour schedule $\mathcal{S}''$ of $\mathcal{N}''$ by skipping all the repeated visits of each customer location on the tours of $\mathcal{S}$. Due to the triangle inequality and the fact that no two tours of $\mathcal{S}''$ visit the same customer location, it can be seen that the cost of $\mathcal{S}$ in $\mathcal{N}$ is larger than or equal to half of the cost of $\mathcal{S}''$ in $\mathcal{N}''$. Thus, we obtain that $\mathrm{OPT}(\mathcal{N}'', p) = 2\mathrm{OPT}(\mathcal{N}, p)$, and that the transformation from $(\mathcal{N}, p)$ to $(\mathcal{N}'', p)$ and the transformation from every p-tour schedule of $\mathcal{N}''$ to itself together form a gap-preserving reduction from the min–max $(\alpha; \beta; \gamma)$-LRP to the min–max $(h = 0; \beta; \gamma)$-LRP.

To prove (ii), let d_{q1} for $1 \le q \le |D|$ denote the available depots in D. We use the following procedure to transform $(\mathcal{N}, p)$ to an instance of the min–max $(\alpha; D; b = 1)$-LRP. Each $d_{q1} \in D$ can be assigned as a home depot for only $b(d_{q1})$ vehicles at most. Thus, we can extend D to define $D' = D \cup \{d_{q2}, d_{q3}, \ldots, d_{q,b(d_{q1})} : \forall d_{q1} \in D\}$, where each d_{qj} for $2 \le j \le b(d_{q1})$ is a replication of d_{q1} that shares the same location as d_{q1}. For consistency, we treat d_{q1} for each $d_{q1} \in D$ as one of its own replications in D'. For each $d_{qj} \in D'$, let its capacity be $b'(d_{qj}) = 1$. Define $V' = D' \cup J$, and let G' denote the complete graph on V'. For each pair of vertices u' and v' of V', let u and v denote the vertices of V that have the same locations as u' and v', respectively. We can then define $w'(u', v') = w(u, v)$ as the edge weight of (u', v') in G', so that the edge weights defined by w' are all integers and form a metric. Thus, $(\mathcal{N}', p)$, where $\mathcal{N}' = (G', D', J, w', h, b')$, is an instance of the min–max $(\alpha; D; b = 1)$-LRP. Moreover, on the one hand, from any p-tour schedule of $\mathcal{N}'$, we can transform it to a p-tour schedule of $\mathcal{N}$ without changing the cost, by contracting all the replications $d_{q1}, d_{q2}, \ldots, d_{q,b(d_{q1})}$ into d_{q1} for each $d_{q1} \in D$. On the other hand, from any p-tour schedule of $\mathcal{N}$, we can transform it to a p-tour schedule of $\mathcal{N}'$ without changing the cost, by expanding each depot in D to their replications in D'. Thus, we obtain that $\mathrm{OPT}(\mathcal{N}, p) = \mathrm{OPT}(\mathcal{N}', p)$, and that the transformation from $(\mathcal{N}, p)$ to $(\mathcal{N}', p)$ and the transformation from any p-tour schedule of $\mathcal{N}'$ to a p-tour schedule of $\mathcal{N}$ together form a gap-preserving reduction from the min–max $(\alpha; \beta; \gamma)$-LRP to the min–max $(\alpha; D; b = 1)$-LRP. □

Our basic idea for developing approximation algorithms for the min–max LRP with a setting $(\alpha; \beta; \gamma)$ is as follows. We first develop a procedure which, for a given problem instance $(\mathcal{N}, p)$ with $\mathcal{N} = (G, D, J, w, h, b)$ and a guessed value λ of $\mathrm{OPT}(\mathcal{N}, p)$, runs in polynomial time to either return a p-tree schedule of $\mathcal{N}$ with a cost at most $\rho\lambda$ where $\rho \ge 1$, or return and guarantee that the guessed value is too low, i.e., $\lambda < \mathrm{OPT}(\mathcal{N}, p)$. We define such a procedure as an ORACLE_ρ for the min–max $(\alpha; \beta; \gamma)$-LRP.

Since $0 \le \mathrm{OPT}(\mathcal{N}, p) \le w(E) + h(J)$, we next use a binary search to obtain an integer λ^* in the interval $[0, w(E) + h(J)]$, such that the ORACLE_ρ returns a p-tree schedule of $\mathcal{N}$ with a cost at most $\rho\lambda$ when $\lambda \ge \lambda^*$, and that it returns that λ is too low when $\lambda < \lambda^*$. The binary search runs in $O(\log[w(E) + h(J)])$ time, which is polynomial to the input length of the problem instance. Since $\mathrm{OPT}(\mathcal{N}, p)$ is an integer, we obtain $\lambda^* \le \mathrm{OPT}(\mathcal{N}, p)$. Thus, executing the ORACLE_ρ with $\lambda = \lambda^*$ yields a p-tree schedule $\mathcal{S}$ of $\mathcal{N}$ with $\mathrm{cost}(\mathcal{S}) \le \rho\lambda^* \le \rho\mathrm{OPT}(\mathcal{N}, p)$. According to Proposition 1, we can construct, from $\mathcal{S}$, a p-tour schedule of $\mathcal{N}$ with a cost at most $2\rho\mathrm{OPT}(\mathcal{N}, p)$. Thus, we have obtained a (2ρ)-approximation algorithm, and have established Theorem 1.

Theorem 1. *For any setting* $(\alpha; \beta; \gamma)$*, if an* ORACLE_ρ *for the min–max* $(\alpha; \beta; \gamma)$*-LRP is given, a* (2ρ)*-approximation algorithm for the min–max* $(\alpha; \beta; \gamma)$*-LRP can be obtained by binary search.*

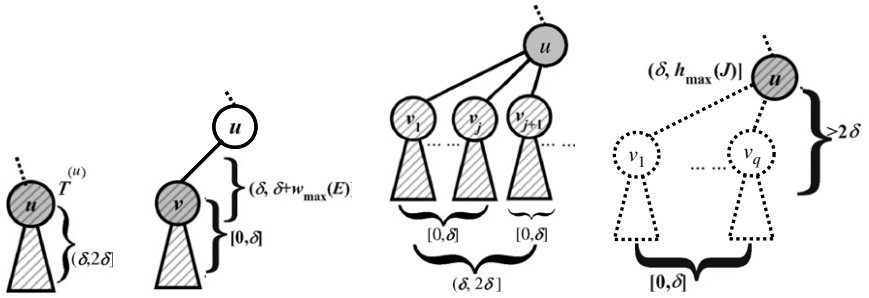
Finally, given a p-tour schedule, we can use the following proposition to obtain an upper bound on the weight of every edge that joins vertices of the same tour in the schedule.

Proposition 2. *Given any instance* $(\mathcal{N}, p)$*, consider any* p*-tour schedule* $\mathcal{S}$ *of* $\mathcal{N}$*. Then, for every edge* (u, v) *with* u *and* v *both belonging to the same tour of* $\mathcal{S}$*, it satisfies that* $w(u, v) \le \mathrm{cost}(\mathcal{S})/2$*.*

Proof. Suppose that $\mathcal{N} = (G, D, J, w, h, b)$. Let Q_i denote the tour of $\mathcal{S}$ that u and v both belong to, which implies that $w(E(Q_i)) \le \mathrm{cost}(\mathcal{S})$, and that $2w(u, v) = w(u, v) + w(v, u) \le w(E(Q_i))$ due to the triangle inequality. Thus, we obtain $w(u, v) \le \mathrm{cost}(\mathcal{S})/2$. □

4. Tree decomposition

We present a tree decomposition procedure in Algorithm 1, which is used extensively in the approximation algorithms developed in this paper. It extends the procedure in [7] by taking the service time at customer locations into consideration, and is particularly useful for the design of the algorithm for the min–max $(h; D; b)$-LRP in Section 6.



(a) Step 2.1. (b) Step 2.2.1. (c) Step 2.2.2.A. (d) Step 2.2.2.B.

Fig. 1. An illustration of the iteration i of Step 2 of Algorithm 1 to construct (r_i, T_i, A_i), where each vertex in gray denotes r_i, each subtree in bold forms T_i, vertices in each area with forward slash but not in A form A_i.

An input of Algorithm 1 consists of a threshold $\delta > 0$ and a weighted network $\mathcal{N} = (G, D, J, w, h, b)$, where $G = (V, E)$ is a rooted tree with the root denoted by r. Since D and b are not used in Algorithm 1, we assume that $D = J$ and $b(d) = \infty$ for $d \in D$.

The output of Algorithm 1 consists of an integer m and an m-tree schedule of $\mathcal{N}$ denoted by $\mathcal{S} = \{S_i : 1 \le i \le m\}$ with $S_i = (r_i, T_i, A_i)$. Algorithm 1 constructs S_i for $i = 1, 2, \ldots, m-1$ iteratively in Step 2. Throughout the iterations in Step 2, it uses T, which is initially set to be G in Step 1, to denote the rooted tree that remains to be decomposed, and uses A, which is initially set to be empty in Step 1, to denote the set of customer locations that have already been allocated to some A_i. As later shown in Theorem 2, the total weight of each S_i satisfies a lower bound and/or an upper bound defined by the threshold δ.

For each vertex $u \in V(T)$, we use $T^{(u)}$ to denote the subtree of T that is rooted at u and includes all the descendants of u in T. For each edge $e = (u, v)$ of T, where u is the parent of v in T, we use $T^{(e)}$ or $T^{(u,v)}$ to denote the subtree of T that consists of u, $T^{(v)}$, and (u, v). Thus, for the given threshold $\delta > 0$ and the subset A of customer locations, we define a rooted subtree $T^{(u)}$ for $u \in V(T)$ to be *light, medium, or heavy* with respect to δ and A, if $w(E(T^{(u)})) + h(J(T^{(u)}) \setminus A)$ is in the intervals $[0, \delta]$, $(\delta, 2\delta]$, or $(2\delta, \infty)$ respectively, and where $J(T^{(u)}) \setminus A$ indicates the set of customer locations in $T^{(u)}$ that have not been allocated.

During each iteration i in Step 2, Algorithm 1 first identifies a subtree T_i of T, a vertex $r_i \in V(T_i)$, and a subset A_i of customer locations not in A. This is done in Step 2.1 if T has a medium subtree $T^{(u)}$ with respect to δ and A, and is done in Step 2.2 otherwise. It aims to ensure $\delta < w(E(T_i)) + h(A_i) \le \max\{2\delta, \delta + w_{\max}(E), h_{\max}(J)\}$, which is shown in Algorithm 1. In Step 2.3, Algorithm 1 sets $S_i = (r_i, T_i, A_i)$, and updates $\mathcal{S}$ and A accordingly. It then splits T_i away from T in Step 2.4, and moves forward to the next iteration of Step 2 unless T becomes light with respect to δ and A. Fig. 1(a)–(d) provide an illustration of Step 2. After the iterations in Step 2, Algorithm 1 constructs $S_m = (r_m, T_m, A_m)$ in Step 3, to assign to T_m the remaining tree T, and to allocate to A_m the customer locations remaining in A (which can be empty). It then returns m and $\mathcal{S} = \{(r_i, T_i, A_i) : 1 \le i \le m\}$ in Step 4.

Algorithm 1 (*Tree-Decomposition*). *Input*: A threshold $\delta > 0$, and a weighted network $\mathcal{N} = (G, D, J, w, h, b)$ where $G = (V, E)$ is a tree with the root denoted by r, $D = J$, and $b(d) = \infty$ for $d \in D$.

Output: An integer m, and an m-tree schedule of $\mathcal{N}$ denoted by $\mathcal{S} = \{S_i : 1 \le i \le m\}$ with $S_i = (r_i, T_i, A_i)$.

1. Initially, set $i = 0$, $\mathcal{S} = \emptyset$, $T = G$, and $A = \emptyset$.
2. Until T is light with respect to δ and A, increase i by 1, and consider the following cases to construct (r_i, T_i, A_i):
   - 2.1. If T has a medium subtree $T^{(u)}$ with respect to δ and A, where $u \in V(T)$, then set $T_i = T^{(u)}$, $r_i = u$, and $A_i = J(T_i) \setminus A$. See Fig. 1(a).
   - 2.2. Otherwise, T must have a subtree $T^{(u)}$ with $u \in V(T)$ such that $T^{(u)}$ is heavy with respect to δ and A, and that $T^{(v)}$ is light with respect to δ and A for every child v of u in T (or u has no child). Then,
     - 2.2.1. If there exists a child v of u such that $\delta < w(E(T^{(v)})) + h(J(T^{(v)}) \setminus A) + w(u, v)$, then set $T_i = T^{(u,v)}$, $r_i = v$, and $A_i = J(T^{(v)}) \setminus A$. See Fig. 1(b).
     - 2.2.2. Otherwise, each child v of u satisfies $w(E(T^{(v)})) + h(J(T^{(v)}) \setminus A) + w(u, v) \le \delta$, (or u has no child). Let $v_1, v_2, \ldots, v_q$ denote the children of u, where $q \ge 0$. If $q > 0$, let j denote the largest index j' such that $\sum_{t=1}^{j'} [w(E(T^{(v_t)})) + h(J(T^{(v_t)}) \setminus A) + w(u, v_t)] \le \delta$, where $1 \le j' \le q$. Otherwise, set $j = 0$.
       - A. If $j < q$, set $T_i = \bigcup_{t=1}^{j+1} T^{(u,v_t)}$, $A_i = J(T_i) \setminus A \setminus \{u\}$, and $r_i = u$. See Fig. 1(c).
       - B. Otherwise, $j = q$, which implies that $w(E(T^{(u)})) + h(J(T^{(u)}) \setminus A \setminus \{u\}) \le \delta$. Since $T^{(u)}$ is heavy with respect to δ and A, it can be seen that $h(u) > \delta$ and $u \notin A$. Set $T_i = (\{u\}, \emptyset)$, which is a tree that contains u only, and set $A_i = \{u\}$ and $r_i = u$. See Fig. 1(d).
   - 2.3. Set $S_i = (r_i, T_i, A_i)$. Update $\mathcal{S}$ by $\mathcal{S} \cup \{S_i\}$. Update A by $A \cup A_i$.
   - 2.4. Split T_i away from T, by first removing from $E(T)$ all the edges in $E(T_i)$, and then removing from $V(T)$ all the vertices that are not connected to r by any path in T.

3. Let $m = i + 1$. Set $S_m = (r_m, T_m, A_m)$, where $r_m = r$, $T_m = T$, and $A_m = J(T_m) \setminus A$. Update $\mathcal{S}$ by $\mathcal{S} \cup \{S_m\}$.
4. Return m and $\mathcal{S}$. □

We next establish Theorem 2 to prove the properties of Algorithm 1 for the tree decomposition.

Theorem 2 (*Tree Decomposition*). *Given a valid input, δ and $\mathcal{N}$, Algorithm 1 returns in polynomial time an integer m and an m-tree schedule $\mathcal{S}$ of $\mathcal{N}$, where $\mathcal{S} = \{S_i : 1 \le i \le m\}$ with $S_i = (r_i, T_i, A_i)$, such that the following properties are satisfied:*

(i) *T_i is a subtree of G for $1 \le i \le m$, and T_m contains the root r of T;*
(ii) *$E(T_j)$ and $E(T_i)$ are disjoint for $1 \le j < i \le m$;*
(iii) *$\delta < w(E(T_i)) + h(A_i) \le \max\{2\delta, \delta + w_{\max}(E), h_{\max}(J)\}$ for $1 \le i \le m-1$, and $0 \le w(E(T_m)) + h(A_m) \le \delta$;*
(iv) *$m \le \lceil [w(E) + h(J)]/\delta \rceil$.*

Proof. To examine the time complexity of Algorithm 1, consider each iteration i of Step 2 of Algorithm 1. If (r_i, T_i, A_i) is constructed in Step 2.1, then $T^{(u)}$ is medium, by which it can be verified that either $E(T_i)$ or A_i is not empty. If (r_i, T_i, A_i) is constructed in Step 2.2.1 or Step 2.2.2.A, then it can be verified that $E(T_i)$ is not empty. Otherwise, (r_i, T_i, A_i) is constructed in Step 2.2.2.B, by which it can be verified that A_i is not empty. Thus, we obtain that either A is increased by adding at least one customer location in Step 2.3, or T is shrunk by removing at least one edge in Step 2.4. Hence, after at most $|J|+|E|$ iterations, T must become light with respect to δ and A, and therefore the iterations of Step 2 must stop. Therefore, Algorithm 1 has a polynomial time complexity.

Property (i) and Property (ii) are easy to see as follows. Since T is always a subtree of G in Algorithm 1, each T_i for $1 \le i \le m$, which is a subtree of T, must be a subtree of G. From Step 2.4 we know that $V(T)$ always contains r, which implies that T_m contains r according to Step 3. Thus, Property (i) is proved. Moreover, for $1 \le i < j \le m$, since all the edges in $E(T_j)$ are removed from T in Step 2.4 during iteration j, we have that $E(T_i)$, which is a subset of $E(T)$, must be disjoint with $E(T_j)$. Thus, Property (ii) is proved.

We can then prove that $\mathcal{S}$ is an m-tree schedule of $\mathcal{N}$. Consider $S_i = (r_i, T_i, A_i)$ for each $1 \le i \le m$. Due to Step 2 of Algorithm 1 and Property (i) of Theorem 2, we know $r_i \in V(T_i)$. Due to Step 2.3 and Step 3, we know that A_i for $1 \le i \le m$ form a partition of J. According to Step 2, we have $A_i \subseteq J(T_i)$. Thus, since $b(d) = \infty$ for all $d \in D$, and since $D = J$ and $m = |\mathcal{S}|$, by definition we obtain that $\mathcal{S}$ is an m-tree schedule of $\mathcal{N}$.

To prove Property (iii), consider the tree T and the set A remaining after the iterations of Step 2 of Algorithm 1. Since T must be light with respect to δ and A, we have $0 \le w(E(T)) + h(J(T) \setminus A) \le \delta$, which, together with $T_m = T$ and $A_m = J(T_m) \setminus A$ in Step 3, implies that $0 \le w(E(T_m)) + h(A_m) \le \delta$.

For $1 \le i \le m-1$, we next derive bounds on $w(E(T_i)) + h(A_i)$ for the following four cases of the construction of $S_i = (r_i, T_i, A_i)$. Case 1: If S_i is constructed in Step 2.1, then since $T_i = T^{(u)}$, which is a medium subtree with respect to A and δ, and since $A_i = J(T_i) \setminus A$, it can be seen from Fig. 1(a) that $w(E(T_i)) + h(A_i) \in (\delta, 2\delta]$. Case 2: If S_i is constructed in Step 2.2.1, then $T_i = T^{(u,v)}$, which satisfies that $w(E(T^{(v)})) + h(J(T^{(v)}) \setminus A) + w(u, v) > \delta$ and $w(E(T^{(v)})) + h(J(T^{(v)}) \setminus A) \le \delta$. Thus, since $A_i = J(T^{(v)}) \setminus A$ and $w(u, v) \le w_{\max}(E)$, it can be seen from Fig. 1(b) that $w(E(T_i)) + h(A_i) \in (\delta, \delta + w_{\max}(E)]$. Case 3: If S_i is constructed in Step 2.2.2.A, then from Step 2.2.2, we know $w(E(T^{(u,v_j)})) + h(J(T^{(v_j)}) \setminus A) + w(u, v_j) \le \delta$, and from Step 2.2.2.A we know $w(E(T_i)) + h(A_i) = \sum_{t=1}^{j+1} [w(E(T^{(u,v_t)})) + h(J(T^{(v_t)} \setminus A)) + w(u, v_t)]$. Thus, by the definition of j in Step 2.2.2, it can seen from Fig. 1(c) that $w(E(T_i)) + h(A_i) \in (\delta, 2\delta]$. Case 4: If S_i is constructed in Step 2.2.2.B, then as shown in Fig. 1(d), since T_i contains no edge and A_i contains u only, we have $w(E(T_i)) + h(A_i) = h(u)$. As shown in Step 2.2.2.B, we have $h(u) > \delta$, which, together with $h(u) \le h_{\max}(J)$, implies that $w(E(T_i)) + h(A_i) \in (\delta, h_{\max}(J)]$. Combining the bounds on $w(E(T_i)) + h(A_i)$ for the above four cases, we can conclude that $w(E(T_i)) + h(A_i) \in (\delta, \max\{2\delta, \delta + w_{\max}(E), h_{\max}(J)\}]$. Thus, Property (iii) is proved.

Finally, from Properties (i), (ii), and (iii) of Theorem 2, we know $\sum_{i=1}^{m-1} w(E(T_i)) \le w(E(T))$ and $(m-1)\delta < \sum_{i=1}^{m-1} [w(E(T_i)) + h(A_i)]$, which, together with $\sum_{i=1}^{m-1} h(A_i) \le h(J)$, implies that $(m-1)\delta < w(E) + h(J)$ and $m \le \lceil [w(E) + h(J)]/\delta \rceil$. Thus, Property (iv) is proved. □

5. Min–max $(h; |D| = p; b = 1)$-LRP

Our approximation algorithm for the min–max $(h; D; b)$-LRP, which will be presented in Section 6, relies on an ORACLE$_\rho$ for the min–max $(h = 0; |D| = p; b = 1)$-LRP, which is presented in Algorithm 2 as follows, where $\rho = 3\theta + 0.5$ and $\theta \ge 1$ is a parameter. In the min–max $(h = 0; |D| = p; b = 1)$-LRP, there are exactly p available depots in D, which are denoted by $d_1, d_2, \ldots, d_p$, with capacity $b(d_j) = 1$ for $1 \le j \le p$. Thus, each d_j must be assigned to a distinct vehicle. Moreover, when $\theta = 1$, Algorithm 2 is an ORACLE$_{3.5}$ for the min–max $(h = 0; |D| = p; b = 1)$-LRP, which, according to Theorem 1 and Lemma 1, implies the existence of a 7-approximation algorithm for the min–max $(h; |D| = p; b = 1)$-LRP.

Algorithm 2. *Input*: An instance $(\mathcal{N}, p)$ of the min–max $(h = 0; |D| = p; b = 1)$-LRP, where $\mathcal{N} = (G, D, J, w, h, b)$ and $G = (V, E)$; a guessed value λ of OPT$(\mathcal{N}, p)$; a parameter θ with $\theta \ge 1$.

Output: "λ is too low", or a p-tree schedule $\mathcal{T}$ of $\mathcal{N}$ with cost$(\mathcal{T}) \le (3\theta + 0.5)\lambda$.

1. Construct a graph G' from $G[\lambda/2]$ by contracting all the available depots in $D = \{d_t : 1 \le t \le p\}$ into a single vertex denoted by d'. If G' is not connected, then return "λ is too low". Otherwise, compute a minimum spanning tree M' of G'. Let $\{M_j : 1 \le j \le p\}$ denote the forest obtained from M' by un-contracting d' to the available depots in D, where M_j is a tree rooted at $d_j \in D$.
2. If $w(E(M')) > p\lambda$, then return "λ is too low". Otherwise, for each tree M_j, where $1 \le j \le p$, construct a weighted network $\mathcal{N}_j = (M_j, V(M_j), V(M_j), w, h, b')$ where $b'(v) = \infty$ for all $v \in V(M_j)$. Then, apply Algorithm 1 on $\mathcal{N}_j$ with the threshold equal to $\theta\lambda$ to obtain an integer m_j and an m_j-tree schedule $\mathcal{S}_j$ of $\mathcal{N}_j$, where $\mathcal{S}_j = \{S_{ji} : 1 \le i \le m_j\}$ and $S_{ji} = (r_{ji}, T_{ji}, A_{ji})$.
3. Construct a bipartite graph, where one side of the vertex set is D, and the other side consists of vertices s_{ji} for $1 \le j \le p$ and $1 \le i \le m_j - 1$, where each s_{ji} represents $S_{ji} = (r_{ji}, T_{ji}, A_{ji})$ in $\mathcal{S}_j$. For each $d_t \in D$ and each s_{ji}, they are joined by an edge in the bipartite graph, if and only if d_t and T_{ji} are joined by an edge in $G[\lambda/2]$. Compute a maximum matching of the bipartite graph.
4. If there exists a vertex s_{ji} of the bipartite graph that is not matched with any $d_t \in D$ in Step 3, then return "λ is too low".
5. Return success with $\mathcal{T} = \{(d_t, T'_t, A'_t) : 1 \le t \le p\}$, which is defined as follows. For $1 \le t \le p$, if the available depot d_t is not matched in Step 3, then let $A'_t = A_{t,m_t}$, and let T'_t be the tree that consists of d_t and T_{t,m_t}; otherwise, d_t is matched with some s_{ji} in Step 5, where $1 \le j \le p$ and $1 \le i \le m_j - 1$. Then, let $A'_t = A_{t,m_t} \cup A_{ji}$, and let T'_t be the tree that consists of d_t, T_{t,m_t}, T_{ji}, and the edge of $G[\lambda/2]$ that joins d_t and T_{ji}. □

To prove the correctness of Algorithm 2, we establish Lemmas 2 and 3 as follows.

Lemma 2. *If Algorithm 2 returns "λ is too low" in Step 4, then $\lambda < \mathrm{OPT}(\mathcal{N}, p)$.*

Proof. Suppose that Algorithm 2 returns "λ is too low" in Step 4. Thus, the maximum matching obtained in Step 3 does not cover all the vertices s_{ji} of the bipartite graph, where $1 \le j \le p$ and $1 \le i \le m_j - 1$. Due to Step 1 and Step 2, we know that G' is connected and that $w(E(M')) \le p\lambda$.

Suppose, to the contrary, that there exists a p-tour schedule $\mathcal{Q} = \{(r_t, Q_t, A_t) : 1 \le t \le p\}$ of $\mathcal{N}$ with $\mathrm{cost}(\mathcal{Q}) \le \lambda$. We next show that the maximum matching of the bipartite graph covers all s_{ji} for $1 \le j \le p$ and $1 \le i \le m_j - 1$, which is a contradiction. For each subset $\mathcal{F} \subseteq \{s_{ji} : 1 \le j \le p, 1 \le i \le m_j - 1\}$, let $N(\mathcal{F})$ denote the set of available depots of D that are adjacent to some $s_{ji} \in \mathcal{F}$ in the bipartite graph. According to Hall's Marriage Theorem [12], to prove that the maximum matching of the bipartite graph covers all s_{ji} for $1 \le j \le p$, $1 \le i \le m_j - 1$, it is sufficient to prove $|\mathcal{F}| \le |N(\mathcal{F})|$ as follows.

Let $\mathcal{Q}(\mathcal{F})$ denote the set of $(r_t, Q_t, A_t) \in \mathcal{Q}$ with Q_t containing at least one vertex of T_{ji} for some $s_{ji} \in \mathcal{F}$. Consider each $(r_t, Q_t, A_t) \in \mathcal{Q}(\mathcal{F})$. Suppose that Q_t contains a vertex v of T_{ji} for some $s_{ji} \in \mathcal{F}$. Since $\mathrm{cost}(\mathcal{Q}) \le \lambda$, by Proposition 2 we have $w(r_t, v) \le \lambda/2$. Thus, r_t and s_{ji} are adjacent in the bipartite graph, which implies that $r_t \in N(\mathcal{F})$. Thus, due to $b(r_t) = 1$ for $1 \le t \le p$, we obtain $|\mathcal{Q}(\mathcal{F})| \le |N(\mathcal{F})|$.

We are next going to show $|\mathcal{F}| \le |\mathcal{Q}(\mathcal{F})|$. Due to Property (iii) of Theorem 2, we have $\theta\lambda \le w(E(T_{ji}))$ for each $s_{ji} \in \mathcal{F}$, which implies:

$$|\mathcal{F}| \cdot \theta\lambda \le \sum_{s_{ji} \in \mathcal{F}} w(E(T_{ji})). \tag{1}$$

Since $\mathrm{cost}(\mathcal{Q}) \le \lambda$, we have:

$$\sum_{(r_t, Q_t, A_t) \in \mathcal{Q}(\mathcal{F})} w(E(Q_t)) \le |\mathcal{Q}(\mathcal{F})| \cdot \lambda. \tag{2}$$

Consider the minimum spanning tree M' of G' and the forest $\{M_j : 1 \le j \le p\}$, which are obtained in Step 1 of Algorithm 2, and satisfy $\sum_{j=1}^{p} w(E(M_j)) = w(E(M'))$. Since $\mathrm{cost}(\mathcal{Q}) \le \lambda$, by Proposition 2 we have $E(Q_t) \subseteq E[\lambda/2]$ for $1 \le t \le p$. Thus, from $\{M_j : 1 \le j \le p\}$, we can construct a subgraph M'' of $G[\lambda/2]$ by deleting all the edges of trees T_{ji} for all $s_{ji} \in \mathcal{F}$, and inserting all the edges of tours Q_t for all $(r_t, Q_t, A_t) \in \mathcal{Q}(\mathcal{F})$. Due to Proposition 2 and $\mathrm{cost}(\mathcal{Q}) \le \lambda$, all the vertices on the tours of $\mathcal{Q}(\mathcal{F})$ must belong to the same connected component of $G[\lambda/2]$. Since each tour in $\mathcal{Q}(\mathcal{F})$ contains an available depot of D, it can be seen that M'' is connected if all the available depots in D are contracted into one vertex. Thus, we obtain:

$$0 \le w(E(M'')) - w(E(M')) = -\sum_{s_{ji} \in \mathcal{F}} w(E(T_{ji})) + \sum_{(r_t, Q_t, A_t) \in \mathcal{Q}(\mathcal{F})} w(E(Q_t)). \tag{3}$$

From (1)–(3), and $\theta \ge 1$, we have $|\mathcal{F}| \le (|\mathcal{F}| \cdot \theta\lambda)/\lambda \le |\mathcal{Q}(\mathcal{F})|$.

Hence, we obtain $|\mathcal{F}| \le |N(\mathcal{F})|$, implying that the maximum matching in Step 3 covers all s_{ji} for $1 \le j \le p$ and $1 \le i \le m_j - 1$, leading to a contradiction. Thus, $\lambda < \mathrm{OPT}(\mathcal{N}, p)$. □

Lemma 3. *Algorithm 2 with a parameter $\theta \ge 1$ is an $\mathrm{ORACLE}_{3\theta+0.5}$ for the min–max $(h = 0; |D| = p; b = 1)$-LRP.*

Proof. It is easy to see that Algorithm 2 runs in polynomial time.

If Algorithm 2 returns "λ is too low" in Step 1, then the contracted graph G' is not connected, implying that there exists a customer location $v \in J$ such that no $d \in D$ satisfies $w(d, v) \le \lambda/2$, which, together with Proposition 2, implies that $\lambda < \mathrm{OPT}(\mathcal{N}, p)$.

If Algorithm 2 returns "λ is too low" in Step 2, then $w(E(M')) > p\lambda$. Consider any optimal p-tour schedule $\mathcal{Q}$ of $\mathcal{N}$ with $\text{cost}(\mathcal{Q}) = \text{OPT}(\mathcal{N}, p)$. By contracting the available depots of D into one vertex, the tours of $\mathcal{Q}$ form a connected subgraph of G' that contains all the vertices of G'. Since M' is a minimum spanning tree of G', we have $w(E(M')) \le w(E(\mathcal{Q})) \le p \cdot \text{OPT}(\mathcal{N}, p)$. Thus, we obtain $\lambda < \text{OPT}(\mathcal{N}, p)$.

If Algorithm 2 returns "λ is too low" in Step 4, then by Lemma 2, we have $\lambda < \text{OPT}(\mathcal{N}, p)$.

Otherwise, Algorithm 2 returns success with $\mathcal{T} = \{(d_t, T'_t, A'_t) : 1 \le t \le p\}$, where $\mathcal{T}$ is constructed in Step 5 by combining the trees of $\mathcal{S}_j$ for $1 \le j \le p$ obtained in Step 2 and the matching obtained in Step 3. Due to Theorem 2, each $\mathcal{S}_j$ is an m_j-tree schedule of $\mathcal{N}_j$ for $1 \le j \le p$. Since each available depot is matched with at most one s_{ji} in the bipartite graph, the available depots that are assigned as home depots in $\mathcal{T}$ must be distinct. Thus, it is easy to verify that $\mathcal{T}$ is a p-tree schedule of $\mathcal{N}$. To bound $\text{cost}(\mathcal{T})$, let us consider $S_{ji} = (r_{ji}, T_{ji}, A_{ji}) \in \mathcal{S}$ for $1 \le j \le p$ and $1 \le i \le m_j$, which are obtained in Step 2 by decomposing the forest $\{M_j : 1 \le j \le p\}$. Since each M_j for $1 \le j \le p$ is a subgraph of $G[\lambda/2]$, we have $w_{\max}(E(M_j)) \le \lambda/2$. Thus, by Property (iii) in Theorem 2 and $\theta \ge 1$, we obtain that for $1 \le j \le p$, $w(E(T_{jt})) \le 2\theta\lambda$ for $1 \le t \le m_j - 1$, and $w(E(T_{j,m_j})) \le \theta\lambda$. Next, we consider the following two cases for each $(d_t, T'_t, A'_t) \in \mathcal{T}$ returned by Step 5. Case 1: d_t is not matched in Step 3. Then, due to Property (i) of Theorem 2, we have $d_t \in V(T_{t,m_t})$, which implies that $w(E(T'_t)) = w(E(T_{t,m_t})) \le \theta\lambda$. Case 2: d_t is matched with some s_{ji}, which implies that there exists an edge $e \in E[\lambda/2]$ that joins d_t and T_{ji}. Thus, $w(T'_t) = w(T_{t,m_t}) + w(T_{ji}) + w(e)$. By $w(T_{t,m_t}) \le \theta\lambda$, $w(T_{ji}) \le 2\theta\lambda$, and $w(e) \le \lambda/2$, we obtain $w(E(T'_t)) \le (3\theta + 0.5)\lambda$. Summarizing Cases 1 and 2, we conclude that $w(E(T'_t)) \le (3\theta + 0.5)\lambda$ for each T'_t of $\mathcal{T}$, which implies that $\text{cost}(\mathcal{T}) \le (3\theta + 0.5)\lambda$.

Hence, Algorithm 2 with a parameter $\theta \ge 1$ is an $\text{ORACLE}_{3\theta+0.5}$ for the min–max $(h = 0; |D| = p; b = 1)$-LRP. □

Thus, from Theorem 1, Lemmas 3 and 1, we can establish Theorem 3.

Theorem 3. *The min–max $(h; |D| = p; b = 1)$-LRP has a 7-approximation algorithm.*

Proof. For the min–max $(h = 0; |D| = p; b = 1)$-LRP, by Lemma 3, Algorithm 2 with $\theta = 1$ is an $\text{ORACLE}_{3.5}$, which implies the existence of a 7-approximation algorithm due to Theorem 1. Thus, by Lemma 1, there exists a 7-approximation algorithm for the min–max $(h; |D| = p; b = 1)$-LRP. □

Moreover, we can establish Lemma 4 to derive an upper bound on the total weight of the p-tree schedule returned by Algorithm 2, which is useful in Section 6.

Lemma 4. *Consider any valid input of Algorithm 2, including $(\mathcal{N}, p)$, λ, and θ. Suppose that Algorithm 2 returns success with a p-tree schedule $\mathcal{T}$ of $\mathcal{N}$ in Step 5. Given any m-tour schedule $\mathcal{Q}$ of $\mathcal{N}$ where $m \ge 1$, let $p' = m \cdot \text{cost}(\mathcal{Q})/\lambda$. Then, $w(E(\mathcal{T})) \le (p' + 0.5\lfloor p'/\theta \rfloor)\lambda \le [(1 + 0.5\theta^{-1})p']\lambda$.*

Proof. From $\mathcal{Q}$ we can construct a connected subgraph of G' that covers all the vertices of G', by contracting the available depots into one vertex. Since M' is a minimum spanning tree of G', and since $|\mathcal{Q}| = m$, we have $w(E(M')) \le m \cdot \text{cost}(\mathcal{Q})$. Thus, from Step 2 and theorem 1, we have $\sum_{j=1}^{p}\sum_{i=1}^{m_j} w(E(T_{ji})) \le w(E(M')) \le m \cdot \text{cost}(\mathcal{Q})$. Moreover, due to Property (iii) of Theorem 2, we have $w(E(T_{ji})) > \theta\lambda$ for each s_{ji} of the bipartite graph of Step 3, which implies that the matching, obtained in Step 3 and denoted by W, contains at most $\lfloor w(E(M'))/(\theta\lambda) \rfloor$ edges. From Step 5, we have $w(E(\mathcal{T})) \le \sum_{j=1}^{p}\sum_{i=1}^{m_j} w(E(T_{ji})) + w(E(W))$. Thus, since the weight of each edge of W is not greater than $\lambda/2$, and since $p' = m \cdot \text{cost}(\mathcal{Q})/\lambda$, we obtain:

$$w(E(\mathcal{T})) \le \sum_{j=1}^{p}\sum_{i=1}^{m_j} w(E(T_{ji})) + w(E(W)) \le w(E(M')) + \left\lfloor \frac{w(E(M'))}{\theta\lambda} \right\rfloor \lambda/2$$
$$\le m \cdot \text{cost}(\mathcal{Q}) + \left\lfloor \frac{m \cdot \text{cost}(\mathcal{Q})}{\theta\lambda} \right\rfloor \lambda/2 \le (p' + 0.5\lfloor p'/\theta \rfloor)\lambda \le [(1 + 0.5\theta^{-1})p']\lambda. \quad \square$$

6. Min–max $(h; D; b)$-LRP

According to Lemma 1 and Theorem 1, to obtain a (2ρ)-approximation algorithm for the min–max $(h; D; b)$-LRP for some constant ρ, we only need to develop an ORACLE_ρ for the min–max $(h = 0; D; b = 1)$-LRP. Unlike the min–max $(h = 0; |D| = p; b = 1)$-LRP, in which each available depot must be assigned to a unique and distinct vehicle, the min–max $(h = 0; D; b = 1)$-LRP needs to select some available depots from D for the vehicles, and is therefore more complicated.

Consider any instance $(\mathcal{N}, p)$ of the min–max $(h = 0; D; b = 1)$-LRP, where $\mathcal{N} = (G, D, J, w, h, b)$, and consider any guessed value $\lambda > 0$ of $\text{OPT}(\mathcal{N}, p)$. We are going to develop an ORACLE_ρ for $(\mathcal{N}, \lambda)$, where ρ depends on a parameter $\theta \ge 1$.

Let $\widetilde{E}_\lambda$ denote the set of edges of $G[\lambda/2]$ that each have at least one endpoint in J. Consider the subgraph $H_\lambda = (V, \widetilde{E}_\lambda)$ of $G[\lambda/2]$. We let m_λ denote the number of the connected components of H_λ that each contain at least one customer location, and use $H_{\lambda,1}, H_{\lambda,2}, \ldots, H_{\lambda,m_\lambda}$ to denote these m_λ connected components of H_λ. For $1 \le t \le m_\lambda$, and let $G_{\lambda,t}$ denote the complete subgraph of G induced by $V(H_{\lambda,t})$. Thus, $J \subseteq \bigcup_{1\le t\le m_\lambda} V(H_{\lambda,t}) = \bigcup_{1\le t\le m_\lambda} V(G_{\lambda,t})$.

Let $\mathcal{Q}^* = \{(r_i^*, Q_i^*, A_i^*) : 1 \le i \le p\}$ denote an optimal p-tour schedule of $(\mathcal{N}, p)$ with $\text{cost}(\mathcal{Q}^*) = \text{OPT}(\mathcal{N}, p)$. Due to the triangle inequality, we can assume, without loss of generality, that $V(Q_i^*) \setminus \{r_i^*\} \subseteq J$ for $1 \le i \le p$. Thus, Lemma 5 can be established.

Lemma 5. *If* $\text{OPT}(\mathcal{N}, p) \le \lambda$, *then* (i) *for each tour* Q_i^* *of* $\mathcal{Q}^*$ *where* $1 \le i \le p$, *all the vertices of* Q_i^* *belong to the same connected component of* H_λ; (ii) $D(G_{\lambda,t})$ *is not empty for* $1 \le t \le m_\lambda$.

Proof. To prove (i), consider each Q_i^* of $\mathcal{Q}^*$ for $1 \le i \le p$. If $V(Q_i^*) = \{r_i^*\}$, the proof of (i) is trivial. Otherwise, consider each $v \in V(Q_i^*) \setminus \{r_i^*\}$, which, by the definition of $\mathcal{Q}^*$, must be a customer location. By Proposition 2, $w(v, r_i^*) \le \lambda/2$, which implies that $(v, r_i^*) \in \widetilde{E}_\lambda$. Thus, all the vertices of Q_i^* belong to the same connected component of H_λ. (i) is proved.

By $V(G_{\lambda,t}) = V(H_{\lambda,t})$ we know that $V(G_{\lambda,t})$ contains at least one customer location denoted by u. Thus, since there must exist a depot $d \in D$ such that d and u belong to the same tour of $\mathcal{Q}^*$, by Proposition 2 we obtain $w(u, d) \le \lambda/2$, which implies that $d \in D(G_{\lambda,t})$. (ii) is proved. □

Consider the case when $\text{OPT}(\mathcal{N}, p) \le \lambda$. For $1 \le t \le m_\lambda$, let $\mathcal{Q}_{\lambda,t}^*$ denote the set of $(r_i^*, Q_i^*, A_i^*) \in \mathcal{Q}^*$ for $1 \le i \le p$ with $V(Q_i^*) \subseteq V(H_{\lambda,t})$, let $p_{\lambda,t}^* = |\mathcal{Q}_{\lambda,t}^*|$. By (ii) of Lemma 5 we have $1 \le |D(G_{\lambda,t})|$. Since $J(H_{\lambda,t})$ is not empty, and since $b(d) = 1$ for $d \in D$, we obtain

$$1 \le p_{\lambda,t}^* = |\mathcal{Q}_{\lambda,t}^*| \le |D(H_{\lambda,t})| = |D(G_{\lambda,t})|. \tag{4}$$

By (i) of Lemma 5, sets $\mathcal{Q}_{\lambda,t}^*$ for $1 \le t \le m_\lambda$ are disjoint with each other, which implies:

$$\sum_{t=1}^{m_\lambda} p_{\lambda,t}^* \le |\mathcal{Q}^*| = p. \tag{5}$$

Let $\mathcal{N}_{\lambda,t} = (G_{\lambda,t}, D(G_{\lambda,t}), J(G_{\lambda,t}), w, h, b)$. Since $\mathcal{Q}_{\lambda,t}^*$ is a $(p_{\lambda,t}^*)$-tour schedule of $\mathcal{N}_{\lambda,t}$, we obtain:

$$\text{OPT}(\mathcal{N}_{\lambda,t}, p_{\lambda,t}^*) \le \text{OPT}(\mathcal{N}, p). \tag{6}$$

The observations above suggest the following approach to develop an ORACLE_ρ for $(\mathcal{N}, p)$ and λ. Firstly, we can guess the values of $p_{\lambda,t}^*$ for $1 \le t \le m_\lambda$, with the guessed values denoted by p_t. Then, for $1 \le t \le m_\lambda$, we apply an ORACLE_ρ on $(\mathcal{N}_{\lambda,t}, p_t)$ and λ, which is developed in Algorithm 3. If Algorithm 3 returns success with a p_t-tour schedule of $\mathcal{N}_{\lambda,t}$ for each t, and if $\sum_{t=1}^{m_\lambda} p_t \le p$, we can combine these schedules, and then include additional $(p - \sum_{t=1}^{m_\lambda} p_t)$ tours that each contain only one available depot, to form a p-tour schedule of $\mathcal{N}$.

In the following, we first illustrate and analyze Algorithm 3, and then explain how we guess the values of $p_{\lambda,t}^*$ for $1 \le t \le m_\lambda$. Given any valid input, including λ, θ, an index t with $1 \le t \le m_\lambda$, and a guessed value p_t of $p_{\lambda,t}^*$ with $1 \le p_t \le |D(G_{\lambda,t})|$, Algorithm 3 either returns "λ is too low", which implies that $\lambda < \text{OPT}(\mathcal{N}_{\lambda,t}, p_t)$ as proved in Lemma 7, or returns a p-tree schedule $\mathcal{T}_t$ of $\mathcal{N}_{\lambda,t}$ with $\text{cost}(\mathcal{T}_t) \le \rho\lambda$ and $\rho = \max\{3\theta + 0.5, 6 + \theta^{-1}\}$ as proved in Lemma 8. To achieve this, it first relaxes the number of vehicles to $n = |D(G_{\lambda,t})|$ in Step 1, and applies Algorithm 2 on $(\mathcal{N}_{\lambda,t}, n)$, which is an instance of the min–max $(h = 0; |D| = p; b = 1)$-LRP. If Algorithm 2 returns an n-tree schedule $\mathcal{T}' = \{(r_i', T_i', A_i') : 1 \le i \le n\}$ of $\mathcal{N}_{\lambda,t}$ in Step 1, and if $\sum_{i=1}^{n} w(E(\mathcal{T}_i')) > (1 + 0.5\theta^{-1})p_t\lambda$ in Step 2, Algorithm 3 computes a minimum spanning tree $\widetilde{M}$ of $W[\lambda]$ in Step 3, where W is the subgraph of $G_{\lambda,t}$ induced by $J(G_{\lambda,t})$, and $W[\lambda]$, as shown in Lemma 6, must be connected. It then constructs a weighted network $\widetilde{\mathcal{N}}$ based on $\widetilde{M}$, and applies Algorithm 1 on $\widetilde{\mathcal{N}}$ with the threshold equal to $3\theta + 0.5$ to decompose $\widetilde{M}$ into an $\widetilde{m}$-tree schedule $\widetilde{\mathcal{T}}$ of $\widetilde{\mathcal{N}}$. To ensure $\widetilde{\mathcal{T}}$ can be further transformed to a p-tree schedule of $\mathcal{N}_{\lambda,t}$, for each i with $1 \le i \le n$ and A_i' not empty, Step 3 of Algorithm 3 chooses any vertex $v_i' \in A_i'$ as the representative of (r_i', T_i', A_i'), and sets the vertex weight of v_i' in $\widetilde{\mathcal{N}}$ as $\widetilde{h}(v_i') = w(E(T_i'))$. Let V' denote the set of all the representatives. Thus, if $\widetilde{m} \le p_t$ in Step 4, then for each $(\widetilde{r}_u, \widetilde{T}_u, \widetilde{A}_u) \in \widetilde{\mathcal{T}}$ with $V' \cap \widetilde{A}_u$ not empty, Algorithm 3 combines the customer locations of A_i' for all $v_i' \in V' \cap \widetilde{A}_u$ to obtain a customer allocation set A_u'', and combines the vertices of T_u and the vertices of T_i' for all $v_i' \in V' \cap \widetilde{A}_u$ to obtain a vertex set V_u'' in Step 5. It then constructs a tree T_u'' by computing a minimum spanning tree of the complete graph G_u'' on V_u'', and sets d_u'' to be the available depot r_i' of any $(r_i', T_i', A_i') \in \mathcal{T}'$ with $v_i' \in V' \cap \widetilde{A}_u$. Let $\mathcal{T}''$ denote the set of (d_u'', T_u'', A_u'') that are defined above. Since $|\mathcal{T}''| \le \widetilde{m} \le p_t$, Algorithm 3 can further extend $\mathcal{T}''$ to obtain a p_t-tree schedule of $\mathcal{N}_{\lambda,t}$, which is proved in Lemma 8.

Algorithm 3. *Input*: The guessed value λ of $\text{OPT}(\mathcal{N}, p)$, the parameter $\theta \ge 1$, an index t with $1 \le t \le m_\lambda$, and a guessed value p_t of $p_{\lambda,t}^*$ with $1 \le p_t \le D(G_{\lambda,t})$.

Output: "λ is too low", or a p-tree schedule $\mathcal{T}_t$ of $\mathcal{N}_{\lambda,t}$ with $\text{cost}(\mathcal{T}_t) \le \rho\lambda$, where $\rho = \max\{3\theta + 0.5, 6 + \theta^{-1}\}$.

1. Let $n = |D(G_{\lambda,t})|$. Consider $(\mathcal{N}_{\lambda,t}, n)$, which is an instance of the min–max $(h = 0; |D| = p; b = 1)$-LRP. Apply Algorithm 2 on $(\mathcal{N}_{\lambda,t}, n)$, λ, and θ. If Algorithm 2 returns "λ is too low", then return "λ is too low"; otherwise, let $\mathcal{T}' = \{(r_i', T_i', A_i') : 1 \le i \le n\}$ denote the n-tree schedule of $\mathcal{N}_{\lambda,t}$ returned by Algorithm 2.
2. If $\sum_{i=1}^{n} w(E(\mathcal{T}_i')) > (1 + 0.5\theta^{-1})p_t\lambda$, then return "$\lambda$ is too low".

3. Let W denote the complete subgraph of $G_{\lambda,t}$ induced by $J(G_{\lambda,t})$. Compute a minimum spanning tree $\widetilde{M}$ of the subgraph $W[\lambda]$ of W. For each (r'_i, T'_i, A'_i) of $\mathcal{T}'$ with A'_i not empty, let v'_i denote any vertex in A'_i, which is referred to as the representative of (r'_i, T'_i, A'_i). Let $V' = \{v'_i : A'_i \neq \emptyset, 1 \leq i \leq n\}$. Construct a weighted network $\widetilde{\mathcal{N}} = (\widetilde{M}, J(\widetilde{M}), J(\widetilde{M}), w, \widetilde{h}, \widetilde{b})$. For each $v \in J(\widetilde{M})$, let $\widetilde{b}(v) = \infty$. If v is a representative of some $(r'_i, T'_i, A'_i) \in \mathcal{T}'$, which implies that $v \in V'$, then let $\widetilde{h}(v) = w(E(T'_i))$, and otherwise let $\widetilde{h}(v) = 0$. Apply Algorithm 1 on $\widetilde{\mathcal{N}}$ with the threshold equal to $3 + 0.5\theta^{-1}$ to obtain an integer $\widetilde{m}$, and an $\widetilde{m}$-tree schedule of $\widetilde{\mathcal{N}}$ denoted by $\widetilde{\mathcal{T}} = \{(\widetilde{r}_u, \widetilde{T}_u, \widetilde{A}_u) : 1 \leq u \leq \widetilde{m}\}$.
4. If $\widetilde{m} > p_t$ then return "λ is too low".
5. Consider each $(\widetilde{r}_u, \widetilde{T}_u, \widetilde{A}_u) \in \widetilde{\mathcal{T}}$ with $V' \cap \widetilde{A}_u$ not empty, where $1 \leq u \leq \widetilde{m}$. Let A''_u denote the union of A'_i for all i with $v'_i \in V' \cap \widetilde{A}_u$. Let V''_u denote the union of $V(\widetilde{T}_u)$ and $V(T'_i)$ for all i with $v'_i \in V' \cap \widetilde{A}_u$. Let G''_u denote the complete subgraph of $G_{\lambda,t}$ induced by V''_u. Compute a minimum spanning tree T''_u of G''_u. Choose any $(r'_i, T'_i, A'_i) \in \mathcal{T}'$ with $v'_i \in V' \cap \widetilde{A}_u$, and set $d''_u = r'_i$. Let $\mathcal{T}'' = \{(d''_u, T''_u, A''_u) : 1 \leq u \leq \widetilde{m}, V' \cap \widetilde{A}_u \neq \emptyset\}$. Moreover, set $d_1, d_2, \ldots, d_{p_t - |\mathcal{T}''|}$ to be $(p_t - |\mathcal{T}''|)$ distinct available depots in $D(G_{\lambda,t})$ but not in $\{d''_u : 1 \leq u \leq \widetilde{m}\}$. Let $\mathcal{T}''' = \{(d_u, \{d_u\}, \emptyset) : 1 \leq u \leq p_t - |\mathcal{T}''|\}$. Return success with $\mathcal{T}_t = \mathcal{T}'' \cup \mathcal{T}'''$. □

Lemma 6. *The subgraph $W[\lambda]$, defined in Step 3 of Algorithm 3, is connected.*

Proof. Consider any vertices v and u of $W[\lambda]$. By the definitions of $W[\lambda]$, W, and $G_{\lambda,t}$, we know $V(W[\lambda]) = V(W) = J(G_{\lambda,t}) = J(H_{\lambda,t})$, which implies that $\{v, u\} \subseteq J(H_{\lambda,t})$. Since $H_{\lambda,t}$ is a connected component of H_λ, there exists a path P in $H_{\lambda,t}$ that connects u and v. Since $E(H_\lambda) = \widetilde{E}_\lambda$, we have $E(P) \subseteq \widetilde{E}_\lambda$, which, due to the definition of $\widetilde{E}_\lambda$, implies that $w(e) \leq \lambda/2$ for each $e \in E(P)$. From P we can obtain a path P' by skipping all the vertices not in J. Thus, $V(P') \subseteq J(H_{\lambda,t}) = V(W[\lambda])$. Since each edge in $\widetilde{E}_\lambda$ has at least one endpoint in J, due to the triangle inequality, we have $w(e) \leq \lambda/2 + \lambda/2 = \lambda$ for each $e \in E(P')$. Thus, $E(P') \subseteq E(W[\lambda])$. Since P' connects v and u, we obtain that $W[\lambda]$ is connected. □

Lemma 7. *If Algorithm 3 returns "λ is too low", then $\lambda < \mathrm{OPT}(\mathcal{N}_{\lambda,t}, p_t)$.*

Proof. Let $\mathcal{Q}$ denote any optimal p_t-tour schedule of $\mathcal{N}_{\lambda,t}$. Thus, $\mathrm{cost}(\mathcal{Q}) = \mathrm{OPT}(\mathcal{N}_{\lambda,t}, p_t)$. Consider the following three cases.

Case 1: If Algorithm 3 returns "λ is too low" in Step 1, then Algorithm 2 on $(\mathcal{N}_{\lambda,t}, n)$ returns "λ is too low", which, due to Lemma 3, implies that $\lambda < \mathrm{OPT}(\mathcal{N}_{\lambda,t}, n)$. Moreover, since $p_t \leq n$, we can extend $\mathcal{Q}$ to obtain an n-tour schedule of $\mathcal{N}_{\lambda,t}$ without changing the cost value, by including $(n - p_t)$ additional tours that each contain only one distinct available depot not assigned as a home depot in $\mathcal{Q}$. From this, we obtain $\mathrm{OPT}(\mathcal{N}_{\lambda,t}, n) \leq \mathrm{OPT}(\mathcal{N}_{\lambda,t}, p_t)$, which implies that $\lambda < \mathrm{OPT}(\mathcal{N}_{\lambda,t}, p_t)$.

Case 2: If Algorithm 3 returns "λ is too low" in Step 2, then $(1 + 0.5\theta^{-1})p_t\lambda < \sum_{i=1}^{n} w(E(T'_i))$. By Lemma 4 we have $\sum_{i=1}^{n} w(E(T'_i)) \leq (1 + 0.5\theta^{-1})p_t \cdot \mathrm{cost}(\mathcal{Q})$. Thus, $\lambda < \mathrm{cost}(\mathcal{Q}) = \mathrm{OPT}(\mathcal{N}_{\lambda,t}, p_t)$.

Case 3: If Algorithm 3 returns "λ is too low" in Step 4, then $\widetilde{m} > p_t$. Moreover, from Step 2 we know $\sum_{i=1}^{n} w(E(T'_i)) \leq (1 + 0.5\theta^{-1})p_t\lambda$. Suppose, to the contrary, that $\mathrm{OPT}(\mathcal{N}_{\lambda,t}, p_t) \leq \lambda$. Thus, $\mathrm{cost}(\mathcal{Q}) \leq \lambda$. By skipping the vertices that are not in $V(W)$, we can transform the p_t tours of $\mathcal{Q}$ to obtain a set $\mathcal{Q}'$ of p_t tours in W, where W, as defined in Step 3 of Algorithm 3, is a complete graph on $J(G_{\lambda,t})$. Due to the triangle inequality, we have that for each tour $Q \in \mathcal{Q}'$, it satisfies $w(E(Q)) \leq \mathrm{cost}(\mathcal{Q}) \leq \lambda$, which implies that all the vertices on Q are connected by edges of $W[\lambda]$. By Lemma 6, $W[\lambda]$ is connected. Thus, we can connect the p_t tours in $\mathcal{Q}'$ by adding at most $(p_t - 1)$ edges of $W[\lambda]$, to obtain a connected subgraph of $W[\lambda]$ that covers all the vertices of $W[\lambda]$. Since $\widetilde{M}$ is a minimum spanning tree of $W[\lambda]$, we obtain $w(E(\widetilde{M})) \leq \sum_{Q \in \mathcal{Q}'} w(E(Q)) + (p_t - 1)\lambda \leq p_t\lambda + (p_t - 1)\lambda$, which implies that $w(E(\widetilde{M})) \leq 2p_t\lambda$. Moreover, from Step 3 of Algorithm 3, we know $h(J(\widetilde{M})) \leq \sum_{i=1}^{n} w(E(T'_i)) = w(E(\mathcal{T}')) \leq (1 + 0.5\theta^{-1})p_t\lambda$. Thus, we have $h(J(\widetilde{M})) + w(E(\widetilde{M})) \leq (3 + 0.5\theta^{-1})p_t\lambda$, which, due to Property (iv) of Theorem 2, implies that $\widetilde{m} \leq p_t$, leading to a contradiction. Hence, $\lambda < \mathrm{OPT}(\mathcal{N}_{\lambda,t}, p_t)$. The proof is completed. □

Lemma 8. *If Algorithm 3 returns success with $\mathcal{T}_t$, then $\mathcal{T}_t$ is a p_t-tree schedule of $\mathcal{N}_{\lambda,t}$ with $\mathrm{cost}(\mathcal{T}_t) \leq \rho\lambda$, where $\rho = \max\{3\theta + 0.5, 6 + \theta^{-1}\}$.*

Proof. Suppose that Algorithm 3 returns success with $\mathcal{T}_t$ in Step 5. Firstly, consider the n-tree schedule $\mathcal{T}' = \{(r'_i, T'_i, A'_i) : 1 \leq i \leq n\}$ of $\mathcal{N}_{\lambda,t}$ obtained in Step 1. Since for $1 \leq i \leq n$, the representative v'_i of (r'_i, T'_i, A'_i) defined in Step 2 belongs to A'_i, and since $A'_1, A'_2, \ldots, A'_n$ form a partition of $J(G_{\lambda,t})$, we know that v'_i must be distinct for $1 \leq i \leq n$. Moreover, since $\widetilde{\mathcal{T}} = \{(\widetilde{r}_p, \widetilde{T}_p, \widetilde{A}_p) : 1 \leq p \leq \widetilde{m}\}$ obtained in Step 3 is an $\widetilde{m}$-tree schedule of $\widetilde{\mathcal{N}}$, we know that $\widetilde{A}_u$ for $1 \leq u \leq \widetilde{m}$ form a partition of $J(\widetilde{M})$, which is also a partition of $J(G_{\lambda,t})$ because $J(\widetilde{M}) = J(W) = J(G_{\lambda,t})$. Thus, since A''_u defined in Step 4 equals the union of A_i for all i with $v_i \in V' \cap \widetilde{A}_u$, it is easy to verify that A''_u for $1 \leq u \leq \widetilde{m}$ with $V' \cap \widetilde{A}_u \neq \emptyset$ form a partition of $J(G_{\lambda,t})$.

Secondly, since $A'_i \subseteq V(T'_i)$ for $1 \leq i \leq n$, by the definitions of A''_u and V''_u in Step 5, we have $A''_u \subseteq V''_u$ for $1 \leq u \leq \widetilde{m}$, which implies that $A''_u \subseteq V(T''_u)$ because $V(T''_u) = V''_u$.

Thirdly, for the available depot r'_i assigned to d''_u in Step 5 where $1 \leq u \leq \widetilde{m}$, since $r'_i \in V(T'_i)$ and $V(T'_i) \subseteq V(T''_u)$, we have $r'_i \in V(T''_u)$. From Step 1, we know that each r'_i for $1 \leq i \leq n$ is distinct. Thus, since the representative v'_i of each (r'_i, T'_i, A'_i)

is distinct, the available depot d''_u of each (d''_u, T''_u, A''_u) is distinct, which implies that the capacity of each available depot is not exceeded.

Hence, $\mathcal{T}' = \{(d''_u, T''_u, A''_u) : 1 \le u \le \widetilde{m}, V' \cap \widetilde{A}_u \ne \emptyset\}$ is a $|\mathcal{T}'|$-tree schedule of $\mathcal{N}_{\lambda,t}$. Moreover, from Step 4 we know $\widetilde{m} \le p_t$, which, together with $|\mathcal{T}'| \le \widetilde{m}$, implies that $|\mathcal{T}'| \le p_t$. Thus, since $p_t \le |D(G_{\lambda,t})|$, it can be seen that $\mathcal{T}_t$ obtained in Step 5 is a p_t-tree schedule of $\mathcal{N}_{\lambda,t}$.

Furthermore, we can bound cost$(\mathcal{T}_t)$ as follows. By Lemma 3, we know $\mathcal{T}'$ obtained in Step 1 satisfies cost$(\mathcal{T}') \le (3\theta+0.5)\lambda$, which implies that $\widetilde{h}(v) \le (3\theta+0.5)\lambda$ for each $v \in J(\widetilde{M}) \cap V'$ in Step 3. Thus, we obtain $\widetilde{h}_{\max}(J(\widetilde{M})) \le (3\theta+0.5)\lambda$, which, together with Property (iii) of Theorem 2, implies that $\widetilde{\mathcal{T}}$ obtained in Step 3 satisfies cost$(\widetilde{\mathcal{T}}) \le \max\{3\theta + 0.5, 6 + \theta^{-1}\}\lambda$ (under w and $\widetilde{h}$). Moreover, consider each $(r''_u, T''_u, A''_u) \in \mathcal{T}''$ obtained in Step 5. For each i with $v'_i \in V' \cap \widetilde{A}_u$, we know $v'_i \in V(T'_i)$ and $v'_i \in V(\widetilde{T}_u)$. Thus, G''_u defined in Step 5 is connected. By combining $\widetilde{T}_u$ and T'_i for all i with $v'_i \in V' \cap \widetilde{A}_u$, we can obtain a connected subgraph of G''_u that covers all the vertices of G''_u. Since T''_u is a minimum spanning tree of G''_u, and since $\widetilde{h}(v'_i) = w(E(T'_i))$ for all i with $v'_i \in V' \cap \widetilde{A}_u$, we obtain $w(E(T''_u)) \le w(E(\widetilde{T}_u)) + \widetilde{h}(\widetilde{A}_u)$. Hence, cost$(\mathcal{T}'')$ (under w and h) is less than or equal to cost$(\widetilde{\mathcal{T}})$ (under w and $\widetilde{h}$). Moreover, from Step 5, we know cost$(\mathcal{T}''') = 0$ and cost$(\mathcal{T}_t) = \max\{\text{cost}(\mathcal{T}''), \text{cost}(\mathcal{T}''')\}$. Thus, we obtain cost$(\mathcal{T}_t) \le \max\{3\theta + 0.5, 6 + \theta^{-1}\}\lambda = \rho\lambda$. □

Next, we are going to determine the guessed value of $p^*_{\lambda,t}$, for any λ and $1 \le t \le m_\lambda$. Let $\hat{p}_{\lambda,t}$ denote the minimum integer value of p_t with $1 \le p_t$ and $p_t \le |D(G_{\lambda,t})|$, such that Algorithm 3 on λ, t, p_t, and θ returns success, and we define $\hat{p}_{\lambda,t} = \infty$ if no such p_t exists. For any two guessed values, p'_t and p''_t, with $p'_t < p''_t$, it is easy to verify that if Algorithm 3 on λ, t, p''_t, and θ returns "λ is too low", then Algorithm 3 on λ, t, p'_t, and θ also returns "λ is too low". Thus, we can apply a binary search to obtain $\hat{p}_{\lambda,t}$ in polynomial time. Moreover, we can establish Lemma 9 for $\hat{p}_{\lambda,t}$.

Lemma 9. *For any $\lambda > 0$, if $\sum_{t=1}^{m_\lambda} \hat{p}_{\lambda,t} > p$, then $\lambda <$ OPT$(\mathcal{N}, p)$.*

Proof. If $\sum_{t=1}^{m_\lambda} \hat{p}_{\lambda,t} > p$, then suppose, to the contrary, that OPT$(\mathcal{N}, p) \le \lambda$. For each t where $1 \le t \le m_\lambda$, by (6) we have OPT$(\mathcal{N}_{\lambda,t}, p^*_{\lambda,t}) \le \lambda$, which, together with the definition of $p^*_{\lambda,t}$ and Lemma 7, implies that Algorithm 3 on λ, t, $p^*_{\lambda,t}$, and θ returns success. Hence, due to (4), we have $\hat{p}_{\lambda,t} \le p^*_{\lambda,t}$, which, together with (5), implies that $\sum_{t=1}^{m_\lambda} \hat{p}_{\lambda,t} \le p$, leading to a contradiction. □

Based on Lemma 9, we can obtain Algorithm 4, which, as shown in Lemma 10, is an ORACLE$_\rho$ for the min–max $(h = 0; D; b = 1)$-LRP with $\rho = \max\{3\theta + 0.5, 6 + \theta^{-1}\}$.

Algorithm 4. *Input*: An instance $(\mathcal{N}, p)$ of the min–max $(h = 0; D; b = 1)$-LRP, where $\mathcal{N} = (G, D, J, w, h, b)$ and $G = (V, E)$, a guessed value λ of OPT$(\mathcal{N}, p)$, a parameter $\theta \ge 1$.

Output: "λ is too low", or a p-tree schedule $\mathcal{T}$ of $\mathcal{N}$ with cost$(\mathcal{T}) \le \rho\lambda$, where $\rho = \max\{3\theta + 0.5, 6 + \theta^{-1}\}$.

1. If $D(G_{\lambda,t})$ is empty for some t, where $1 \le t \le m_\lambda$, then returns "λ is too low".
2. Let $H_\lambda = (V, \widetilde{E}_\lambda)$, where $\widetilde{E}_\lambda$ denotes the set of edges of $G[\lambda/2]$ that each have at least one endpoint in J. Let m_λ denote the number of connected components of H_λ that each have at least one customer location. Let $H_{\lambda,1}, \ldots, H_{\lambda,m_\lambda}$ denote these m_λ connected components.
3. For each t, where $1 \le t \le m_\lambda$, apply a binary search to determine $\hat{p}_{\lambda,t}$, which is the minimum integer value of p_t with $1 \le p_t$ and $p_t \le |D(G_{\lambda,t})|$, such that Algorithm 3 on λ, t, p_t, and θ returns success, and where $\hat{p}_{\lambda,t} = \infty$ if no such p_t exists.
4. If $\sum_{t=1}^{m_\lambda} \hat{p}_{\lambda,t} > p$, return "$\lambda$ is too small". Otherwise, let $\Delta = p - \sum_{t=1}^{m_\lambda} \hat{p}_{\lambda,t}$. Set $d_1, d_2, \ldots, d_\Delta$ to be Δ distinct available depots not assigned as home depots in $\mathcal{T}_t$ for $1 \le t \le m_\lambda$. Let $\mathcal{T}$ be the union of $\bigcup_{1\le t\le m_\lambda} \mathcal{T}_t$ and $\{(d_u, \{d_u\}, \emptyset) : 1 \le u \le \Delta\}$. Return success with $\mathcal{T}$. □

Lemma 10. *Algorithm 4 with a parameter $\theta \ge 1$ is an* ORACLE$_\rho$ *for the min–max $(h = 0; D; b = 1)$-LRP, where $\rho = \max\{3\theta + 0.5, 6 + \theta^{-1}\}$.*

Proof. It is easy to see that Algorithm 4 runs in polynomial time. If Algorithm 4 returns "λ is too low" in Step 1, then by (ii) of Lemma 5, $\lambda <$ OPT$(\mathcal{N}, p)$. If Algorithm 4 returns "λ is too low" in Step 4, then $\sum_{t=1}^{m_\lambda} \hat{p}_{\lambda,t} > p$. Thus, by Lemma 9, $\lambda <$ OPT$(\mathcal{N}, p)$. Otherwise, $\sum_{t=1}^{m_\lambda} \hat{p}_{\lambda,t} \le p$. Then, since Algorithm 3 on λ, t, p_t, and θ returns success, by Lemma 8, $\mathcal{T}_t$ is a $(\hat{p}_{\lambda,t})$-tree schedule of $\mathcal{N}_{\lambda,t}$ with cost$(\mathcal{T}_t) \le \rho\lambda$. Since $J \subseteq \bigcup_{1\le t\le m_\lambda} V(H_{\lambda,t})$, it is easy to verify that $\mathcal{T}$, which is constructed by combining $\mathcal{T}_t$ for $1 \le t \le m_\lambda$, is a p-tree schedule of $\mathcal{N}$ with cost$(\mathcal{T}) \le \max_{1\le t\le m_\lambda}$ cost$(\mathcal{T}_t) \le \rho\lambda$. Hence, Algorithm 4 is an ORACLE$_\rho$ for the min–max $(h = 0; D; b = 1)$-LRP. □

From Theorem 1, Lemmas 10 and 1, we can establish Theorem 4.

Theorem 4. *The min–max $(h; D; b)$-LRP has a* 13*-approximation algorithm.*

Proof. For the min–max $(h = 0; D; b = 1)$-LRP, by Lemma 10 we know that Algorithm 4 with $\theta = 2$ is an ORACLE$_{6.5}$, which implies the existence of a 13-approximation algorithm due to Theorem 1. Thus, by Lemma 1, the min–max $(h; D; b)$-LRP has a 13-approximation algorithm. □

7. Min–max (h; $D = J$; b)-LRP

In this section, we develop a 6-approximation algorithm for the min–max (h; $D = J$; b)-LRP, in which each customer location can be assigned as the home depot of a vehicle. We first present an ORACLE$_3$ for the min–max ($h = 0$; $D = J$; b)-LRP in Algorithm 5.

Algorithm 5. *Input*: An instance $(\mathcal{N}, p)$ of the min–max ($h = 0$; $D = J$; b)-LRP, where $\mathcal{N} = (G, D, J, w, h, b)$ and $G = (V, E)$; a guessed value λ of OPT$(\mathcal{N}, p)$.
Output: "λ is too low", or a p-tree schedule $\mathcal{T}$ of $\mathcal{N}$ with cost$(\mathcal{T}) \leq 3\lambda$.

1. For $1 \leq j \leq q$, where q denotes the number of connected components of $G[\lambda/2]$, compute a minimum spanning tree M_j of the j-th connected component of $G[\lambda/2]$, and let $p_j = \lfloor w(E(M_j))/(3\lambda/2) \rfloor$.
2. If $\sum_{j=1}^{q}(p_j + 1) > p$, then return "$\lambda$ is too low".
3. For each tree M_j with $1 \leq j \leq q$, construct a weighted network $\mathcal{N}_j = (M_j, V(M_j), V(M_j), w, h, b')$ with $b'(v) = \infty$ for $v \in V(M_j)$. Apply Algorithm 1 on $\mathcal{N}_j$ with the threshold equal to $3\lambda/2$ to obtain an integer m_j and an m_j-tree schedule $\mathcal{S}_j = \{(r_{ji}, T_{ji}, A_{ji}) : 1 \leq i \leq m_j\}$ of $\mathcal{N}_j$.
4. Let $\mathcal{T}'$ be the set of (d_{ji}, T_{ji}, A_{ji}) with $A_{ji} \neq \emptyset$ for $1 \leq j \leq q$ and $1 \leq i \leq m_j$, where d_{ji} denotes any vertex in A_{ji}. As shown in the proof of Lemma 12, $|\mathcal{T}'| \leq p$. Let $d_1, d_2, \ldots, d_{(p-|\mathcal{T}'|)}$ denote $(p - |\mathcal{T}'|)$ distinct available depots not in $\{d_{ji} : 1 \leq j \leq q, 1 \leq i \leq m_j, A_{ji} \neq \emptyset\}$. Let $\mathcal{T}'' = \{(d_u, \{d_u\}, \emptyset) : 1 \leq u \leq (p - |\mathcal{T}'|)\}$. Return success with $\mathcal{T} = \mathcal{T}' \cup \mathcal{T}''$. □

We next establish Lemma 11, which is proved in Appendix B, for Step 2 of Algorithm 5.

Lemma 11. *If $\sum_{j=1}^{q}(p_j + 1) > p$ in Step 2 of Algorithm 5, then $\lambda <$ OPT$(\mathcal{N}, p)$.*

Thus, we can establish Lemma 12, which guarantees the correctness of Algorithm 5.

Lemma 12. *Algorithm 5 is an* ORACLE$_3$ *for the min–max ($h = 0$; $D = J$; b)-LRP.*

Proof. It is easy to see that Algorithm 5 runs in polynomial time. Moreover, if Algorithm 5 returns "λ is too low" in Step 2, then by Lemma 11, we have $\lambda <$ OPT$(\mathcal{N}, p)$. Otherwise, $\sum_{j=1}^{q}(p_j + 1) \leq p$, and Algorithm 5 returns success with $\mathcal{T}$ in Step 4. By Property (iv) of Theorem 2, each m_j obtained in Step 3 for $1 \leq j \leq q$ satisfies $m_j \leq p_j + 1$. Thus, from Step 4, we know $|\mathcal{T}'| \leq \sum_{j=1}^{q} m_j \leq p$, which implies that $|\mathcal{T}| = p$. Moreover, for $1 \leq j \leq q$, the m_j-tree schedule $\mathcal{S}_j = \{(r_{ji}, T_{ji}, A_{ji}) : 1 \leq i \leq m_j\}$ of $\mathcal{N}_j$, obtained in Step 3, satisfies that $A_{ji} \subseteq J(T_{ji})$, and that A_{ji} for $1 \leq j \leq q$ and $1 \leq i \leq m_j$ form a partition of J. This, together with $D = J$, implies that all the available depots in $\mathcal{T}$ are distinct, which implies that the capacity of each available depot is not exceeded. Therefore, $\mathcal{T}$ is a p-tree schedule of $\mathcal{N}$. Furthermore, according to Step 1, M_j is a subgraph of $G[\lambda/2]$ for $1 \leq j \leq q$, which implies that $w_{\max}(E(M_j)) \leq \lambda/2$. Thus, due to $h(v) = 0$ for $v \in J$ and Property (iii) of Theorem 2, we obtain cost$(\mathcal{S}_j) \leq 3\lambda$ for $1 \leq j \leq q$, which, according to Step 4, implies that cost$(\mathcal{T}) \leq 3\lambda$. Hence, Algorithm 5 is an ORACLE$_3$ for the min–max ($h = 0$; $D = J$; b)-LRP. □

Hence, in a similar way to the proof of Theorem 4, from Theorem 1, Lemmas 12 and 1, we obtain directly Theorem 5.

Theorem 5. *The min–max (h; $D = J$; b)-LRP has a 6-approximation algorithm.*

8. Min–max (h; D; $b = \infty$)-LRP

In this section, we develop a 7-approximation algorithm for the min–max (h; D; $b = \infty$)-LRP, in which each available depot in D has unlimited capacity. We first present an ORACLE$_{3.5}$ for the min–max ($h = 0$; D; $b = \infty$)-LRP in Algorithm 6.

Algorithm 6. *Input*: An instance $(\mathcal{N}, p)$ of the min–max ($h = 0$; D; $b = \infty$)-LRP, where $\mathcal{N} = (G, D, J, w, h, b)$ and $G = (V, E)$; a guessed value λ of OPT$(\mathcal{N}, p)$.
Output: "λ is too low", or a p-tree schedule $\mathcal{T}$ of $\mathcal{N}$ with cost$(\mathcal{T}) \leq 3.5\lambda$.

1. For each $v \in J$, let $d(v)$ denote an available depot in D that minimizes $w(v, d(v))$.
2. If $\max_{v \in J} w(v, d(v)) > \lambda/2$, then return "$\lambda$ is too low".
3. Set $\mathcal{N}' = (G(J), J, J, w, h, b)$. Apply Algorithm 5 on $(\mathcal{N}', p)$ and λ. If Algorithm 5 returns "λ is too low", then return "λ is too low". Otherwise, Algorithm 5 returns a p-tree schedule of $\mathcal{N}'$ denoted by $\mathcal{T} = \{(d_i, T_i, A_i) : 1 \leq i \leq p\}$.
4. For $1 \leq i \leq p$, if T_i contains an available depot $d \in D$, then let $d_i' = d$ and $T_i' = T_i$; otherwise, let $d_i' = d(v)$, where v denotes any vertex of T_i, and let T_i' be a tree that consists of T_i, edge $(d(v), v)$, and $d(v)$. Return success with $\mathcal{T}' = \{(d_i', T_i', A_i) : 1 \leq i \leq p\}$. □

We show the correctness of Algorithm 6 in Lemma 13, which is proved in Appendix C.

Lemma 13. *Algorithm 6 is an* ORACLE$_{3.5}$ *for the min–max ($h = 0$; D; $b = \infty$)-LRP.*

Hence, in a similar way to the proof of Theorem 4, from Theorem 1, Lemmas 13 and 1, we obtain directly Theorem 6.

Theorem 6. *The min–max (h; D; $b = \infty$)-LRP has a 7-approximation algorithm.*

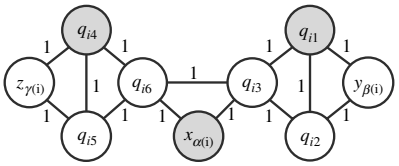


Fig. 2. A component for $M_i = (x_{\alpha(i)}, y_{\beta(i)}, z_{\gamma(i)}) \in \mathcal{M}$ in transforming 3DM to the min–max ($h = 0, |D| = p, b = 1$)-LRP, where vertices shown in gray are available depots, and the 13 edges shown in lines constitute an edge set E'_i.

9. Approximation hardness

Firstly, we establish Theorem 7, which shows an inapproximability bound of 4/3 for the min–max ($h = 0$; $|D| = p$; $b = 1$)-LRP, and therefore the bound is also valid for the min–max ($h = 0$; D; $b = 1$)-LRP and the min–max (h; D; b)-LRP.

Theorem 7. *For any $\epsilon > 0$, there is no $(4/3 - \epsilon)$-approximation algorithm for the min–max ($h = 0$; $|D| = p$; $b = 1$)-LRP, unless $\mathbb{NP} = \mathbb{P}$.*

Proof. Given a set $\mathcal{M} \subseteq X \times Y \times Z$ with $|\mathcal{M}| = m$, where $X = \{x_1, x_2, \ldots, x_n\}$, $Y = \{y_1, y_2, \ldots, y_n\}$, and $Z = \{z_1, z_2, \ldots, z_n\}$ are disjoint sets having the same number n of elements, let us consider the 3-dimensional matching (3DM) problem, which aims to decide whether or not M contains an exact matching, i.e., a subset $\mathcal{M}' \subseteq \mathcal{M}$ such that $|\mathcal{M}'| = n$ and no two elements of $\mathcal{M}'$ agree in any coordinate. It is well-known that 3DM is $\mathbb{NP}$-complete [9].

Suppose, to the contrary, that for the min–max ($h = 0$; $|D| = p$; $b = 1$)-LRP, there exists a $(4/3 - \epsilon)$-approximation algorithm $\mathcal{A}$ for $\epsilon > 0$, which has a polynomial running time. We show as follows that algorithm $\mathcal{A}$ can be used to solve 3DM in polynomial-time, which contradicts $\mathbb{NP} \neq \mathbb{P}$.

Given any 3DM instance, we construct the corresponding instance $(\mathcal{N}, p)$ of the min–max ($h = 0$; $|D| = p$; $b = 1$)-LRP as follows. Let $G = (V, E)$ be a complete undirected graph with $V = X \cup Y \cup Z \cup \bigcup_{i=1}^{m} \{q_{ij} : 1 \leq j \leq 6\}$. The set of customer locations is $J = V$. The set of available depots is $D = X \cup \bigcup_{i=1}^{m} \{q_{i1}, q_{i4}\}$. For each $M_i = (x_{\alpha(i)}, y_{\beta(i)}, z_{\gamma(i)}) \in \mathcal{M}$, where $1 \leq i \leq m$, let E'_i denote the edge set consisting of the 13 edges as depicted in Fig. 2, where $\alpha(i), \beta(i), \gamma(i) \in \{1, 2, \ldots, n\}$, and $i = 1, 2, \ldots, m$. Let $E' = \bigcup_{i=1}^{m} E'_i$. Let $w(e) = 1$ for each $e \in E'$, and $w(e) = 2$ for each $e \in E \setminus E'$. Clearly, the edge weight function w forms a metric. Let $h(v) = 0$ for all $v \in V$. We let $b(d) = 1$ for each $d \in D$. Finally, let $p = n + 2m$, which, together with $|D| = n + 2m$, implies that $|D| = p$. Thus, $(\mathcal{N}, p)$ defined above is an instance of the min–max ($h = 0$; $|D| = p$; $b = 1$)-LRP.

Since all the edge and the vertex weights are integers, the $(4/3-\epsilon)$-approximation algorithm $\mathcal{A}$ returns a feasible solution to $(\mathcal{N}, k)$ with an integer cost value. Consider the following two cases.

Case 1: The $(4/3 - \epsilon)$-approximation algorithm $\mathcal{A}$ returns a feasible solution to $(\mathcal{N}, k)$ with a cost greater than or equal to 4. In this case, we will show that the given instance of 3DM does not contain an exact matching. By contradiction, suppose that it contains an exact matching $\mathcal{M}'$. Then, we can construct the following solution $\mathcal{S}$ to instance $(\mathcal{N}, k)$: For $i = 1, 2, \ldots, m$, if $(x_{\alpha(i)}, y_{\beta(i)}, z_{\gamma(i)}) \in \mathcal{M}'$, then we let $\mathcal{S}$ include (r_{i1}, Q_{i1}, A_{i1}), (r_{i2}, Q_{i2}, A_{i2}), (r_{i3}, Q_{i3}, A_{i3}), where $r_{i1} = x_{\alpha(i)}$, $Q_{i1} = (x_{\alpha(i)} q_{i3} q_{i6} x_{\alpha(i)})$, and $A_{i1} = \{x_{\alpha(i)}, q_{i3}, q_{i6}\}$; $r_{i2} = q_{i4}$, $Q_{i2} = (q_{i4} q_{i5} z_{\gamma(i)} q_{i4})$, and $A_{i2} = \{q_{i4}, q_{i5}, z_{\gamma(i)}\}$; $r_{i3} = q_{i1}$, $Q_{i3} = (q_{i1} q_{i2} y_{\beta(i)} q_{i1})$, and $A_{i3} = \{q_{i1}, q_{i2}, y_{\beta(i)}\}$. Otherwise, let $\mathcal{S}$ include (r_{i4}, Q_{i4}, A_{i4}) and (r_{i5}, Q_{i5}, A_{i5}), where $r_{i4} = q_{i1}$, $Q_{i4} = (q_{i1} q_{i2} q_{i3} q_{i1})$, and $A_{i4} = \{q_{i1}, q_{i2}, q_{i3}\}$; $r_{i5} = q_{i4}$, $Q_{i5} = (q_{i4} q_{i5} q_{i6} q_{i4})$, and $A_{i5} = \{q_{i4}, q_{i5}, q_{i6}\}$. Since $\mathcal{M}'$ is an exact matching, it is easy to verify that the customer allocation sets in $\mathcal{S}$ form a partition of J, and that each tour in $\mathcal{S}$ contains a unique and distinct available depot in D. Thus, since $|\mathcal{S}| = 3|\mathcal{M}'| + 2(m - |\mathcal{M}'|) = n + 2m = p$, we obtain that $\mathcal{S}$ is a p-tour schedule of $\mathcal{N}$. Since all the edges of tours in $\mathcal{S}$ are in E', it is easy to see that $\text{cost}(\mathcal{S}) = 3$. This implies that $\text{OPT}(\mathcal{N}, p)$ is no greater than 3. Thus, the approximation algorithm $\mathcal{A}$ must return a feasible solution with a cost value no greater than $(4/3 - \epsilon)(3) < 4$, which is a contradiction. Therefore, the given instance of 3DM does not contain any exact matching.

Case 2: The $(4/3 - \epsilon)$-approximation algorithm $\mathcal{A}$ returns a feasible solution $\mathcal{S} = \{(r_j, Q_j, A_j) : 1 \leq j \leq p\}$ to $(\mathcal{N}, k)$ with $\text{cost}(\mathcal{S}) \leq 3$. In this case, each tour Q_j for $1 \leq j \leq p$ contains no more than three vertices. Thus, since $|V| = |J| = |W| + |X| + |Y| + 6m = 3p$, and since tours in $\mathcal{S}$ cover all the vertices, we obtain that each tour Q_j must contain exactly three vertices, and that no two tours in $\mathcal{S}$ visit the same vertices. This, together with $V = J$ and $\text{cost}(\mathcal{S}) \leq 3$, implies that $A_j = V(Q_j)$, and $w(e) = 1$ for $e \in E(Q_j)$, for $1 \leq j \leq p$.

Therefore, for each $j = 1, 2, \ldots, n$, there exists a unique and distinct tour in $\mathcal{S}$ that covers y_j, which, according to Fig. 2, must be either tour $(q_{\sigma(j)1} q_{\sigma(j)2} y_{\beta(\sigma(j))} q_{\sigma(j)1})$ or tour $(q_{\sigma(j)1} y_{\beta(\sigma(j))} q_{\sigma(j)2} q_{\sigma(j)1})$, for some $\sigma(j)$ with $\beta(\sigma(j)) = j$ and $1 \leq \sigma(j) \leq m$. Moreover, it can be seen from Fig. 2 that $\mathcal{S}$ must include $(x_{\alpha(\sigma(j))} q_{\sigma(j)3} q_{\sigma(j)6} x_{\alpha(\sigma(j))})$ or $(x_{\alpha(\sigma(j))} q_{\sigma(j)6} q_{\sigma(j)3} x_{\alpha(\sigma(j))})$ to cover $q_{\sigma(j)3}$, and must include $(q_{\sigma(j)4} q_{\sigma(j)5} z_{\gamma(\sigma(j))} q_{\sigma(j)4})$ or $(q_{\sigma(j)4} z_{\gamma(\sigma(j))} q_{\sigma(j)5} q_{\sigma(j)4})$ to cover $q_{\sigma(j)5}$. Let $\mathcal{M}' = \{(x_{\alpha(\sigma(j))}, y_{\beta(\sigma(j))}, z_{\gamma(\sigma(j))}) : j = 1, 2, \ldots, n\}$, which implies that $|\mathcal{M}'| = n$. Since no two tours in $\mathcal{S}$ visit the same vertices, no two elements of $\mathcal{M}'$ agree in any coordinates. Thus, $\mathcal{M}'$ is an exact matching in the given instance of 3DM.

Summarizing Cases 1 and 2, we conclude that the $(4/3 - \epsilon)$-approximation algorithm, which has a polynomial running time, can be used to determine whether or not the given instance of 3DM contains an exact matching. This is impossible unless $\mathbb{P} = \mathbb{NP}$. □

Next, we establish Theorem 8, which shows an inapproximability bound of 4/3 for the min–max $(h = 0; D = J; b)$-LRP and the min–max $(h = 0; D, b = \infty)$-LRP, and therefore the bound is also valid for the min–max $(h; D = J; b)$-LRP and the min–max $(h; D, b = \infty)$-LRP.

Theorem 8. *For any $\epsilon > 0$, there is no polynomial time approximation algorithm that can achieve an approximation ratio of $(4/3 - \epsilon)$ for the min–max $(h = 0; D = J; b = \infty)$-LRP, or for the min–max $(h = 0; D, b = \infty)$-LRP, unless $\mathbb{NP} = \mathbb{P}$.*

Proof. For the min–max $(h = 0; D = J; b)$-LRP, the proof is similar to that of Theorem 7, by a reduction from any instance of 3DM to an instance $(\mathcal{N}, p)$, in which we define $D = J$, and define all the other components of $(\mathcal{N}, p)$ the same as in the proof of Theorem 7.

For the min–max $(h = 0; D; b = \infty)$-LRP, the proof is also similar to that of Theorem 7, by a reduction from any instance of 3DM to an instance $(\mathcal{N}, p)$, where we define $b(d) = \infty$ for each $d \in D$, and define all the other components of $(\mathcal{N}, k)$ the same as in the proof of Theorem 7. □

10. Conclusions

Compared with the existing literature, we have developed better constant ratio approximation algorithms and better approximation hardness results for the min–max LRP and its special cases, which aim to determine both the home depots and the tours of a given set of vehicles to service customer locations, with the maximum working time of the vehicles kept to minimum.

The research can be extended in several directions. Firstly, one might try to improve the approximation algorithms or the approximation hardness results. Secondly, one might study some extensions of the min–max LRP with the cost of establishing home depots taken into consideration. Although, in the min–max objective, such a depot establishing cost cannot be directly combined with the maximum working time of the vehicles, it can be restricted by a budget constraint. The results presented in this paper can be helpful for development of constant ratio approximation algorithms for this extension, which is unknown in the existing literature. Thirdly, while we study the min–max objective in this paper, it might be more common in reality to deal with the min–sum objective, which is to minimize the sum of the costs of establishing home depots, servicing customer locations, and traveling. Although there is a rich literature concerning the min–sum objective for the location-routing problems [17], only a 2-approximation algorithm is known for a simple special case where an arbitrary number of vehicles can be used, and every vertex is an available depot with unlimited capacity [10]. It would be interesting to see to what extent constant approximation algorithms can be obtained for the min–sum objective.

Acknowledgments

We would like to thank the anonymous referees for helpful comments. This research was supported in part by Hong Kong Research Grants Council (RGC), General Research Fund (GRF) No. PolyU 532010. The second author was supported in part by the National Natural Science Foundation of China under grant 71001109 and the Fundamental Research Funds for the Central Universities.

Appendix A. Proof of Proposition 1

Consider any p-tree schedule $\mathcal{S} = \{S_i : 1 \le i \le p\}$ of $\mathcal{N}$ where $S_i = (r_i, T_i, A_i)$. For each tree T_i, we can duplicate every edge of T_i to obtain a Eulerian graph, of which a Eulerian cycle Q_i can be obtained in polynomial time [12], satisfying $w(E(Q_i)) = 2w(E(T_i))$. Let Q_i' denote the tour embedded in Q_i that skips repeated vertices, which satisfies $w(E(Q_i')) \le w(E(Q_i))$ due to the triangle inequality. Since $\mathcal{S}$ is a p-tree schedule, it is easy to verify that $\mathcal{S}' = \{S_i' : 1 \le i \le p\}$ with $S_i' = (r_i, Q_i', A_i)$ is a p-tour schedule of $\mathcal{N}$. Moreover, for $1 \le i \le p$, we have $w(E(Q_i')) + h(A_i) \le w(E(Q_i)) + h(A_i) \le 2w(E(T_i)) + 2h(A_i) \le 2\text{cost}(\mathcal{S})$, which implies that $\text{cost}(\mathcal{S}') \le 2\text{cost}(\mathcal{S})$. □

Appendix B. Proof of Lemma 11

Consider the case when $\sum_{j=1}^{q}(p_j + 1) > p$ in Step 2 of Algorithm 5. Suppose, to the contrary, that there exists a p-tour schedule $\mathcal{Q} = \{(r_i, Q_i, A_i) : 1 \le i \le p\}$ of $\mathcal{N}$ with $\text{cost}(\mathcal{Q}) \le \lambda$. For each $1 \le j \le q$, consider the minimum spanning tree M_j of the j-th connected component of $G[\lambda/2]$, obtained in Step 1 of Algorithm 5. Let $\mathcal{Q}(M_j)$ denote the set of $(r_i, Q_i, A_i) \in \mathcal{Q}$ with Q_i containing at least one vertex of M_j. Let $p_j^* = |\mathcal{Q}(M_j)|$. Since $J(M_j)$ is not empty, we have $p_j^* \ge 1$.

For each $(r_i, Q_i, A_i) \in \mathcal{Q}$, where $1 \le i \le q$, by Proposition 2 we know that all the vertices of Q_i must belong to the same connected component of $G[\lambda/2]$. Thus, $\mathcal{Q}(M_j)$ for $1 \le j \le q$ are disjoint to each other, and constitute a partition of $\mathcal{Q}$, which implies:

$$\sum_{j=1}^{q} p_j^* = \sum_{j=1}^{q} |\mathcal{Q}(M_j)| = p. \tag{B.1}$$

Moreover, for each M_j, where $1 \le j \le m$, since $D = J$, we have $V(M_j) = \bigcup_{\forall (r_i, Q_i, A_i) \in \mathcal{Q}(M_j)} V(Q_i)$. Thus, connecting all the tours Q_i of $(r_i, Q_i, A_i) \in \mathcal{Q}(M_j)$ by adding at most $(p_j^* - 1)$ edges of $G[\lambda/2]$, we obtain a subgraph of the j-th connected component of $G[\lambda/2]$ that covers all the vertices of M_j. Since M_j is a minimum spanning tree of the j-th connected component of $G[\lambda/2]$, we obtain:

$$w(E(M_j)) \le w(E(\mathcal{Q}(M_j))) + (p_j^* - 1)\lambda/2. \tag{B.2}$$

Therefore, since $\text{cost}(\mathcal{Q}) \le \lambda$ and $h(v) = 0$ for $v \in J$, we have $w(E(\mathcal{Q}(M_j))) \le p_j^*\lambda$. Due to $\lambda > 0$ and (B.2) we obtain $2w(E(M_j))/(3\lambda) < p_j^*$. Thus, $p_j + 1 = \lfloor 2w(E(M_j))/3\lambda \rfloor + 1 \le p_j^*$, because p_j and p_j^* are integers. By (B.1) we obtain $\sum_{j=1}^{q}(p_j + 1) \le \sum_{j=1}^{q} p_j^* = p$, leading to a contradiction. □

Appendix C. Proof of Lemma 13

If Algorithm 6 returns "λ is too low" in Step 2, then there exists $v \in J$ such that no available depot $d \in D$ satisfies $w(d, v) \le \lambda/2$. Thus, by Proposition 2, we have $\lambda < \text{OPT}(\mathcal{N}, p)$.

If Algorithm 6 returns "λ is too low" in Step 3, then by Lemma 12, $\lambda < \text{OPT}(\mathcal{N}', p)$. Since $G(J)$ of $\mathcal{N}'$ is a subgraph of G of $\mathcal{N}$, we can transform each p-tour schedule of $\mathcal{N}$ to a p-tour schedule of $\mathcal{N}'$, by skipping all the vertices that are in G but not in $G(J)$, with the cost not increased, due to the triangle inequality. Thus, we obtain $\lambda < \text{OPT}(\mathcal{N}', p) \le \text{OPT}(\mathcal{N}, p)$.

Otherwise, in Step 3 of Algorithms 6 and 5 must return a p-tree schedule $\mathcal{T}$ of $\mathcal{N}'$, with $\text{cost}(\mathcal{T}) \le 3\lambda$ due to Lemma 12. According to Step 4, we have $A_i \subseteq J(T_i) \subseteq J(T_i')$ and $r_i' \in V(T_i') \cap D$ for $1 \le i \le p$, and $|\mathcal{T}'| = |\mathcal{T}| = p$. Thus, $\mathcal{T}'$ is a p-tree schedule of $\mathcal{N}$. Moreover, for each $(r_i', T_i', A_i) \in \mathcal{T}'$, its weight is either equal to the weight of $(r_i, T_i, A_i) \in \mathcal{T}$, which is not greater than 3λ, or equal to the weight of (r_i, T_i, A_i) plus the weight of edge $(v, d(v))$ for some $v \in V(T_i)$, which is not greater than $(3\lambda + \lambda/2)$. Thus, $\text{cost}(\mathcal{T}') \le 3.5\lambda$.

Moreover, since Algorithm 6 runs in polynomial time, we obtain that it is an $\text{ORACLE}_{3.5}$. □

References

[1] E.M. Arkin, R. Hassin, A. Levin, Approximations for minimum and min–max vehicle routing problems, Journal of Algorithms 59 (1) (2006) 1–18.
[2] I. Averbakh, O. Berman, A heuristic with worst-case analysis for minimax routing of two travelling salesmen on a tree, Discrete Applied Mathematics 68 (1–2) (1996) 17–32.
[3] I. Averbakh, O. Berman, $(p-1)/(p+1)$-approximate algorithms for p-traveling salesmen problems on a tree with minmax objective, Discrete Applied Mathematics 75 (3) (1997) 201–216.
[4] I. Averbakh, O. Berman, Minmax p-traveling salesmen location problems on a tree, Annals of Operations Research 110 (1) (2002) 55–68.
[5] R.T. Berger, C.R. Coullard, M.S. Daskin, Location-routing problems with distance constraints, Transportation Science 41 (1) (2007) 29–43.
[6] A.M. Campbell, D. Vandenbussche, W. Hermann, Routing for relief efforts, Transportation Science 42 (2) (2008) 127–145.
[7] G. Even, N. Garg, J. Könemann, R. Ravi, A. Sinha, Minmax tree covers of graphs, Operations Research Letters 32 (4) (2004) 309–315.
[8] G.N. Frederickson, M.S. Hecht, C.E. Kim, Approximation algorithms for some routing problems, SIAM Journal on Computing 7 (1978) 178–193.
[9] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, W.H. Freeman & Co., New York, USA, 1979.
[10] M.X. Goemans, D.P. Williamson, A general approximation technique for constrained forest problems, in: Proceedings of the Third Annual ACM–SIAM Symposium on Discrete Algorithms, 1992, pp. 307–316.
[11] Y. Karuno, H. Nagamochi, 2-approximation algorithms for the multi-vehicle scheduling problem on a path with release and handling times, Discrete Applied Mathematics 129 (2–3) (2003) 433–447.
[12] B. Korte, J. Vygen, Combinatorial Optimization: Theory and Algorithms, Springer, 2002.
[13] H. Min, V. Jayaraman, R. Srivastava, Combined location-routing problems: a synthesis and future research directions, European Journal of Operational Research 108 (1) (1998) 1–15.
[14] H. Nagamochi, Approximating the minmax rooted-subtree cover problem, IEICE Transactions on Fundamentals of Electronics, Communications and Computer Science 88 (5) (2005) 1335–1338.
[15] H. Nagamochi, K. Okada, A faster 2-approximation algorithm for the minmax p-traveling salesmen problem on a tree, Discrete Applied Mathematics 140 (1–3) (2004) 103–114.
[16] H. Nagamochi, K. Okada, Approximating the minmax rooted-tree cover in a tree, Information Processing Letters 104 (5) (2007) 173–178.
[17] G. Nagy, S. Salhi, Location-routing: issues, models and methods, European Journal of Operational Research 177 (2) (2007) 649–672.
[18] Z. Xu, Q. Wen, Approximation hardness of min–max tree covers, Operations Research Letters 38 (3) (2010) 169–173.
[19] Z. Xu, L. Xu, C.-L. Li, Approximation results for min–max path cover problems in vehicle routing, Naval Research Logistics 57 (8) (2010) 728–748.