Mathematics of Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

Capacitated Vehicle Routing with Nonuniform Speeds

Inge Li Gørtz, Marco Molinaro, Viswanath Nagarajan, R. Ravi

MATHEMATICS OF OPERATIONS RESEARCH
Vol. 41, No. 1, February 2016, pp. 318–331
ISSN 0364-765X (print) | ISSN 1526-5471 (online)

http://dx.doi.org/10.1287/moor.2015.0729
© 2016 INFORMS

Capacitated Vehicle Routing with Nonuniform Speeds

Inge Li Gørtz
Technical University of Denmark, 2800 Kgs. Lyngby, Denmark, ilg@imm.dtu.dk

Marco Molinaro
Georgia Institute of Technology, Atlanta, Georgia 30332, marco.molinaro@isye.gatech.edu

Viswanath Nagarajan
Department of Industrial and Operations Engineering, University of Michigan, Ann Arbor, Michigan 48109, viswa@umich.edu

R. Ravi
Tepper School of Business, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213, ravi@andrew.cmu.edu

The capacitated vehicle routing problem (CVRP) involves distributing identical items from a depot to a set of demand locations using a single capacitated vehicle. We introduce the heterogeneous capacitated vehicle routing problem, a generalization of CVRP to the setting of multiple vehicles having nonuniform speeds, and present for it a constant-factor approximation algorithm.

Our main contribution is an approximation algorithm for the heterogeneous traveling salesman problem, which is the special case of heterogeneous CVRP with uncapacitated vehicles. Given a metric denoting distances between vertices, a depot r containing k vehicles having respective speeds $\{\lambda_i\}_{i=1}^k$, the objective in heterogeneous TSP is to find a tour for each vehicle (starting and ending at r) so that every vertex is covered in some tour and the maximum completion time is minimized; the completion time of a vehicle is the distance traveled divided by its speed.

Our algorithm relies on a new approximate minimum spanning tree construction called *Level-Prim*, which is related to but different from *Light Approximate Shortest-path Trees*. We also extend the widely used tour-splitting technique to nonuniform speeds, using ideas from the 2-approximation algorithm for scheduling in unrelated machines.

Keywords: vehicle routing; traveling salesman problem; approximation algorithms
MSC2000 subject classification: Primary: 90B06; secondary: 68W25
OR/MS subject classification: Primary: vehicle routing; secondary: approximation algorithm
History: Received November 3, 2012; revised February 25, 2014. Published online in *Articles in Advance* January 8, 2016.

1. Introduction. The capacitated vehicle routing problem (CVRP) is an extensively studied combinatorial optimization problem (see, e.g., the book by Toth and Vigo [21] and references therein). CVRP is defined on a metric space (V, d), where V is a finite set of locations/vertices and $d\colon V \times V \to \mathbb{R}_+$ a distance function. There is a depot vertex $r \in V$ that contains an infinite supply of an item, and each vertex $u \in V$ demands some units q_u of this item. A single vehicle of capacity $Q \geq 0$ is used to distribute the items. The objective is to find a minimum length tour of the vehicle that satisfies all demands, subject to the constraint that the vehicle carries at most Q units at any time.

CVRP is closely related to the Traveling Salesman Problem (TSP). It is clear that CVRP reduces to TSP in the absence of the capacity constraint. More interestingly, a reverse relation is also known—essentially the best known approximation algorithm for CVRP (Haimovich and Kan [16]) achieves a guarantee of $\rho + 1$, where ρ is the best approximation ratio for TSP.

In practice, it is natural to have a fleet of *multiple* vehicles that can run in parallel. The objective can then be either to minimize the sum of completion times of all the vehicles or to minimize the maximum completion time over all vehicles (also called the makespan). Furthermore, the vehicles can all be identical (same speed) or heterogeneous (have different speeds). It can easily be seen that the total completion time objective always reduces to the usual CVRP on a single vehicle (having maximum speed), and constant-factor approximation algorithms readily follow.

The *Heterogeneous Capacitated Vehicle Routing Problem* (HetCVRP) addresses the *makespan* objective. Here, a fleet of k vehicles with *nonuniform speeds* and uniform capacity Q is initially located at the depot vertex r. A valid routing consists of a tour for each vehicle subject to the capacity constraint such that all demands are satisfied. The objective in HetCVRP is to minimize the makespan, i.e., the maximum completion time of the routing; the completion time of a vehicle equals the distance traveled divided by its speed. Our main result is a constant-factor approximation algorithm for HetCVRP.

Our contributions are twofold:

• We extend the tour-splitting technique to the setting of nonuniform speed vehicles. The tour-splitting approach has been very useful in obtaining approximation algorithms for a number of vehicle routing problems with uniform speeds (see a more detailed discussion below).

- We introduce a new approximate Minimum Spanning Tree called *Level Prim* that is used in the "tour splitting" step and that might be of some independent interest.

Most of our algorithmic ideas, in fact, lie in solving the special case of HetCVRP when there is no capacity constraint. This problem, which we call Heterogeneous Traveling Salesman Problem (HetTSP), is a generalization of the Traveling Salesman Problem in the presence of multiple nonuniform speed vehicles. The formal definitions appear next.

1.1. Problem definition. The input to the HetTSP consists of a metric (V, d) denoting distances between vertices, a depot $r \in V$, and k vehicles with speeds $\{\lambda_i\}_{i=1}^k$. The distance function d: $V \times V \to \mathbb{R}_+$ is symmetric and satisfies triangle inequality. We assume (without loss of generality) by scaling that all speeds are greater than or equal to one. The objective is to find tours $\{\tau_i\}_{i=1}^k$ (starting and ending at r) for each vehicle so that every vertex is covered in some tour and we minimize the makespan:

$$\max_{i=1}^{k} \frac{d(\tau_i)}{\lambda_i}.$$

For each $i \in \{1, \ldots, k\}$, $d(\tau_i)$ denotes the length of tour τ_i, so the time taken by vehicle i is $d(\tau_i)/\lambda_i$.

The HetCVRP has the same input as above, and in addition there is a capacity Q and demands $\{q_v \in \mathbb{Z}_+ : v \in V\}$ of an identical item. There is an infinite supply of this item located at the depot. A solution to HetCVRP consists of tours $\{\sigma_i\}_{i=1}^k$ (starting and ending at r) for each vehicle so that each vertex v receives q_v units of the item, and each vehicle carries at most Q items at any point in time. Each tour might visit the depot multiple times to refill items. The objective is again to minimize the *maximum completion time*, $\max_{i=1}^k d(\sigma_i)/\lambda_i$.

We study the *unsplit-delivery* version of HetCVRP (Altinkemer and Gavish [1]), where the entire demand at a vertex must be served in a single visit; thus we assume that $\max_{v \in V} q_v \le Q$. It is clear that the optimal value under unsplit-deliveries is at least the optimal value under the less restrictive *split-delivery* model (Altinkemer and Gavish [2]), where the demand at a vertex may be served by multiple visits. We give an algorithm for unsplit-delivery HetCVRP that has the stronger guarantee of achieving a constant factor approximation relative to the optimal split-delivery routing.

1.2. Previous techniques. Here we discuss some relevant previously used techniques.

Tour-splitting solutions: A popular heuristic approach for vehicle routing problems is to obtain a solution by partitioning a traveling salesman tour or a minimum spanning tree. To illustrate this, we first outline a constant-factor approximation algorithm for HetTSP with *uniform* speeds. By scaling, we may assume that all speeds are one. Let OPT denote the optimal value (i.e., makespan) of the given uniform HetTSP instance. Notice that the union of the k tours in any solution connects all vertices, and hence the minimum spanning tree (MST) has length at most $k \cdot OPT$. Now consider an MST, and take an Euler tour C by doubling edges; clearly, the length $d(C) \le 2k \cdot OPT$. Next, partition the vertices on C into k connected segments, each of length at most $d(C)/k \le 2 \cdot OPT$. Finally, the solution tour for the ith vehicle is obtained by connecting both endpoints of the ith segment of C to the depot. Observe that twice the distance from the depot to any vertex is a lower bound on OPT, so the length of each tour is at most $3 \cdot OPT$; hence, this solution is a 3-approximation. We remark that this can be extended to obtain an $O(1)$-approximation algorithm for HetCVRP with uniform speeds (e.g., using Theorem 4 in §3).

At a high level, this strategy has two main components: (1) Partitioning an MST into manageably sized connected parts and (2) assigning these parts to vehicles. This idea—which was already present in the 1970s—is the central piece of many heuristics and approximation algorithms for vehicle routing problems; e.g., Frederickson et al. [14], Haimovich and Kan [16], Altinkemer and Gavish [1, 2], Even et al. [13], Arkin et al. [3], Gupta et al. [15]. However, it is not clear how to directly employ this technique in the presence of nonuniform speeds. This is because the two main components now need some correlation: the part of the MST that is assigned to a slow vehicle must also be relatively close to the depot in order to be reachable by this vehicle.

Set-cover based solutions: A different approach is to use a set-covering formulation of the vehicle routing problem. We illustrate this on the general HetTSP (with nonuniform speeds), where it leads to a logarithmic approximation ratio. Suppose that the algorithm knows a value B such that $B/2 \le OPT \le B$ (we will run the algorithm over all choices for B and output the best solution found). If each vehicle of speed s is given a length budget of $s \cdot B \ge s \cdot OPT$, then the vehicles can collectively cover all vertices. Using an $O(1)$-approximation algorithm for the *orienteering problem* (Blum et al. [6], Chekuri et al. [9]), one can obtain for each vehicle i, a tour of length at most $\lambda_i \cdot B$ containing (approximately) the maximum number of vertices. This can be used within a maximum-coverage framework (see, e.g., Chekuri and Kumar [8]), to obtain one tour for each vehicle so that a constant fraction of all vertices are covered, and the tour of the ith vehicle has length at most $\lambda_i \cdot B$. To cover all vertices, this "maximum coverage" step is repeated $O(\log n)$ times, resulting in a solution to HetTSP of makespan $O(\log n) \cdot OPT$.

The intrinsic problem of this approach is that it is too general—in fact, the above algorithm also yields a logarithmic approximation ratio in the setting where the travel-time metric faced by each vehicle is arbitrary (instead of being scaled by its speed), and this generalization of HetTSP can be shown to be $\Omega(\log n)$ hard to approximate, by a reduction from set-cover. It is therefore unclear whether the performance of this set-covering based approach can be improved for HetTSP.

1.3. Results, techniques and outline. We extend the tour-splitting approach to nonuniform speeds and obtain the following result.

Theorem 1. *There are constant-factor approximation algorithms for* HetTSP *and* HetCVRP.

To apply the tour-splitting strategy with nonuniform speed vehicles, we modify the requirements of the two components: (1) partitioning the MST and (2) assigning vehicles.

First, we specify conditions that guarantee that a collection of connected parts is "assignable"; that is, each vehicle can visit the vertices of the parts assigned to it within time $O(OPT)$. These conditions (formalized in Definition 1) are based on the 2-approximation algorithm for *scheduling in unrelated machines* by Lenstra et al. [18]. It turns out to be important in this definition that each connected part be "r-rooted," i.e., contains the root r. Lemma 2 shows that any assignable collection of r-rooted subtours can be used to obtain a constant-factor approximate solution to HetTSP. These details appear in §2.2.

Secondly, instead of partitioning an MST as in usual applications of tour-splitting, we use more structured spanning trees that we call Level-Prim trees (to obtain an assignable r-rooted collection). Consider partitioning the vertices into *levels* according to their distance from r, where the ith level includes all vertices at distance between 2^{i-1} and 2^i. The Level-Prim tree is simply the tree resulting from running Prim's MST algorithm with the restriction that all vertices in a level are spanned before including vertices from the next level. We show in §2.3 that the Level-Prim tree has two important properties; informally, these are (i) the vertices along every root-leaf path are monotonically nondecreasing in level and (ii) for every suffix $\{j, j+1, \dots\}$ of levels, the subgraph induced on those vertices costs at most $O(1)$ times their induced MST. The first condition is the departing point from MSTs. The second property is related to the assignability conditions in Definition 1 and guarantees that we can decompose a Level-Prim tree into an assignable collection. These properties are formalized in Theorem 2.

The Level-Prim construction combines aspects of both MST and shortest-path distances from a root, so it is not surprising that this structure is related to *Light Approximate Shortest-Path Trees* (LAST) introduced by Khuller et al. [17]. Indeed, we use the existence of a suitably defined LAST in proving Theorem 2. We remark, however, that the properties guaranteed by LASTs are not sufficient for our purposes (see §2.3).

The third main component of our algorithm for HetTSP is decomposing Level-Prim into an assignable collection of r-rooted subtours. Roughly speaking, we partition the edges of Level-Prim into subtrees while ensuring that each subtree consisting of level-i vertices also has length $\Theta(2^i)$. This partition, which relies on the two properties of Level-Prim, yields a collection of *unrooted* subtours that is "assignable." Moreover, each level-i subtour can be connected to the root r using an extra edge of length at most 2^i that can be charged to the Level-Prim subtree itself since it has length $\Omega(2^i)$. Altogether, we obtain an assignable collection of r-rooted subtours as required by Definition 1. This step appears in §2.4.

Finally, to obtain an approximation algorithm for HetCVRP, we reduce this problem to approximating HetTSP in a suitably modified metric space. The new distance function encodes any additional trips to and from the root that a vehicle has to make when it runs out of capacity. The exact transformation is presented in §3.

1.4. Related work. For the CVRP, the best known approximation ratio (Haimovich and Kan [16]) is essentially $\rho+1$ where ρ is the best guarantee for TSP. The current best values for ρ are $\rho = \frac{3}{2}$ for general metrics (Christofides [10]) and $\rho = 1+\epsilon$ (for any constant $\epsilon > 0$) for constant dimensional Euclidean metrics (Arora [4], Mitchell [19]). This has been improved slightly to $1+\rho\cdot(1-1/Q)-(1/3)Q^3$ when $Q \geq 3$ (Bompadre et al. [7]). Recently, Das and Mathieu [12] gave a quasi-polynomial time approximation scheme for CVRP on the Euclidean plane.

Several variants of TSP have been studied, most of which have a min-sum objective. One related problem with min-max objective is *nurse station location* (Even et al. [13]), where the goal is to obtain a collection of trees (each rooted at a distinct depot) such that all vertices are covered and the maximum tree length is minimized. Even et al. [13] gave a 4-approximation algorithm for this problem. This is based on partitioning the

MST and assigning to trees along the lines of §1.2; their second step, however, involves a nontrivial bipartite matching subproblem.

In proving the properties of Level-Prim, we use *Light Approximate Shortest-Path Trees* introduced by Khuller et al. [17], building on the work on shallow-light trees of Awerbuch et al. [5]. An (α, β)-LAST is a rooted tree that has (a) length at most β times the MST and (b) the distance from any vertex to the root (along the tree) is at most α times the distance in the original metric. Khuller et al. [17] showed that every metric has an $(\alpha, 1+2/(\alpha-1))$-LAST (for any $\alpha > 1$) and this is best possible.

The first phase of our algorithm uses some ideas from scheduling on unrelated machines (Lenstra et al. [18]), which also has a min-max objective. In this problem, there is a set of jobs and machines, where each job j has processing time p_{ij} on machine i; the goal is to assign jobs to machines while minimizing the maximum completion time. Lenstra et al. [18] gave an LP-based 2-approximation algorithm for this problem.

Notation. We let $G = (V, E)$ be the complete graph on vertices V with edge weights corresponding to the distance function d. For any set $F \subseteq E$ of edges, we set $d(F) = \sum_{e\in F} d_e$. Whenever we consider a subgraph H of G, the edges of H inherit the weight that they have in G (i.e., the weight of edge (u, v) is $d(u, v)$). We use $d_H(u, v)$ to denote the shortest-path distance in H between u and v.

Given any (multi)graph H and a subset U of its vertices, $H[U]$ denotes the subgraph induced on U and H/U denotes the graph obtained by contracting vertices U to a single vertex (we retain parallel edges).

For any $s \geq 0$, we let $\Lambda(s) = \sum_{j\in[k]:\, \lambda_j \geq s} \lambda_j$ denote the sum of those speeds that are at least s.

2. Algorithm for HetTSP.

Our algorithm involves the following components. First, we partition the vertices into levels according to geometrically increasing distances from the depot. Second, we compute an approximate minimum spanning tree (the Level-Prim) based on this partition. Third, we decompose the Level-Prim tree into a suitable collection of subtours. Finally, we show that these subtours can be assigned to the vehicles to obtain a solution of near-optimal makespan. The algorithm is outlined below for future reference; we describe the precise details in the following subsections.

Algorithm 1 (Algorithm outline for HetTSP)

```
 1: Initialize $M \leftarrow \max_v(d(r,v)/\max_i \lambda_i)$.        // Initial (low) estimate of OPT.
 2: loop
 3: Using the current value of $M$, classify vertices into levels $V_0, V_1, \ldots$ (§2.1).
 4: Using these levels, obtain the approximate spanning tree $\mathcal{H} \leftarrow$ Level-Prim$(G)$ (§2.3).
 5: if inequality (1) is satisfied by $\mathcal{H}$, then
 6:                               // Estimate $M$ is not too low.
 7: Decompose $\mathcal{H}$ into an assignable collection $\{\mathcal{T}_i\}_i$ of subtours (§2.4).
 8: Using $\{\mathcal{T}_i\}_i$, obtain a feasible solution to HetTSP with makespan $O(M)$ (§2.2).
 9: Exit the procedure.        // Approximate solution found.
10: else
11: $M \leftarrow (1+\delta)M$.        // Increase estimate of OPT, $\delta > 0$ is a small constant.
```

The variable M can be thought of as a "guess" of OPT that starts at a low value and is increased in every iteration of the main loop; the **if** condition in the main loop "verifies" this guess. Therefore, it is instructive to view M as a good approximation to the optimal makespan.

In the description of the algorithm, for concreteness we compute various constant factors: this leads to a final approximation ratio of 90 for HetTSP. However, we have not tried to optimize these constant factors since that is not the focus of this paper.

2.1. Classifying into levels.

We now describe the classification of vertices used in step 3 of the algorithm. Given M and $\epsilon \geq 1$ (the precise value to be fixed later), partition the vertices V according to their distance to r:

$$V_0 = \{u \in V\colon\ d(r,u) \leq M\}, \quad \text{and}$$
$$V_i = \{u \in V\colon\ (1+\epsilon)^{i-1}M < d(r,u) \leq (1+\epsilon)^i M\}, \qquad \text{for all } i \geq 1.$$

The vertices in V_i are referred to as *level* i vertices. For any $i \geq 0$, we use $V_{\leq i}$ as a shorthand for $\bigcup_{j=0}^{i} V_j$ and similarly $V_{<i} = \bigcup_{j=0}^{i-1} V_j = V_{\leq i-1}$. For notational convenience, we set $V_{<0} = \{r\}$.

We also define the *level of an edge* $(u, v) \in E$ as the larger of the levels of u and v. For each $i \geq 0$, E_i denotes the edges in E of level i. We use the notation $E_{\leq i} = \bigcup_{j=0}^{i} E_j$ and $E_{\geq i} = \bigcup_{j\geq i} E_j$.

2.2. Assignable subtours. Here we characterize some properties that imply good near-optimal solutions to HetTSP. We start with a lower bound on the optimal value of any HetTSP instance.

Lemma 1 (Lower Bound). *Assume that $M \geq OPT$. Then for each level $l \geq 0$,*

$$MST(G/V_{<l}) \leq M \cdot \Lambda((1+\epsilon)^{l-1}),$$

where $MST(G/V_{<l})$ is the length of the minimum spanning tree in the graph obtained by contracting vertices $V_{<l}$ and $\Lambda((1+\epsilon)^{l-1})$ is the total of speeds that exceed $(1+\epsilon)^{l-1}$.

Proof. Consider an optimal solution for HetTSP and let E^* be the set of edges traversed by vehicles in this solution; label each edge in E^* by the vehicle that traversed it. Clearly E^* connects all vertices to the root r.

Observe that in an optimal solution, only vehicles having speed at least $(1+\epsilon)^{l-1}$ can even reach any vertex in $V_{\geq l}$. This is because vertices in $V_{\geq l}$ are located at distance at least $(1+\epsilon)^{l-1}M$ from the depot, and a vehicle of speed s travels distance at most $s \cdot OPT \leq s \cdot M$. Thus every edge in $E^* \cap E_{\geq l}$ must be labeled by some vehicle of speed at least $(1+\epsilon)^{l-1}$. Now, since OPT denotes the optimal makespan, the total distance covered by vehicles having speed at least $(1+\epsilon)^{l-1}$ is upper bounded by $OPT \cdot \Lambda((1+\epsilon)^{l-1}) \leq M \cdot \Lambda((1+\epsilon)^{l-1})$. This implies that $d(E^* \cap E_{\geq l}) \leq M \cdot \Lambda((1+\epsilon)^{l-1})$.

On the other hand, since E^* connects all vertices, $E^* \cap E_{\geq l}$ contains a spanning tree of $G/V_{<l}$. Thus we have $MST(G/V_{<l}) \leq d(E^* \cap E_{\geq l}) \leq M \cdot \Lambda((1+\epsilon)^{l-1})$. □

Along lines of the tour-splitting approach, our algorithm first obtains a collection of subtours that cover all vertices and then assigns these subtours to vehicles. Motivated by the lower bound in Lemma 1, we define properties (on the subtours) that correspond to near-optimal solutions to HetTSP. We want to guarantee that the subtours can be assigned to vehicles so that each vehicle completes its subtours in time $O(M)$. Each subtour is a cycle that contains the depot r (also called *r-cycle*).

Definition 1 (Assignable Subtours). For each $i \in \mathbb{Z}_{\geq 0}$ let $\mathcal{T}_i$ be a collection of r-cycles. Then $\{\mathcal{T}_i\}_{i\geq 0}$ is called (α, β)-*assignable* if it covers all vertices V and has the following properties.

1. For each $i \geq 0$ and every $T \in \mathcal{T}_i$: $d(T) \leq \alpha(1+\epsilon)^i M$.
2. For each $i \geq 0$: $\sum_{j\geq i} d(\mathcal{T}_j) \leq \beta M \cdot \Lambda((1+\epsilon)^{i-1})$.

Above, for any collection $\mathcal{T}_j$ of r-cycles, $d(\mathcal{T}_j) = \sum_{T\in\mathcal{T}_j} d(T)$ denotes its total length.

In the intended HetTSP solution, subtours in $\mathcal{T}_i$ will only be handled by vehicles of speed at least $(1+\epsilon)^{i-1}$. We view each vehicle of speed s as having a "length capacity" of $s \cdot M$, which corresponds to a bound on the total length of subtours assigned to it. The first condition implies that each subtour in $\mathcal{T}_i$ can be assigned to a vehicle of speed $(1+\epsilon)^i$ so as to be completed in time αM. The second condition guarantees that the total length of subtours $\{\mathcal{T}_j\}_{j\geq i}$ (that are targeted by vehicles of speed at least $(1+\epsilon)^{i-1}$) is at most β times the total "length capacity" of those vehicles. These minimal conditions are enough to eventually assign all subtours to vehicles while guaranteeing small makespan. The following lemma describes step 8 of Algorithm 1.

Lemma 2. *Given an (α, β)-assignable collection $\{\mathcal{T}_i\}_{i\geq 0}$ of r-cycles, we can obtain in polynomial time a solution for* HetTSP *with makespan at most $((1+\epsilon)\alpha + \beta)M$.*

Proof. To prove this lemma, we show that the second condition in Definition 1 (by Hall's theorem) guarantees the existence of a fractional assignment of subtours to vehicles, where each vehicle incurs load at most βM. Then using the first condition and a result on scheduling on unrelated machines (Lenstra et al. [18]), we round this fractional assignment into an integral one while increasing the load on each vehicle by at most $(1+\epsilon)\alpha M$.

Consider the bipartite graph H whose left side contains one node for each subtour in $\{\mathcal{T}_i\}_{i\geq 0}$ and whose right side contains one node for each vehicle. (We identify the nodes with their respective subtours/vehicles.) The edge set $E(H)$ contains an edge between subtour $T \in \mathcal{T}_i$ and a vehicle of speed s if and only if $s \geq (1+\epsilon)^{i-1}$.

Fractional assignment. Consider the following *b-matching problem* (Schrijver [20]) on the graph H. For each subtour $T \in \mathcal{T}_i$, we set $b(T) := d(T)$. For each vehicle u of speed s, we set $b(u) := \beta s \cdot M$. A left-saturating b-matching is an assignment x: $E(H) \to \mathbb{R}_+$ such that

$$\sum_{u:\text{vehicle},\ (T,u)\in E(H)} x_{T,u} = b(T), \quad \text{for every subtour } T$$

$$\sum_{T:\text{subtour},\ (T,u)\in E(H)} x_{T,u} \leq b(u), \quad \text{for every vehicle } u$$

Notice that this gives a fractional assignment of subtours to vehicles. We claim that there is indeed such a b-matching in H. Using a standard generalization of Hall's Theorem (e.g., see page 54 of Cook et al. [11]), H has a feasible b-matching if and only if for every subset U of subtours, $\sum_{T\in U} b(T) \le \sum_{w\in N(U)} b(w)$, where $N(U) = \{w\colon \exists T \in U, (T, w) \in E(H)\}$ is the neighborhood of U. However, by the structure of H, it suffices to verify this condition for sets U that are equal to $\bigcup_{j\ge i} \mathcal{T}_j$ for some i. To see that all other inequalities are dominated by those coming from such sets, first notice that if U contains a subtour in $\mathcal{T}_i$, then $N(U)$ already contains all vehicles of speed at least $(1+\epsilon)^{i-1}$. Adding to U all the subtours in $\bigcup_{j\ge i} \mathcal{T}_j$ does not change its neighborhood and thus leads to a dominating inequality. Using this revised condition, H has a b-matching if and only if

$$\sum_{j\ge i} d(\mathcal{T}_j) \le \sum_{w\in N(\bigcup_{j\ge i}\mathcal{T}_j)} b(w) = \beta M \cdot \Lambda((1+\epsilon)^{i-1}), \quad \forall\, i \ge 0.$$

Since this is exactly the second condition in Definition 1, it follows that H indeed has a b-matching x; this can also be obtained in polynomial time using any maximum flow algorithm (Cook et al. [11]).

Scheduling unrelated machines. We now round the fractional assignment x to an integral assignment where each subtour is assigned to exactly one vehicle. Consider an instance of scheduling on *unrelated machines* (Lenstra et al. [18]) with each subtour as a "job" and each vehicle as a "machine." Define the processing time of subtour T on vehicle u (of speed s) as $p_{T,u} := d(T)/s$ if $(T, u) \in E(H)$. The goal is to assign jobs to machines so that the maximum machine load is minimized; the load on a machine is the total processing time of jobs that are assigned to it.

Let $y_{T,u} := x_{T,u}/d(T)$ denote the fraction of subtour T assigned to vehicle u in the b-matching x. For any vehicle u having speed s, we have

$$\sum_T y_{T,u} \cdot p_{T,u} = \sum_{(T,u)\in E(H)} y_{T,u} \cdot \frac{d(T)}{s} = \sum_{(T,u)\in E(H)} \frac{x_{T,u}}{s} \le \frac{b(u)}{s} = \beta M.$$

Consider any edge $(T, u) \in E(H)$ with $T \in \mathcal{T}_i$ and u having speed s. Using the first property in Definition 1, we have $d(T) \le \alpha(1+\epsilon)^i M$. By construction of the edges of H, since $T \in \mathcal{T}_i$, vehicle u has speed $s \ge (1+\epsilon)^{i-1}$. Thus $p_{T,u} = d(T)/s \le (1+\epsilon)\alpha M$, for all $(T, u) \in E(H)$.

Thus y is a feasible solution to the natural LP relaxation for the unrelated machine scheduling instance, with makespan βM and maximum processing time $(1+\epsilon)\alpha M$. Theorem 1 of Lenstra et al. [18] then asserts that y can be rounded into an integral assignment of subtours to vehicles such that the load on any vehicle is at most $((1+\epsilon)\alpha + \beta)M$. $\square$

2.3. Level-Prim.

To obtain an assignable collection of subtours, we start with a specific approximate minimum spanning tree of G, the Level-Prim tree (which is used in step 4 of Algorithm 1). A Level-Prim tree is formally defined as one that can be obtained as the output of the following procedure.

Algorithm 2 (Level-Prim(G, M, ϵ))

```
1: Partition vertices into levels $V_0, V_1, \ldots$ using the values of $M$ and $\epsilon$ (§2.1).
2: For each $i \ge 0$, let $H_i$ be edge-set of a minimum spanning tree for $G[V_{\le i}]/V_{<i}$.
3: return $\mathcal{H} = \bigcup_{i\ge 0} H_i$.
```

Note that Level-Prim trees can alternatively be defined by modifying Prim's minimum spanning tree algorithm such that vertices in level i are only considered to be added to the tree after all vertices in levels below i have already been added.

Theorem 2. *For $\epsilon \ge 1$, a* Level-Prim (G, M, ϵ) *tree $\mathcal{H} = \bigcup_{i\ge 0} H_i$ satisfies the following:*
- *The vertex levels along every root-leaf path are nondecreasing.*
- *For each $i \ge 0$, $\sum_{j\ge i} d(H_j) \le (6 + 6/\epsilon) \cdot MST(G/V_{<i})$.*

Note that the second property in Theorem 2 mirrors the second property in Definition 1. Lemma 1 gives a formal connection between these, assuming that M is a good estimate for OPT. We then get the following corollary of Theorem 2.

Corollary 1. *A* Level-Prim (G, M, ϵ) *tree $\mathcal{H} = \bigcup_{i\ge 0} H_i$ satisfies the following:*
1. *The vertex levels along every root-leaf path are nondecreasing.*
2. *If $M \ge OPT$*

$$\sum_{j\ge i} d(H_j) \le \left(6 + \frac{6}{\epsilon}\right) \cdot M \cdot \Lambda((1+\epsilon)^{i-1}) \quad \textit{for all } i \ge 0. \tag{1}$$

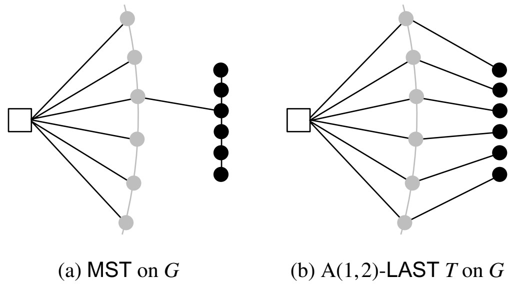


FIGURE 1. Instance with depot (square node), n gray nodes, and n black nodes. The distance between the depot and each gray node, and the distance between pairs of gray nodes, is M. The distance between a gray node and a black node is also M, and the distance between any pair of black nodes is M/n. With $\epsilon = 1$, we have V_0 consisting of the depot and gray nodes, and V_1 containing all black nodes. Notice that $d(T \cap E_1) = n \cdot M$, whereas $MST(G/V_0) < 2M$.

In the rest of this section, we prove Theorem 2. It is easy to see that for every $l \geq 0$, $\bigcup_{j=0}^{l} H_j$ spans $G[V_{\leq l}]$; hence the procedure produces a spanning tree for G. Moreover, by construction we obtain that every root-leaf path in $\mathscr{H}$ traverses the levels in nondecreasing order as desired. Thus, we focus on proving the second property in the theorem.

We compare the length of $\mathscr{H}$ to that of a specific LAST (light approximate shortest path tree) instead of directly comparing it to an MST. The following LAST is implicit in the construction given in Khuller et al. [17]; for completeness, we outline a proof in Appendix A. Recall that a *spider* is a tree with at most one vertex (the center) having degree greater than two.

THEOREM 3 (KHULLER ET AL. [17]). *Given any metric space (V, d) with root r and $\epsilon > 0$, there exists a spanning spider L with center r (seen as a subgraph of the complete graph of the metric space) such that*

- *For each $u \in V$, the shortest-path distance $d_L(r, u)$ from r to u in L is at most $(1+\epsilon) \cdot d(r, u)$.*
- *The length of L is $d(L) \leq 2(1 + 1/\epsilon) \cdot MST$.*

We remark that we cannot use a LAST directly instead of Level-Prim since the former does not need to have the properties asserted by Theorem 2; it is easy to find a LAST that does not satisfy the first property, whereas Figure 1 also shows that the second can also be violated by an arbitrary amount.

Proof of Theorem 2. We prove the second condition for any $i \geq 0$. Let $G' = G/V_{<i}$ denote the graph obtained by contracting vertices $V_{<i}$ to a new depot r'. We let d' denote the resulting shortest path metric on G'. For each $l \geq 0$ and $v \in V_{i+l}$, we have

$$d'(r', v) \leq d(r, v) \leq (1+\epsilon)^{i+l} M \tag{2}$$

$$d'(r', v) \geq d(r, v) - (1+\epsilon)^{i-1} M > (1+\epsilon)^{i-1} M \cdot ((1+\epsilon)^l - 1) \tag{3}$$

Inequality (2) follows immediately from the definition of d' and V_{i+l}. To see inequality (3), note that $d'(r', v) = \min\{d(u, v)\colon u \in V_{<i}\} = d(w, v)$ for some $w \in V_{<i}$; by triangle inequality, we have $d(w, v) \geq d(r, v) - d(r, w)$, which is at least $d(r, v) - (1+\epsilon)^{i-1} M$ by definition of $V_{<i}$.

Notice that levels V_js are defined relative to the original metric; however, (2) and (3) show that even under the new distances d', vertices in a level are at roughly the same distance from the new depot r'.

Let L be a spider LAST obtained by applying Theorem 3 on metric d' with root r'. Let $\mathscr{P}$ denote the set of all root-leaf paths in L, and note that $\mathscr{P}$ is an edge-disjoint collection. Also set $M' := (1+\epsilon)^{i-1} M$ and $\theta := 3$. The next claim shows that each root-leaf path in $\mathscr{P}$ crosses levels *almost* in increasing order.

CLAIM 1. *Consider any root-leaf path $P = (r' = u_1 \to u_2 \to \cdots \to u_k)$ in $\mathscr{P}$. For every pair of nodes $u_a, u_b \in P$ with $a < b$, if $u_a \in V_{i+l+\theta}$ for some $l \geq 0$, then we have $u_b \in V_{>i+l}$.*

PROOF. Suppose (for a contradiction) that there is some value $l \geq 0$ and a pair of nodes $u_a, u_b \in P$ with $a < b$, $u_a \in V_{i+l+\theta}$ and $u_b \in V_{\leq i+l}$. Then

$$d'_L(r', u_b) > d'_L(r', u_a) \geq d'(r', u_a) \geq M' \cdot ((1+\epsilon)^{l+\theta} - 1) \geq M'(1+\epsilon)^{l+2},$$

where the third inequality is because of (3) since $u_a \in V_{i+l+\theta}$, and the last inequality uses

$$(1+\epsilon)^{l+\theta} - 1 = \epsilon((1+\epsilon)^{l+\theta-1} + (1+\epsilon)^{l+\theta-2} \cdots + 1) > (1+\epsilon)^{l+2}$$

for $\epsilon \geq 1$ and $\theta = 3$. On the other hand, $d'(r', u_b) \leq (1+\epsilon)^{l+1} M'$ by (2) since $u_b \in V_{\leq i+l}$, so we obtain $d'_L(r', u_b) > (1+\epsilon) \cdot d'(r', u_b)$, which contradicts the definition of L (see Theorem 3). □

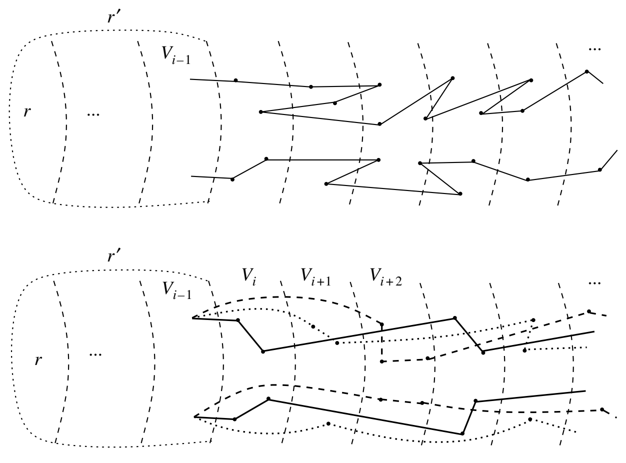


FIGURE 2. Spiders L (top) and L' (bottom). Each root-leaf path in L corresponds to three *monotone* root-leaf paths in L' depicted by solid, dotted, and dashed paths.

Now we transform L into another spider L' that traverses levels in nondecreasing order as follows (see also Figure 2). For each root-leaf path $P = (r = u_1 \to u_2 \to \cdots \to u_k)$, perform the following modification. For each $q \in \{0, 1, \ldots, \theta - 1\}$, let P_q denote the path obtained from P by shortcutting over all vertices *not in* levels indexed q modulo θ; more precisely, if $u_{a_1}, u_{a_2}, \ldots, u_{a_{k'}}$ are the nodes in P at levels q modulo θ (with $a_1 < a_2 < \cdots < a_{k'}$), we set $P_q = (u_{a_1} \to u_{a_2} \to \cdots \to u_{a_{k'}})$. Notice that P_q crosses levels *monotonically*: if not, there must be some $a < b$ in P with $u_a \in V_{\geq i+l+\theta}$ and $u_b \in V_{\leq i+l}$, contrary to Claim 1. Also, by employing the triangle inequality, we have that $d'(P_q) \leq d'(P)$ for all q. Finally define the spider L' as the union of the paths $\{P_0, P_1, \ldots, P_{\theta-1}\}$ over all root-leaf paths $P \in \mathcal{P}$.

By construction, the vertex levels along each root-leaf path of L' are nondecreasing. Additionally $d'(L') \leq \theta \cdot \sum_{P\in\mathcal{P}} d'(P) = \theta \cdot d'(L) \leq \theta(2 + 2/\epsilon) \cdot MST(G/V_{<i})$ by Theorem 3.

Now partition the edges of L' as

$$\Delta_l = \begin{cases} L'[V_{\leq i}] & \text{if } l = 0, \\ L'[V_{\leq i+l}] \backslash L'[V_{<i+l}] & \text{if } l \geq 1. \end{cases}$$

By the monotone property of paths in L', it follows that $L'[V_{\leq i+l}]$ is connected for every $l \geq 0$; recall that $L'[V_{\leq i+l}]$ is a subgraph of $G' = G/V_{<i}$. Thus Δ_l corresponds to a *spanning tree* in the graph $G[V_{\leq i+l}]/V_{<i+l}$. Since H_{i+l} in the Level-Prim construction is chosen to be an MST in $G[V_{\leq i+l}]/V_{<i+l}$, we obtain $d(H_{i+l}) \leq d'(\Delta_l)$.[1] Thus,

$$\sum_{j\geq i} d(H_j) \leq \sum_{l\geq 0} d'(\Delta_l) = d'(L') \leq 6\left(1 + \frac{1}{\epsilon}\right) \cdot MST(G/V_{<i}).$$

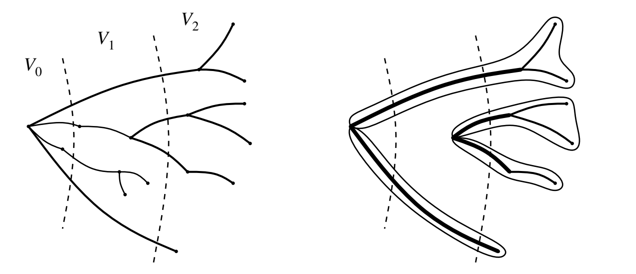
This concludes the proof of Theorem 2. Theorem 3 is proved in the appendix.

2.4. Decomposition procedure. In this section we decompose a Level-Prim tree $\mathcal{H}$ into an assignable collection $\{\mathcal{T}_i\}_{i\geq 0}$ of r-cycles (step 7 of Algorithm 1). Motivated by Corollary 1, the idea is to essentially partition each subgraph H_i into many pieces and connect them to r.

To get an idea of the decomposition algorithm, suppose that each connected component in H_i is large enough, i.e., has length at least $(1+\epsilon)^i M$. Then we can proceed as follows. For each $i \geq 0$, take a Euler tour of each connected component of H_i and partition them into paths of length approximately $(1+\epsilon)^i M$ each; finally, add edges from r to the ends of each path to obtain collection $\mathcal{T}_i$ of r-cycles. The main observation is that each edge added to connect a path to the root has approximately the same length as the path itself. Then by construction of the paths, we get that $\{\mathcal{T}_i\}_{i\geq 0}$ satisfies the first property of an assignable collection; further, using (1), we get that the second property is satisfied as well.

Notice that it was crucial to partition H_i into parts of length $\Omega(1) \cdot (1+\epsilon)^i M$. But this is problematic when H_i has a small connected component. To deal with this, we show that every small component in H_i is attached

[1] Note that each edge in Δ_l is present in $G[V_{\leq i+l}]/V_{<i+l}$, and the shortest-path distance between its end points may only have reduced in comparison to $G' = G/V_{<i}$.



(a) A Level-Prim tree $\mathscr{H}$, with edges H_2 in bold (b) Partition of H_2 into four sub-trees, each with its head-edge high-lighted

Figure 3. Illustration for step 1 of the decomposition procedure.

to ("dangling" from) a large enough component in H_{i-1}; then we simply treat the small H_i component as an integral part of the latter.

Now we formally describe the proposed decomposition procedure.

Step 1 (*create subtrees*). Let $\mathscr{S}_0 = \{H_0\}$. For each level $i \geq 1$, partition the edges of H_i into a collection $\mathscr{S}_i$ of (unrooted) subtrees such that each subtree contains exactly one edge from $V_{<i}$ to V_i. For any $\tau \in \mathscr{S}_i$ call the unique edge from $V_{<i}$ to V_i its *head-edge* $h(\tau)$. Note that such a partition is indeed possible since $H_i/V_{<i}$ is connected. See Figure 3.

Any subtree in $\mathscr{S}_i$ (for $i \geq 0$) is referred to as a level i subtree. Note that head-edges are defined only for subtrees in level 1 and above.

Step 2 (*mark subtrees*). Define $\gamma := \epsilon/((2+\epsilon)(1+\epsilon))$. For each level $i \geq 0$, *mark* those $\tau \in \mathscr{S}_i$ that have $d(\tau) \geq \gamma \cdot (1+\epsilon)^i M$. In addition, *mark* the tree H_0 in $\mathscr{S}_0$. Let $\mathscr{S}_i^m$ and $\mathscr{S}_i^u$ denote the marked and unmarked subtrees in $\mathscr{S}_i$.

Step 3 (*assign unmarked subtrees*). For each level $i \geq 1$ and unmarked $\sigma \in \mathscr{S}_i^u$, define its "parent" $\pi(\sigma)$ as the subtree in $\bigcup_{j<i} \mathscr{S}_j$ containing the other end point of $h(\sigma)$.

Claim 2. *For $i \geq 1$ and unmarked $\sigma \in \mathscr{S}_i^u$, $\pi(\sigma) \in \mathscr{S}_{i-1}$. Moreover, $\pi(\sigma)$ is marked.*

Proof. Since σ is unmarked in level $i \geq 1$, we have $d(h(\sigma)) \leq d(\sigma) < \gamma \cdot (1+\epsilon)^i M$. Let $h(\sigma) = (v, w)$ where $v \in \pi(\sigma)$ and $w \in V_i$. By definition $v \in V_{<i}$, but in fact we have $v \in V_{i-1}$: otherwise, $d(h(\sigma)) = d(v, w) \geq d(r, w) - d(r, v) > (1+\epsilon)^{i-1} M - (1+\epsilon)^{i-2} M > \gamma \cdot (1+\epsilon)^i M$, meaning that σ is marked, contrary to the assumption. Thus $\pi(\sigma) \in \mathscr{S}_{i-1}$.

For the second part of the claim, notice that if $i = 1$, then $\pi(\sigma) = H_0$, which is always marked. Suppose $i \geq 2$. From the above, $\pi(\sigma)$ is in level $i-1 \geq 1$ and hence contains a head-edge. This implies that $\pi(\sigma)$ contains some vertex $u \in V_{<i-1}$, namely an end point of $h(\pi(\sigma))$. Employing triangle inequality twice, we have

$$d(u, v) + d(v, w) \geq d(u, w) \geq d(r, w) - d(r, u).$$

Since $w \in V_i$ and $u \in V_{\leq i-2}$, we have $d(r, w) - d(r, u) > (1+\epsilon)^{i-1} M - (1+\epsilon)^{i-2} M = \epsilon(1+\epsilon)^{i-2} M$. That is, $d(u, v) + d(v, w) > \epsilon(1+\epsilon)^{i-2} M$. Since σ is unmarked, $d(v, w) < \gamma(1+\epsilon)^i M$. Hence,

$$d(\pi(\sigma)) \geq d(u, v) > \epsilon(1+\epsilon)^{i-2} M - \frac{\epsilon}{2+\epsilon}(1+\epsilon)^{i-1} M = \gamma \cdot (1+\epsilon)^{i-1} M.$$

This implies that subtree $\pi(\sigma)$ is marked (in level $i-1$). □

Step 4 (*partition marked subtrees*). For each level $i \geq 0$ and marked $\tau \in \mathscr{S}_i^m$, define $\mathsf{Dangle}(\tau) = \pi^{-1}(\tau)$ as the set of all unmarked $\sigma \in \mathscr{S}_{i+1}^u$ having $\pi(\sigma) = \tau$; see also Figure 4. We will partition the tree $\tau \cup \mathsf{Dangle}(\tau)$ into subtours. Take a Euler tour of $\tau \cup \mathsf{Dangle}(\tau)$ by "doubling" its edges and split the resulting tour into maximal paths of length at most $2(1+\epsilon)^{i+1} M$ each; let $P_1, \ldots, P_q$ denote the resulting paths. Finally, add edges from r to the end points of each of these paths to obtain a collection $\mathscr{T}_i(\tau)$ of r-cycles.

Claim 3. *For any $T \in \mathscr{T}_i(\tau)$, we have $d(T) \leq (4+4\epsilon) \cdot (1+\epsilon)^i M$.*

Proof. Notice that every $T \in \mathscr{T}_i(\tau)$ consists of a path P_j (for some $1 \leq j \leq q$) and edges from r to both end points of P_j. By construction, $d(P_j) \leq 2(1+\epsilon)^{i+1} M$. Moreover, since P_j only contains vertices from $V_{\leq i+1}$, the two additional edges increase the length by at most $2(1+\epsilon)^{i+1} M$. □

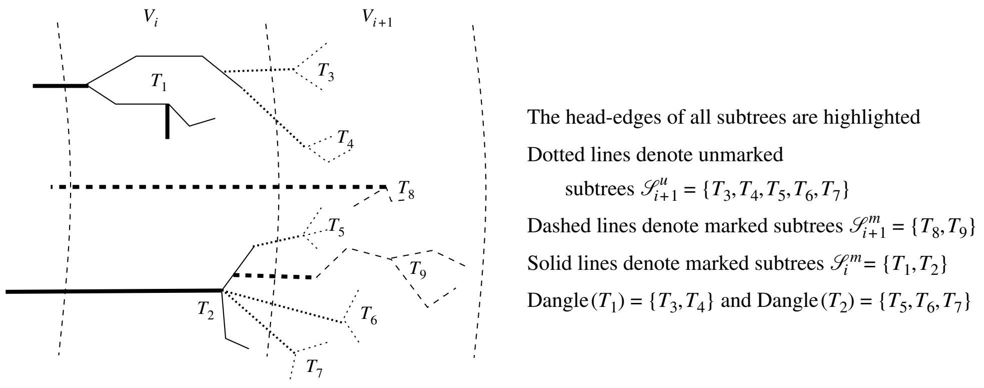

V_i
V_{i+1}
T_1
T_2
T_3
T_4
T_5
T_6
T_7
T_8
T_9
The head-edges of all subtrees are highlighted
Dotted lines denote unmarked subtrees $\mathcal{S}^u_{i+1} = \{T_3, T_4, T_5, T_6, T_7\}$
Dashed lines denote marked subtrees $\mathcal{S}^m_{i+1} = \{T_8, T_9\}$
Solid lines denote marked subtrees $\mathcal{S}^m_i = \{T_1, T_2\}$
Dangle$(T_1) = \{T_3, T_4\}$ and Dangle$(T_2) = \{T_5, T_6, T_7\}$

FIGURE 4. Marked and unmarked subtrees, and dangles.

CLAIM 4. $\sum_{T\in\mathcal{T}_i(\tau)} d(T) \le \max\{4+4/\epsilon, 6\}\cdot[d(\tau)+d(\mathsf{Dangle}(\tau))]$.

PROOF. We break the analysis into two cases depending on q. Suppose first that $q=1$; namely, $\mathcal{T}_i(\tau)$ consists of a single subtour T. If $i=0$, then τ contains r and $d(T)\le 2\cdot d(\tau)+2\cdot d(\mathsf{Dangle}(\tau))$. Assume $i>1$, in which case τ has a vertex $u\in V_{<i}$, namely, an end point of $h(\tau)$. By adding edges from r to this vertex u, we can ensure $d(T)\le 2(1+\epsilon)^{i-1}M+2\cdot d(\tau)+2\cdot d(\mathsf{Dangle}(\tau))$. Also, because τ is marked, we have $d(\tau)\ge\gamma\cdot(1+\epsilon)^i M$ implying that $(1+\epsilon)^{i-1}M\le((2+\epsilon)/\epsilon)\cdot d(\tau)$. Thus $d(T)\le(4+4/\epsilon)\cdot d(\tau\cup\mathsf{Dangle}(\tau))$.

Now suppose $q\ge 2$. This means $d(\tau\cup\mathsf{Dangle}(\tau))>(q-1)(1+\epsilon)^{i+1}M$. Recall that each edge added from the root has length at most $(1+\epsilon)^{i+1}M$ since $\tau\cup\mathsf{Dangle}(\tau)$ lies in $V_{\le i+1}$. Therefore,

$$\sum_{T\in\mathcal{T}_i(\tau)} d(T)\le 2q\cdot(1+\epsilon)^{i+1}M+2\cdot d(\tau\cup\mathsf{Dangle}(\tau))\le\left(\frac{2q}{q-1}+2\right)\cdot d(\tau\cup\mathsf{Dangle}(\tau))$$

Since $q\ge 2$, the last term is at most $6\cdot d(\tau\cup\mathsf{Dangle}(\tau))$. □

For each level $i\ge 0$, define collection $\mathcal{T}_i=\bigcup_{\tau\in\mathcal{S}^m_i}\mathcal{T}_i(\tau)$ of r-cycles. The following lemma summarizes the key property of our decomposition procedure.

LEMMA 3. *Suppose that inequality* (1) *holds*; *then the collection* $\{\mathcal{T}_i\}_{i\ge 0}$ *obtained from the above procedure is* (α,β)*-assignable, with* $\alpha=4+4\epsilon$ *and* $\beta=(6+6/\epsilon)\cdot\max\{6,4+4/\epsilon\}$.

PROOF. We will show that the collection $\{\mathcal{T}_i\}_{i\ge 0}$ satisfies the two properties in Definition 1. By Claim 3, each subtour in $\mathcal{T}_i$ has length at most $\alpha\cdot(1+\epsilon)^i M$, proving the first property in Definition 1.

To see the second property in Definition 1, fix any $i\ge 0$. Due to the assumption that (1) holds, it suffices to prove that $\sum_{j\ge i} d(\mathcal{T}_j)\le\max\{6,4+4/\epsilon\}\cdot\sum_{j\ge i} d(H_j)$. Using Claim 4 we obtain that

$$d(\mathcal{T}_j)=\sum_{\tau\in\mathcal{S}^m_j} d(\mathcal{T}_i(\tau))\le\beta'\cdot\sum_{\tau\in\mathcal{S}^m_j}[d(\tau)+d(\mathsf{Dangle}(\tau))]=\beta'\cdot(d(\mathcal{S}^m_j)+d(\mathcal{S}^u_{j+1})),$$

where $\beta'=\max\{6,4+4/\epsilon\}$. The last equality above uses the fact that $\{\mathsf{Dangle}(\tau)\colon \tau\in\mathcal{S}^m_j\}$ is a partition of $\mathcal{S}^u_{j+1}$. The desired inequality follows by observing that

$$\sum_{j\ge i} d(\mathcal{S}^m_j)+\sum_{j\ge i} d(\mathcal{S}^u_{j+1})\le\sum_{j\ge i} d(\mathcal{S}_j)=\sum_{j\ge i} d(H_j).$$

This concludes the proof of Lemma 3. □

2.5. Final analysis of algorithm. First, notice that every step of Algorithm 1 can indeed be implemented in polynomial time. By Corollary 1 we are guaranteed that the **if** condition (1) holds whenever $M\ge OPT$. By the update rule for M, it follows that the step 7 is executed with $M\le(1+\delta)OPT$. In this case, Lemma 3 implies (with $\epsilon=2$) that we obtain a $(12,54)$-assignable collection $\{\mathcal{T}_i\}_{i\ge 0}$. Given this, Lemma 2 guarantees that step 8 finds a solution to HetTSP with makespan at most $90\cdot M\le 90(1+\delta)\cdot OPT$.

Finally, we bound the number of iterations. As noted above, the algorithm halts before M becomes larger than $(1+\delta)OPT$, which is at most $4(\sum_v d(r,v)/\max_i\lambda_i)\le 4|V|\max_v(d(r,v)/\max_i\lambda_i)$ (attained by the solution that uses the fastest vehicle to visits all the vertices). Since M is initialized at $\max_v(d(r,v)/\max_i\lambda_i)$ and increased by a $1+\delta$ factor in every iteration, it follows that the loop is executed at most $O(1/\delta\log|V|)$ times. Therefore, the entire algorithm runs in polynomial time. This proves the first part of Theorem 1.

3. Algorithm for HetCVRP. Recall the *Heterogeneous CVRP* (HetCVRP) consisting of a metric (V, d) denoting distances between vertices, depot $r \in V$ (containing an infinite supply of items), demands $\{q_v\}_{v\in V}$, and k vehicles with speeds $\{\lambda_i\}_{i=1}^k$, each having capacity Q. A solution to HetCVRP consists of tours $\{\sigma_i\}_{i=1}^k$ (each starting and ending at r) for the k vehicles so that all demands are satisfied and each vehicle carries at most Q items at any point in time. The objective is to minimize the *maximum completion time*, $\max_{i=1}^k (d(\sigma_i)/\lambda_i)$.

We study the unsplit-delivery version of HetCVRP here, where the entire demand at a vertex must be served in a single visit to it, so the demands satisfy $\max_{v\in V} q_v \le Q$. In the alternative (less restrictive) split-delivery HetCVRP, the demand at a vertex may be served by multiple visits to it. It is clear that for any instance of HetCVRP, the optimal value under split-deliveries is at most that under unsplit-deliveries. We give an algorithm for unsplit-delivery HetCVRP that achieves a constant factor approximation relative to the optimal split-delivery routing.

Before proceeding to the HetCVRP algorithm, we recall some known lower bounds for (single vehicle) CVRP. Consider an instance of CVRP with depot r, vehicle capacity Q, and demands $\{q_v\}_{v\in V}$. Let $S = \{v \in V: q_v > 0\}$ be the set of vertices with strictly positive demands. Then the *optimal split-delivery* CVRP value is at least

$$\max\left\{\frac{2}{Q}\sum_{v\in S} q_v \cdot d(r, v),\ TSP(\{r\} \cup S)\right\}. \tag{4}$$

Above, $TSP(\{r\} \cup S)$ denotes the minimum length of a traveling salesman tour on vertices $\{r\} \cup S$. On the other hand, given any TSP tour τ on $\{r\} \cup S$, it is known (Altinkemer and Gavish [1]) that one can obtain in polynomial time an *unsplit-delivery* CVRP solution of length at most

$$d(\tau) + \frac{4}{Q}\sum_{v\in S} q_v \cdot d(r, v). \tag{5}$$

The main idea in the following algorithm is to reduce each HetCVRP instance to a suitable instance of HetTSP. This involves modifying the input metric based on the above lower-bounds for (single vehicle) CVRP. To avoid ambiguity, we use OPT^s_{vrp} to denote the optimum for split-delivery *HetCVRP* and OPT_{tsp} to denote the optimum for *HetTSP*.

Theorem 4. *Given any instance $\mathcal{I}$ of* HetCVRP, *there is a polynomial-time constructible instance $\mathcal{J}$ of* HetTSP *such that*

- $OPT_{tsp}(\mathcal{J}) \le 5 \cdot OPT^s_{vrp}(\mathcal{I})$.
- *Any solution to $\mathcal{J}$ of makespan M can be converted (in polynomial time) to an unsplit-delivery solution to $\mathcal{I}$ having makespan M.*

Proof. Let $\mathcal{I}$ be an instance of HetCVRP on metric (V, d), depot r, vehicle speeds $\{\lambda_i\}_{i=1}^k$, capacity Q, and demands $\{q_v\}_{v\in V}$. We construct a HetTSP instance $\mathcal{J}$ on the same vertex set V, with root r and k vehicles of speeds $\{\lambda_i\}_{i=1}^k$. The metric (V, l) in $\mathcal{J}$ is defined as follows. Set $\Delta_v = 2q_v \cdot d(r, v)/Q$ for each $v \in V$. The distances l are

$$l(u, v) := d(u, v) + \Delta_u + \Delta_v, \quad \forall\, u, v \in V.$$

Notice that (V, l) is indeed a metric. We now prove the two claimed properties.

Below, we use the notation $[k] := \{1, 2, \ldots, k\}$.

Split-delivery HetCVRP *to* HetTSP. Consider any *split-delivery* solution $\{\sigma_i\}_{i=1}^k$ to $\mathcal{I}$. For each $i \in [k]$ and $v \in V$, let $c_i(v) \in [0, q_v]$ denote the amount of vertex v's demand that is served by vehicle i. (Note that we can have $0 < c_i(v) < q_v$ since multiple vehicles may serve the demand at vertex v.) Let $S_i = \{v \in V: c_i(v) > 0\}$ for each $i \in [k]$. Using the lower bound (4) we have

$$d(\sigma_i) \ge \max\left\{\frac{2}{Q}\sum_{v\in S_i} c_i(v) \cdot d(r, v),\ TSP_d(\{r\} \cup S_i)\right\}, \quad \forall\, i \in [k]. \tag{6}$$

The subscript d in $TSP_d(\cdot)$ denotes that it is the minimum length TSP tour in metric (V, d).

We will construct a solution $\{\tau_i\}_{i=1}^k$ to $\mathcal{J}$ such that $l(\tau_i) \le 5 \cdot d(\sigma_i)$ for each $i \in [k]$. This would prove the first property. To obtain $\{\tau_i\}_{i=1}^k$, we consider the following instance of *scheduling on unrelated machines*. The k vehicles are "machines," and each vertex $v \in V$ is a "job." The processing times are

$$p_{i,v} := \begin{cases} 2 \cdot \Delta_v & \text{if } v \in S_i, \\ \infty & \text{otherwise,} \end{cases} \quad \forall\, i \in [k],\ v \in V.$$

Consider the fractional assignment $x_{i,v} := c_i(v)/q_v$ for $i \in [k]$, $v \in V$. Note that for each $v \in V$, $\sum_{i=1}^{k} x_{i,v} = 1$. Since $x_{i,v} = 0$ for $v \notin S_i$, assignment x induces a load on machine $i \in [k]$ of

$$2\sum_{v\in S_i} x_{i,v}\cdot \Delta_v = \frac{4}{Q}\sum_{v\in S_i} c_i(v)\cdot d(r,v) \le 2\cdot d(\sigma_i),$$

where the inequality uses (6). Furthermore, for any $i \in [k]$ and $v \in V$ with $x_{i,v} > 0$, we have

$$p_{i,v} = 2\Delta_v = 4d(r,v)\cdot\frac{q_v}{Q} \le 4\cdot d(r,v) \le 2\cdot TSP_d(\{r\}\cup S_i) \le 2\cdot d(\sigma_i)$$

The first inequality uses $q_v \le Q$, the second uses $v \in S_i$, and the last is by (6). Thus the rounding algorithm from Lenstra et al. [18] implies an *integral assignment* π: $V \to [k]$ such that for each $i \in [k]$,

$$\sum_{v\in\pi^{-1}(i)} p_{i,v} \le \sum_v p_{i,v}\cdot x_{i,v} + \max\{p_{i,v}\colon x_{i,v} > 0\} \le 2\cdot d(\sigma_i) + 2\cdot d(\sigma_i) = 4\cdot d(\sigma_i).$$

In particular,

$$\pi^{-1}(i)\subseteq S_i \quad\text{and}\quad 2\sum_{v\in\pi^{-1}(i)}\Delta_v \le 4\cdot d(\sigma_i), \qquad \forall\, i\in[k].$$

We now define for each $i \in [k]$, tour τ_i in metric l to be the optimal TSP tour on $r \cup \pi^{-1}(i)$. Thus,

$$l(\tau_i) = TSP_l(r\cup\pi^{-1}(i)) = TSP_d(r\cup\pi^{-1}(i)) + \sum_{v\in\pi^{-1}(i)} 2\cdot\Delta_v \le 5\cdot d(\sigma_i), \quad \forall\, i\in[k].$$

The second equality uses the property of metric l that for any $S\subseteq V$, we have $TSP_l(S) = TSP_d(S) + \sum_{v\in S} 2\cdot\Delta_v$. The inequality uses the above properties of $\pi^{-1}(i)$ and that $TSP_d(\{r\}\cup S_i) \le d(\sigma_i)$ by (6).

HetTSP *to unsplit-delivery* HetCVRP. Consider any solution $\{\tau_i'\}_{i=1}^{k}$ to $\mathcal{J}$. We will construct in polynomial time an *unsplit-delivery* solution $\{\sigma_i'\}_{i=1}^{k}$ to $\mathcal{J}$ such that $d(\sigma_i') \le l(\tau_i')$. This would prove the second property. Fix any $i \in [k]$. Let $V_i\subseteq V$ denote the vertices visited in τ_i'. By definition of the metric l, we have $l(\tau_i') = d(\tau_i') + \sum_{v\in V_i} 2\cdot\Delta_v = d(\tau_i') + (4/Q)\sum_{v\in V_i} q_v\cdot d(r,v)$. Using the Algorithm (Altinkemer and Gavish [1]) for unsplit-delivery CVRP (single vehicle) on metric (V, d) with depot r, demands $\{q_v\colon v \in V_i\}$, and the given TSP tour τ_i', we obtain a solution σ_i'. By (5),

$$d(\sigma_i') \le d(\tau_i') + \frac{4}{Q}\sum_{v\in V_i} q_v\cdot d(r,v) = l(\tau_i').$$

This completes the proof. □

Combined with the constant factor approximation algorithm for HetTSP (previous section), this implies the second part of Theorem 1. Moreover, our performance guarantee for HetCVRP is relative to the less restrictive split-delivery version.

4. Concluding remarks. In this paper we gave constant-factor approximation algorithms for the TSP and CVRP problems in the presence of multiple nonuniform speed vehicles. An interesting open question regards the approximability of *HetTSP* and *HetCVRP* when vehicles originate at multiple different depots across the space. The current definition of an assignable collection and the definition of Level-Prim crucially depend on the presence of a unique depot; hence, an extension to the multidepot case is likely to require new ideas. Another interesting question is HetCVRP with nonuniform capacities, where the ideas presented in §3 do not seem to generalize directly.

Acknowledgments. Supported in part by The Danish Council for Independent Research—Natural Sciences grant DFF–1323-00178. Supported in part by NSF grant CCF-1218382.

Appendix A. Proof of Theorem 3. Given metric (V, d) with root r and any $\epsilon > 0$, we will show that the following algorithm produces an (α, β)-spider, where $\alpha := 1+\epsilon$ and $\beta := 2+2/\epsilon$. In particular, we will show that the solution S^* is a spider where (a) its length $d(S^*) \le \beta \cdot MST$ and (b) the distance in S^* from any vertex v to the root r is at most $\alpha \cdot d(r, v)$.

Algorithm 3 ((α, β)-Spider)

```
1: Consider an MST for $G$ and traverse it along a Euler tour to obtain path $S^0 = S = (r = u_1 \to u_2 \to \cdots \to u_n)$
   such that $d(S^0) \le 2 \cdot MST$.
2: **for** $i$ from 1 to $n$ **do**
3: **if** $d_S(u_1, u_i) > \alpha d(u_1, u_i)$, **then**
4: **add** $(u_1, u_i)$ to $S$ and **mark** $u_i$.
5: **for each** marked node $u_i$, remove $(u_{i-1}, u_i)$ from $S$.
6: **return** $S^* = S$.
```

Lemma 4. *The graph S^* returned by the algorithm is a spider.*

Proof. The initial graph S^0 is a path, and steps 2–4 add some edges. It is clear that after these additions (just before step 5) only the root u_1 and marked nodes have degree larger than two. Moreover, each marked node has degree exactly three. Thus, after step 5, only the root has degree larger than two in S^*, and the lemma follows. □

Lemma 5. *Spider S^* is an (α, β)-LAST.*

Proof. Define $\sigma(0) = 1$ and let $\sigma(1) < \sigma(2) < \cdots < \sigma(k)$ denote the indices of the *marked* nodes in the algorithm. We will also call the root u_1 a marked node.

First we prove that the distance in S^* from root u_1 to u_i is $d_{S^*}(u_1, u_i) \le \alpha d(u_1, u_i)$, for all $i \in \{1, \ldots, n\}$. This is immediate for all marked nodes since S^* contains the edges $\{(u_1, u_{\sigma(j)})\}_{j=0}^{k}$. To see this for any unmarked node u_i, consider graph S in the ith iteration of the for-loop (steps 2–4). If l denotes the highest index of a marked node at this point, then the shortest path in S from u_1 to u_i is $P_i = (u_1 \to u_l \to u_{l+1} \to \cdots u_i)$. Since u_i was not marked, $d(P_i) \le \alpha \cdot d(u_1, u_i)$. Finally, observe that path P_i is also the u_1–u_i path in the final solution S^*, which proves $d_{S^*}(u_1, u_i) \le \alpha d(u_1, u_i)$.

Now we prove that the length of S^* is $d(S^*) \le \beta \cdot MST$; recall $\beta = 2+2/\epsilon$. It is clear that

$$d(S^*) \le d(S^0) + \sum_{j=1}^{k} d(u_1, u_{\sigma(j)}) \le 2 \cdot MST + \sum_{j=1}^{k} d(u_1, u_{\sigma(j)}).$$

It now suffices to upper bound the last summation by $(2/\epsilon) \cdot MST$.

Consider any marked node $u_{\sigma(j)}$, and the iteration $\sigma(j)$ where it is marked. Notice that graph S at this point contains the path $(u_1 \to u_{\sigma(j-1)} \to u_{\sigma(j-1)+1} \to \cdots u_{\sigma(j)})$ and so

$$d_S(u_1, u_{\sigma(j)}) \le d(u_1, u_{\sigma(j-1)}) + d_S(u_{\sigma(j-1)}, u_{\sigma(j)}) \le d(u_1, u_{\sigma(j-1)}) + d_{S^0}(u_{\sigma(j-1)}, u_{\sigma(j)}).$$

The last inequality is because S^0 is a subgraph of S. However, since $u_{\sigma(j)}$ was marked, we have $d_S(u_1, u_{\sigma(j)}) > \alpha \cdot d(u_1, u_{\sigma(j)})$, and so

$$\alpha \cdot d(u_1, u_{\sigma(j)}) < d(u_1, u_{\sigma(j-1)}) + d_{S^0}(u_{\sigma(j-1)}, u_{\sigma(j)}).$$

Adding the previous inequality over all j's we get

$$\alpha \sum_{j=1}^{k} d(u_1, u_{\sigma(j)}) < \sum_{j=1}^{k} d(u_1, u_{\sigma(j-1)}) + \sum_{j=1}^{k} d_{S^0}(u_{\sigma(j-1)}, u_{\sigma(j)}).$$

Reorganizing this inequality, we have

$$(\alpha - 1) \sum_{j=1}^{k} d(u_1, u_{\sigma(j)}) \le \sum_{j=1}^{k} d_{S^0}(u_{\sigma(j-1)}, u_{\sigma(j)}) \le d(S^0).$$

The last inequality follows from the fact that the marked nodes $u_{\sigma(1)}, u_{\sigma(2)}, \ldots, u_{\sigma(k)}$ appear in that order on path S^0. Since $\alpha = 1+\epsilon$, this implies $\sum_{j=1}^{k} d(u_1, u_{\sigma(j)}) \le (2/\epsilon) \cdot MST$, as desired. □

References

[1] Altinkemer K, Gavish B (1987) Heuristics for unequal weight delivery problems with a fixed error guarantee. *Oper. Res. Lett.* 6(4):149–158.

[2] Altinkemer K, Gavish B (1990) Heuristics for delivery problems with constant error guarantees. *Transportation Res.* 24:294–297.

[3] Arkin EM, Hassin R, Levin A (2006) Approximations for minimum and min-max vehicle routing problems. *J. Algorithms* 59(1):1–18.

[4] Arora S (1998) Polynomial time approximation schemes for Euclidean traveling salesman and other geometric problems. *J. ACM* 45(5):753–782.

[5] Awerbuch B, Baratz A, Peleg D (1990) Cost-sensitive analysis of communication protocols. *Proc. Sympos. Principles of Distributed Comput.* (ACM, New York), 177–187.

[6] Blum A, Chawla S, Karger DR, Lane T, Meyerson A, Minkoff M (2007) Approximation algorithms for orienteering and discounted-reward TSP. *SIAM J. Comput.* 37(2):653–670.
[7] Bompadre A, Dror M, Orlin JB (2006) Improved bounds for vehicle routing solutions. *Discrete Optim.* 3(4):299–316.
[8] Chekuri C, Kumar A (2004) Maximum coverage problem with group budget constraints and applications. *Proc. Workshop on Approximation Algorithms for Combin. Optim.* (Springer, Berlin), 72–83.
[9] Chekuri C, Korula N, Pál M (2012) Improved algorithms for orienteering and related problems. *ACM Trans. Algorithms* 8(3):23.
[10] Christofides N (1976) Worst-case analysis of a new heuristic for the travelling salesman problem. Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh.
[11] Cook WJ, Cunningham WH, Pulleyblank WR, Schrijver A (1998) *Combinatorial Optimization* (John Wiley & Sons, New York).
[12] Das A, Mathieu C (2010) A quasi-polynomial time approximation scheme for Euclidean capacitated vehicle routing. *Proc. Sympos. Discrete Algorithms* (SIAM, Philadelphia), 390–403.
[13] Even G, Garg N, Könemann J, Ravi R, Sinha A (2004) Min-max tree covers of graphs. *Oper. Res. Lett.* 32(4):309–315.
[14] Frederickson GN, Hecht MS, Kim CE (1978) Approximation algorithms for some routing problems. *SIAM J. Comput.* 7(2):178–193.
[15] Gupta A, Nagarajan V, Ravi R (2012) Approximation algorithms for VRP with stochastic demands. *Oper. Res.* 60(1):123–127.
[16] Haimovich M, Kan AHGR (1985) Bounds and heuristics for capacitated routing problems. *Math. Oper. Res.* 10(4):527–542.
[17] Khuller S, Raghavachari B, Young NE (1995) Balancing minimum spanning trees and shortest-path trees. *Algorithmica* 14(4):305–321.
[18] Lenstra JK, Shmoys DB, Tardos É (1990) Approximation algorithms for scheduling unrelated parallel machines. *Math. Programming* 46(1–3):259–271.
[19] Mitchell JSB (1999) Guillotine subdivisions approximate polygonal subdivisions: A simple polynomial-time approximation scheme for geometric TSP, k-MST, and related problems. *SIAM J. Comput.* 28(4):1298–1309.
[20] Schrijver A (2003) *Combinatorial Optimization* (Springer, Berlin).
[21] Toth P, Vigo D, eds. (2002) *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications (SIAM, Philadelphia).