Minimum Vehicle Routing with a Common Deadline

Viswanath Nagarajan[⋆] and R. Ravi[⋆⋆]

Tepper School of Business, Carnegie Mellon University, Pittsburgh PA 15213
{viswa, ravi}@cmu.edu

Abstract. In this paper, we study the following vehicle routing problem: given n vertices in a metric space, a specified root vertex r (the depot), and a length bound D, find a minimum cardinality set of r-paths that covers all vertices, such that each path has length at most D. This problem is $\mathcal{NP}$-complete, even when the underlying metric is induced by a weighted star. We present a 4-approximation for this problem on tree metrics. On general metrics, we obtain an $O(\log D)$ approximation algorithm, and also an $(O(\log \frac{1}{\epsilon}), 1+\epsilon)$ bicriteria approximation. All these algorithms have running times that are almost linear in the input size. On instances that have an optimal solution with one r-path, we show how to obtain in polynomial time, a solution using at most 14 r-paths.

We also consider a linear relaxation for this problem that can be solved approximately using techniques of Carr & Vempala [7]. We obtain upper bounds on the integrality gap of this relaxation both in tree metrics and in general.

1 Introduction

A common version of vehicle routing problems involves locations that demand service, and a single depot that has to send vehicles to satisfy these demands. It may be important to service *all* demands before a deadline, so several vehicles may need to be deployed. In this context, meeting demands are hard constraints which must be satisfied, while the objective is to minimize the number of vehicles used.

Vehicle routing problems are extensively studied in the Operations Research literature [10,11,13,15,16]. Most of these papers focus on developing heuristic solutions or solving the problems optimally. The methods used in these papers include branch and bound, cutting plane algorithms, local search, and genetic algorithms. There has been considerably less work on these problems in the approximation algorithms literature, perhaps due to the inapproximability of natural formulations of these problem. The version that we study is more tractable from the point of view of obtaining approximation guarantees.

Approximation guarantees for the problem we consider have been studied in Li et al. [9], and Bazgan et al. [5]. Li et al. [9] suggested a tour-splitting heuristic

[⋆] Supported by NSF ITR grant CCR-0122581. (The ALADDIN project)

[⋆⋆] Supported in part by NSF grants CCF-0430751 and ITR grant CCR-0122581. (The ALADDIN project) which has a performance guarantee which depends on some *values* in the input instance, but does not yield any worst case approximation bounds. Bazgan et al. [5] suggested algorithms achieving provable worst case bounds under a *differential approximation* measure. However, bounds in this measure do not imply any bounds in the standard approximation measure. In this paper, we study standard approximation algorithms for the common deadline vehicle routing problem.

There has been some interesting recent work [3,6] on approximating the related *orienteering* problem. In this problem, there is a single vehicle, and the goal is to find a bounded length path from the depot, that maximizes the number of vertices covered. Improving on work by Blum et al. [6], Bansal et al. [3] presented a 3-approximation algorithm for orienteering on general metrics. Bansal et al. [3] also considered extensions where vertices can only be covered in individual time windows, giving poly-logarithmic approximations for this case. Improved algorithms for special classes of metrics were obtained in [2,4].

Problem definition: We model locations as points in a metric space (V, d), with $|V| = n$. Here d is a function $d : V \times V \to \mathbb{Z}^+$ that is symmetric and satisfies the triangle inequality. We assume throughout that all distances are integral. The input to the *minimum vehicle routing problem* consists of a metric space (V, d), one designated *root* vertex $r \in V$, and a length bound D. The root corresponds to the depot, and the length bound D represents the common deadline of the demand locations. The objective is to find a minimum cardinality set of paths originating from r, that covers all vertices in V. In addition, all these paths are required to have length at most D. Paths originating from r are called r-paths. We also refer to the minimum vehicle routing problem as rooted vehicle routing.

A related problem is the *unrooted vehicle routing* problem. In this problem, there is no designated root, and vehicles can start at *any* vertex. The goal here is to find a minimum cardinality set of paths that cover all the vertices. The paths are again required to have length at most the length bound D. This problem has been studied recently in Arkin et al. [1] as minimum path cover, and they gave a 3-approximation algorithm for it.

Our results: For minimum vehicle routing on a tree metric, we obtain a 4-approximation algorithm in Section 2. We note that the problem is $\mathcal{NP}$-complete even in this special case. For minimum vehicle routing on general metrics, we obtain an $O(\log D)$-approximation algorithm, and a bi-criteria result in Section 3. We also consider an integer programming formulation of this problem. We show that the integrality gap of its linear relaxation is upper bounded by a constant in the case of tree metrics (Section 2.1), and by $O(\min\{\log n, \log D\})$ in case of general metrics (Section 3.1). Determining a tight bound on the integrality gap of this relaxation is still open. We consider the following promise problem in Section 4: *given* an instance of minimum vehicle routing, where the optimal solution uses just a single vehicle, *find* a solution using a small number of vehicles. We show how the minimum excess path problem [6] can be used to obtain (in polynomial time) a solution to this promise problem having at most 14 vehicles.

Due to lack of space, we omit some proofs in this version of the paper. The interested reader may refer to [14] for the proofs missing here.

2 Minimum Vehicle Routing on a Tree

In this section, we consider the special case of minimum vehicle routing, when the metric space $T = (V, d)$ is induced by a tree. Even in the special case of a star, the problem remains $\mathcal{NP}$-complete (reduction from 3-partition). Here we present a 4-approximation for minimum vehicle routing on trees.

We assume without loss of generality, that the tree is binary, and rooted at r. This can be ensured by splitting high degree vertices, and adding edges of zero length. Suppose the input consists of tree $T = (V, d)$, root $r \in V$, and length bound D. Algorithm minTVR for minimum vehicle routing on trees is as follows.

```
1. Initialize $T' = T$.
2. While ($T' \neq \{r\}$) do
   (a) Pick a deepest vertex $v \in T'$ s.t. the subtree $T'_v$ below $v$ *can not* be
       covered by just one $r$-path, of length at most $D$. If no such $v$ exists, add
       an $r$-path covering $T'$, and exit loop.
   (b) Let $w_1$ and $w_2$ be the two children of $v$. For $i = 1, 2$, set $W_i$ to be the
       minimum length $r$-path traversing subtree $T'_{w_i}$.
   (c) Add $r$-paths $W_1$ and $W_2$.
   (d) $T' = T' \setminus T'_v$.
```

Note that it is easy to find the minimum length r-path covering all the vertices of a tree - the longest r to leaf path is traversed once, and all other edges are traversed 2 times. See Figure 1b for the structure of an r-path on a tree. Thus the condition in step 2a can be checked efficiently.

Theorem 1. *Algorithm minTVR obtains a 4-approximation to the minimum vehicle routing problem on trees.*

Proof: It is not hard to see that algorithm minTVR can be implemented in a single depth-first search of the tree; so the time complexity is linear in the input size. Suppose we are given an instance of minimum vehicle routing on a tree $T = (V, d)$, with root $r \in V$.

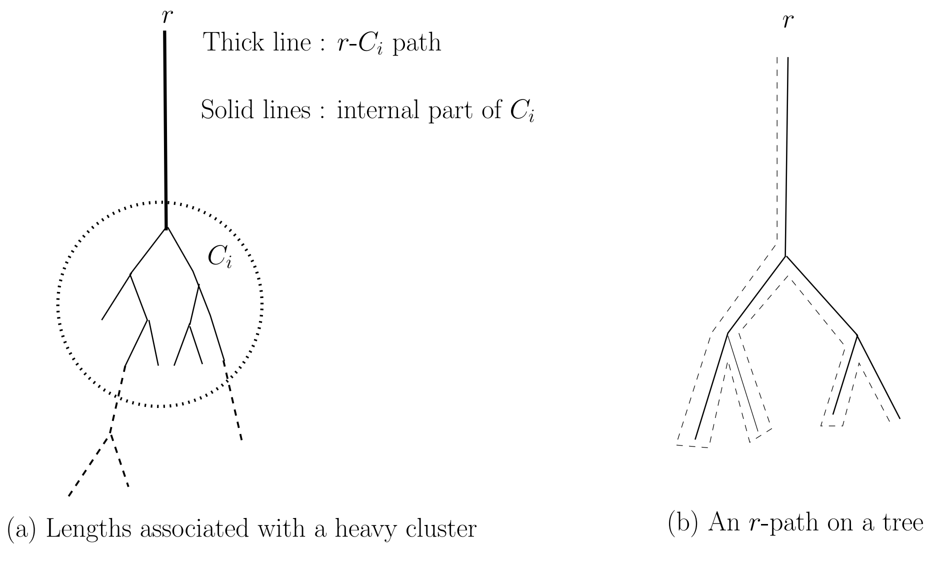
A *heavy cluster* is a set of vertices $C \subseteq V$ such that the induced subgraph $T[C]$ is connected, and the vertices in C can not all be covered by a single r-path of length at most D. Note that the subtrees T'_v seen in step 2a of the algorithm are heavy clusters. Suppose, in its entire execution, the algorithm finds k heavy clusters $C_1, \cdots C_k$ (these vertex sets will be disjoint). Then algorithm minTVR uses at most $2k+1$ r-paths to cover all the vertices. From the definition of vertex v (in step 2a), each r-path W_i added in step 2c (corresponding to the children of v), has length at most D. So the algorithm indeed produces a feasible solution. The following lemma shows that the optimal solution requires at least $\frac{k+1}{2}$ vehicles, and thus proves Theorem 1.

Lemma 1. *If there are k disjoint heavy clusters $C_1, \cdots C_k \subseteq V$ in the tree T, the minimum number of r-paths of length at most D required to cover $\bigcup_{i=1}^{k} C_i$ is more than $\lfloor \frac{k+1}{2} \rfloor$.*

Proof: The proof of this lemma is by induction on k. For $k = 1$, the lemma is trivially true. Suppose $k > 1$, and assume that the lemma holds for all values up to $k - 1$. Suppose the minimum number of r-paths required to cover all these clusters, $OPT \leq \lfloor \frac{k+1}{2} \rfloor$. Note that OPT can not be smaller than $\lfloor (k + 1)/2 \rfloor$: taking any $k - 1$ of these k clusters, we get a contradiction to the induction hypothesis with $k - 1$ clusters! Similarly, k can not be even because in that case, $\lfloor \frac{k+1}{2} \rfloor = \lfloor \frac{(k-1)+1}{2} \rfloor$. So we may assume that k is odd, and $OPT = (k + 1)/2$.

Every cluster C_i forms a connected subgraph of T. It will be convenient to think of the lengths associated with C_i in the following parts - the path from r to the highest vertex in C_i, and the internal part of C_i (see Figure 1a).

Now consider the bipartite graph $H = (\Gamma, \mathcal{C}, E)$ where $\Gamma = \{t_1, \cdots, t_{(k+1)/2}\}$ is the set of r-paths in the optimal cover (note that $|\Gamma| = OPT = (k+1)/2$), and $\mathcal{C} = \{C_1, \cdots, C_k\}$ is the set of the k heavy clusters. There is an edge $(t_j, C_i) \in E$ iff path t_j visits some vertex of cluster C_i. A set of edges M is said to be a *1-2-matching* from $\mathcal{C}$ to Γ, if the number of edges of M incident on a vertex of $\mathcal{C}$ is exactly 1, and the number of edges of M incident on a vertex of Γ is at most 2. In other words, it is a perfect matching of $\mathcal{C}$ in the graph H' obtained from H by duplicating all the vertices in Γ and the edges in E.



r
Thick line : r-C_i path
Solid lines : internal part of C_i
C_i
(a) Lengths associated with a heavy cluster
r
(b) An r-path on a tree

Fig. 1. Structures of a heavy cluster and an r-path

We claim that H must have a 1-2-matching from $\mathcal{C}$ to Γ. Suppose not - then by Hall's Theorem, we get a set $S \subseteq \mathcal{C}$ such that S has fewer than $|S|/2$ neighbors in Γ. Note that $S \neq \mathcal{C}$, as $\mathcal{C}$ has $OPT > \frac{|\mathcal{C}|}{2}$ neighbors. This implies that the clusters in S are visited completely by fewer than $|S|/2$ r-paths, which contradicts the induction hypothesis with clusters S ($|S| < k$). Let $\pi : \mathcal{C} \to \Gamma$ be a 1-2-matching in H. Since there are $(k+1)/2$ vertices in Γ, and only k vertices in $\mathcal{C}$, there is one vertex in Γ which is matched to only one cluster. Let this vertex be $t_{(k+1)/2}$.

Let $l_1, l_2, \cdots, l_{(k+1)/2}$ denote the lengths of the paths in Γ. Clearly each $l_i \le D$. Assign a *capacity* to each edge $e \in T$, equal to $n_e(t_{(k+1)/2}) + 2\sum_{j=1}^{(k-1)/2} n_e(t_j)$, where $n_e(t_j)$ is the number of times e is traversed in path t_j. Note that the total weighted capacity over all edges is exactly $2\sum_{j=1}^{(k-1)/2} l_j + l_{(k+1)/2} \le kD$. As observed before, every r-path on T has a unique path from r to some leaf which is traversed only once, and all other edges on the r-path are traversed 2 times each (see Figure 1b). Let P denote this r to leaf path in the r-path $t_{(k+1)/2}$. Note that the capacity of every edge $e \in T \setminus P$ is at least twice the *number* of paths of Γ containing e.

We will now *charge* each edge an amount at most its capacity, and show that the total charge over all edges is larger than kD, which would be a contradiction. For cluster C_i, charge an amount equal to the path from r to C_i: each edge on this path is charged one unit against the capacity on that edge attributed to r-path $\pi(C_i)$. So far no edge has a charge more than its capacity - as edges of r-path $t_{(k+1)/2}$ are charged against only once, and other r-paths were doubled. Now we will show that we can further charge an additional amount corresponding to a path on the internal part of each cluster $C_1, \cdots, C_k$.

Consider an edge $e \notin P$ which is on the internal part of some cluster C_i. Let m denote the number of clusters (C_i not included) that appear *below* e in tree T. If $m = 0$, this edge has never been charged so far, and thus has at least 2 units of residual capacity. If $0 < m \le k-1$, by induction on the set of clusters below e, there are at least $(m+2)/2$ r-paths using e. *i.e.* e has a capacity of at least $m+2$. But we have charged e exactly m times so far. So, again we have at least 2 units of residual capacity. For an edge $e \in P$ on the internal part of C_i, a similar argument shows that there is at least 1 unit of residual capacity. The total charge can now be written as follows:

$$\sum_{i=1}^{k} \left[d(r, C_i) + 2 \cdot d((\text{internal part of } C_i) \setminus P) + d((\text{internal part of } C_i) \cap P) \right]$$

The i-th term above corresponds to an r-path covering C_i : where the edges charged just 1 are all on the path P. Since each C_i is a heavy cluster, this is more than D. So the total charge is more than kD, the total capacity! Thus $OPT > (k+1)/2$, and the lemma is proved. ■

We note that the lower bound in Lemma 1 does not hold in general metrics. In fact, even if we require the distance between the heavy clusters $C_1, \cdots, C_k$ to be 'large', there are instances in which $\cup_{i=1}^{k} C_i$ can be covered using $\frac{k}{\log D}$ r-paths.

2.1 An LP Relaxation

We consider the following integer programming formulation for the minimum vehicle routing problem, which is valid even for general metrics. For every r-path T, having length at most D, there is a binary variable x_T. The constraints require that every vertex be covered by at least one such path. The LP relaxation is obtained by dropping the integrality on the variables and is as follows.

$$
\begin{array}{lll}
 & \min \sum_T x_T & \\
 & s.t. & \\
(\mathcal{LP}) & \sum_{T:v\in T} x_T \geq 1 & \forall v \in V \setminus r \\
 & x_T \geq 0 & \forall T : r\text{-path of length at most } D
\end{array}
$$

Although this LP has an exponential number of variables, it can be approximately solved in polynomial time using the framework of Carr & Vempala [7]. The dual separation problem is orienteering, for which there is a 3-approximation algorithm [3]. This implies that we can solve $\mathcal{LP}$ within a factor of 3 in polynomial time, via the ellispoid method. In this section, we show that the integrality gap of $\mathcal{LP}$ on tree metrics is at most a constant.

We may assume, without loss of generality that the tree is binary. Recall the definition of a heavy cluster from Theorem 1. Then, similar to Lemma 1, we have the following lemma.

Lemma 2. *If $C_1, \cdots, C_k$ are k disjoint heavy clusters in the tree, the optimal value of $\mathcal{LP}$ is at least $\frac{k}{32}$.*

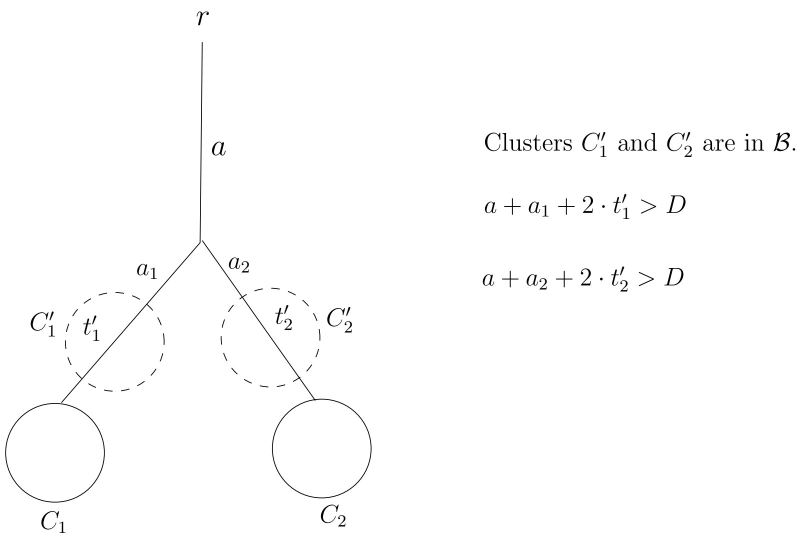
Proof: The dual of $\mathcal{LP}$ is the following.

$$
\begin{array}{ll}
\max \sum_{v\in V\setminus r} p_v & \\
s.t. & \\
\sum_{v\in T} p_v \leq 1 & \forall T : r\text{-path of length at most } D \\
p_v \geq 0 & \forall v \in V \setminus r
\end{array}
$$

We will construct an appropriate dual solution which has value at least $\frac{k}{32}$. Then the lemma would follow by weak duality. The proof also uses the following claim, which we state without a proof.

Claim 1. *For any edge weighted tree H with root s and a weight function w, it is possible to distribute a total profit of 1 among the leaves of H such that the profit contained in any rooted subtree F of H is at most $\frac{w(F)}{w(H)}$.*

First, we preprocess the set of clusters. The internal part of a cluster C_i is the subtree that C_i induces (see Figure 1a). The internal part of C_i is divided into two parts: the edges that lie on the r-path to some other cluster constitute the *through part* of C_i (length denoted by t_i); all other edges constitute the *local part* of C_i (length denoted by l_i). A leaf cluster is one that has no cluster below it in the tree. Note that leaf clusters have zero through length. It is clear that the number of clusters with a branching in their through part is at most the number of leaf clusters. Thus the number of clusters with *no* branching in their through part is $m \geq k/2$. In the rest of the proof we restrict our attention to only these m clusters. For a cluster C_i with no branching in its through part, one r-path that covers it is as follows: take the path from r to C_i, the through part, and twice the local part. This r-path has length $d(r, C_i) + t_i + 2l_i$ which is more than D, as C_i is a heavy cluster. We divide these m clusters into two sets : $\mathcal{A}$ consisting of clusters with $l_i \geq t_i/2$, and $\mathcal{B}$ consisting of clusters with $l_i < t_i/2$. We consider the following two cases.



r
a
a_1
a_2
C'_1
t'_1
t'_2
C'_2
C_1
C_2
Clusters C'_1 and C'_2 are in $\mathcal{B}$.
$a + a_1 + 2 \cdot t'_1 > D$
$a + a_2 + 2 \cdot t'_2 > D$

Fig. 2. r-path Π in case 2

Case 1: $|\mathcal{A}| \geq m/4$. In this case, we only consider clusters in $\mathcal{A}$. The dual solution is as follows: for each such cluster $C_i \in \mathcal{A}$, we shrink the through part to a root and distribute a total profit of 1/4 among the vertices in its local part using Claim 1. Let Π be any r-path with profit more than 1 w.r.t. this dual solution. Let α_i denote the fraction of the local part of C_i in Π. From Claim 1, we have $\frac{1}{4}\sum \alpha_i > 1$. Let C_m denote the cluster of minimum local length visited by Π. Then we have $len(\Pi) \geq d(r, C_m) + \sum \alpha_i \cdot l_i > d(r, C_m) + 4l_m \geq d(r, C_m) + 2l_m + d_m > D$. So the profit in any r-path of length at most D is at most 1.

Case 2: $|\mathcal{B}| \geq 3m/4$. Note that the number of clusters appearing immediately below a branching in the tree is at most twice the number of leaf-clusters. But the number of leaf clusters is at most $|\mathcal{A}| \leq m/4$. Thus, ignoring clusters of $\mathcal{B}$ appearing just after a branching, leaves us with at least $m/4$ clusters. The dual solution here assigns a profit of 1/4 to each remaining cluster, in the same manner as in case 1. Let Π be any r-path with profit more than 1 w.r.t. this dual solution. Suppose clusters $C_1, C_2 \in \mathcal{B}$ appear as leaves in Π. Since we ignore clusters of $\mathcal{B}$ just after any branching, there are clusters $C'_1, C'_2 \in \mathcal{B}$ above C_1 and C_2 (but before any branching) as in Figure 2. We have $D < d(r, C'_j) + t'_j + 2l'_j = a + a_j + t'_j + 2l'_j \leq a + a_j + 2t'_j$ for $j = 1, 2$. So $len(\Pi) \geq a + a_1 + t'_1 + a_2 + t'_2 \geq a + 2\min\{a_1 + t'_1, a_2 + t'_2\} > D$. Thus we may assume that Π has at most one cluster of $\mathcal{B}$ that is a leaf in it. Ignoring the profit from this cluster, we are left with a profit of at least 3/4 from clusters whose through parts are contained in Π. By an argument similar to case 1, we can show that in this case also $len(\Pi) > D$.

In both cases above, we have a feasible dual solution of value $\frac{m}{16} \geq \frac{k}{32}$. ■

Algorithm $minTVR$ (Section 2) finds k^* disjoint heavy clusters such that there is an integral solution of value at most $2k^*$. Using Lemma 2 on these k^* clusters, we obtain that the integrality gap of $\mathcal{LP}$ is $O(1)$.

3 Minimum Vehicle Routing on General Metrics

In this section, we present an approximation algorithm achieving a guarantee of $O(\log D)$ for minimum vehicle routing on general metrics. Using orienteering as a subproblem in a greedy algorithm, one can obtain an $O(\log n)$ approximation algorithm for minimum vehicle routing.[1] However, due to the large running time of the orienteering algorithm, this approach yields an algorithm with a running time of $O(n^{12})$. The algorithm that we present here is simpler and has a running time of $O(n^2 \cdot \log n \cdot \log D)$. This algorithm uses an algorithm for unrooted vehicle routing, which is obtained from Arkin et al. [1].

The basic idea of the algorithm for rooted vehicle routing is that, if an r-path visits some points a "large" distance from the root, it resembles an unrooted path (with smaller length) over just those vertices. More concretely, we divide the vertices of the graph into $\lg D$ parts, roughly according to their distance from the root, and solve an unrooted vehicle routing in each part (with appropriate path length). We state (without proof) the following theorem.

Theorem 2. *There is an $O(\log D)$-approximation algorithm for the minimum vehicle routing problem on general metrics, that runs in $O(n^2 \cdot \lg n \cdot \lg D)$ time.*

As a consequence of this Theorem, we also obtain a bi-criteria approximation algorithm for minimum vehicle routing. In particular, if we are allowed to violate the deadline D by a small factor ϵ, we can cover all the vertices using $O(\log \frac{1}{\epsilon}) \cdot OPT$ r-paths. Here OPT is the minimum number of r-paths of length at most D, required to cover all vertices. This result can be compared to the tour-splitting heuristic discussed in Li et al. [9]. The best guarantee one can obtain using tour-splitting is an $(O(\frac{1}{\epsilon}), 1+\epsilon)$ bi-criteria approximation.

Corollary 3. *For every $0 < \epsilon < 1$, there is an $(O(\log \frac{1}{\epsilon}), 1+\epsilon)$ bi-criteria approximation algorithm for minimum vehicle routing.*

We note that the above bicriteria approximation is also obtained independently by Khuller et al. [12].

3.1 Integrality Gap of the Linear Relaxation

We consider the LP relaxation $\mathcal{LP}$ for minimum vehicle routing introduced in Section 2.1, and show that its integrality gap is at most $O(\log D)$ in general

[1] Even though we use a constant factor approximation for orienteering, the greedy framework only gives an $O(\log n)$ guarantee.

metrics. We first show that a similar linear program for unrooted vehicle routing has a constant integrality gap. Then in a manner similar to the algorithm of Section 3, we obtain the result for rooted vehicle routing. We state (without proofs) the following theorems.

Theorem 4. *The LP relaxation for unrooted vehicle routing on general metrics has an integrality gap of at most 49/3.*

Corollary 5. *The LP relaxation for rooted vehicle routing on general metrics has an integrality gap of at most* $O(\log \frac{D}{D-d_{max}+1})$*, where* d_{max} *is the maximum distance of any vertex from the root.*

Note also that a trivial randomized rounding (as in set cover) shows that the integrality gap of $\mathcal{LP}$ is at most $\log n$. So $\mathcal{LP}$ has an integrality gap at most $O(\min\{\log D, \log n\})$. It will be interesting to know if the integrality gap of $\mathcal{LP}$ is bounded above by a constant, irrespective of the metric.

4 The $\boldsymbol{OPT = 1}$ Promise Problem

In this section, we look at the problem of *finding* a small set of r-paths covering all the vertices, given a promise that there exists a single r-path that covers all vertices. Note that even testing whether all vertices can be covered by one path is NP-complete. So unless P=NP, it is not possible to find in polynomial time, a path that covers V, even if we know that there exists one. Here we present an algorithm that finds a cover using at most 14 r-paths.

This algorithm is based on guessing the structure of the optimal path, and approximating it. First we need a definition from Blum et al. [6]. For an s-t path P, we define the *excess* of path P to be $\epsilon(P) = d(P) - d(s,t)$, where $d(P)$ denotes the length of path P, and $d(s,t)$ is the shortest path distance between s and t. Given vertices s and t, and a target k, the minimum excess path problem is to find an s-t path of minimum possible excess that contains *at least* k vertices. Blum et al. [6] gave a $2+\delta$ approximation algorithm for minimum excess path, for any fixed $\delta > 0$.[2]

We divide the vertex set into roughly $\lg D$ *blocks* as follows:

$$V_j = \begin{cases} \{v : D-1 < d(r,v) \le D\} & j = 0 \\ \{v : D-2^j < d(r,v) \le D-2^{j-1}\} & 1 \le j \le \lfloor \lg D \rfloor \\ \{v : 0 < d(r,v) \le D - 2^{\lfloor \lg D \rfloor}\} & j = \lfloor \lg D \rfloor + 1 \end{cases}$$

Note that if an r-path visits a vertex in V_{j+2} after a vertex in V_j, its length would be more than $D - 2^j + d(V_{j+2}, V_j) > D - 2^j + 2^j = D$. So the optimal path visits vertices of V_j strictly after all vertices of V_{j+2}. Thus we can split the optimal path O into two paths O_1 and O_2 such that O_1 visits the vertices

[2] We note that a 2-approximation to minimum excess path would imply a 2 approximation to the $OPT = 1$ promise problem considered here. However, no such algorithm is currently known.

in even numbered blocks, and O_2 visits vertices in odd numbered blocks. So O_1 (O_2) is obtained by restricting O to vertices from even (odd) numbered blocks, and short-cutting over the other vertices. O_1 is monotone across even blocks, and O_2 is monotone across odd blocks. This suggests that we can approximate these paths using a dynamic program.

4.1 Approximating Path O_1

We know that O_1 is monotone over the even blocks $V_0, V_2, \cdots$. Let l_j denote the length of the part of O_1 in V_j (which is contiguous), and d_j the shortest path distance between the first and last vertices of V_j in O_1. Let $\epsilon_j = l_j - d_j$ denote the excess in block V_j. Also, let Δ denote the total length of edges going from one block (say V_j) to the next (V_{j-2}) in path O_1. Then the length of O_1 is $\sum_{j:even}(d_j + \epsilon_j) + \Delta \leq D$. We will denote $\sum_{j:even} \epsilon_j$ by ϵ.

In this Section, we show a weaker approximation guarantee of covering O_1 with 16 r-paths. A block V_j with $\epsilon_j > \frac{\epsilon}{6}$ is called *heavy*. Clearly there are at most 5 heavy blocks, and these can be guessed (there are at most $\log^5 D$ possibilities). Consider the non-heavy blocks: each has excess $\epsilon_j \leq \epsilon/6$. So the set of non-heavy blocks can be partitioned (in a greedy fashion) into 6 groups such that each group is contiguous and has a total excess of at most $\frac{\epsilon}{3}$. Since these groups are contiguous, they can also be guessed (there are at most $\log^5 D$ possibilities). We now describe two algorithms: one that covers heavy blocks, and the other for the groups of non-heavy blocks defined above.

Algorithm TS: Each heavy block is covered separately using a tour splitting algorithm TS, which works as follows. Suppose V_j is the heavy block to be covered.

1. Compute an approximate minimum Hamilton path H on V_j.
2. Split H into 3 pieces $\sigma_1, \sigma_2, \sigma_3$, such that each has length at most $\frac{d(H)}{3}$.
3. Output the r-paths $\{(r \cdot \sigma_i) : 1 \leq i \leq 3\}$. Here, $r \cdot \sigma_i$ is the path obtained by concatenating an edge from r to path σ_i.

To verify that each of the paths output has length at most D, note that the minimum Hamilton path on V_j has length at most 2^j. This is because every vertex in V_j is distant at least $D - 2^j$ from r, and O_1 (of length $\leq D$) when restricted to $r \cup V_j$ is a Hamilton path along with an edge from r. Christofides' heuristic [8] achieves an approximation guarantee of 3/2 for the Hamilton path problem. Using this, $d(H) \leq \frac{3}{2} \cdot 2^j$, and $d(\sigma_i) \leq 2^{j-1}$ ($i = 1, 2, 3$). Thus $d(r \cdot \sigma_i) \leq D - 2^{j-1} + 2^{j-1} = D$.

Algorithm EXS: Every group in the partition of non-heavy blocks is covered separately, using a dynamic program EXS. Let $W_1, \cdots, W_q$ denote the blocks in the group to be covered, in increasing order of distance from r. Let ϵ'_j be the excess of block W_j ($1 \leq j \leq q$). For every block W_j and a pair of vertices $u_j, v_j \in W_j$, we use the minimum excess algorithm to compute a u_j-v_j path in W_j, covering at least $|W_j|$ vertices (*i.e.* all of W_j). Let $A[u_j, v_j, j]$ denote the length of this path. The following dynamic program finds the best way to piece such paths together to obtain a single monotone path covering $\bigcup_{j=1}^{q} W_j$.

$$
\begin{aligned}
PATH[v_1, 1] &= \min\{d(r, u_1) + A[u_1, v_1, 1] : u_1 \in W_1\}, \qquad v_1 \in W_1 \\
PATH[v_j, j] &= \min \left\{ \begin{array}{r} PATH[v_{j-1}, j-1] + d(v_{j-1}, u_j) + A[u_j, v_j, j] : \\ v_{j-1} \in W_{j-1}, u_j \in W_j \end{array} \right\}, \\
&\qquad\qquad\qquad v_j \in W_j \quad \& \quad 2 \le j \le q
\end{aligned}
$$

Finally we output the path corresponding to $\min\{PATH[*, q]\}$.

Suppose $u_j^*, v_j^* \in W_j$ denote the entry and exit points of O_1, for each block W_j, $1 \le j \le q$. This dynamic program will consider a path corresponding to these points. The length of such a path would be at most $d(r, u_1^*) + d(u_1^*, v_1^*) + \sum_{j=2}^{q}(d(v_{j-1}^*, u_j^*) + d(u_j^*, v_j^*)) + 3\sum_{j=1}^{q} \epsilon_j'$. This follows since there is a u_j^*-v_j^* path (namely $O_1 \cap W_j$) covering W_j of excess ϵ_j', and the minimum excess algorithm is a 3-approximation. But from the way a group is constructed, $\sum_{j=1}^{q} \epsilon_j' \le \frac{\epsilon}{3}$. So the length of this path is at most the length of O_1, which is at most D.

To summarize, the overall algorithm for approximating O_1 is as follows.

1. Guess the heavy blocks $b_1, \cdots, b_5$ (repetitions allowed).
2. Guess the partition of non-heavy blocks into groups $G_1, \cdots, G_6$.
3. For each block b_l ($l = 1, \cdots, 5$), use algorithm TS to cover it.
4. For each group G_t ($1 \le t \le 6$), use algorithm EXS to cover it.
5. Over all the guesses, return the solution of minimum size which is feasible.

We now argue the performance guarantee of this algorithm. Suppose there are $0 \le h \le 5$ heavy blocks. Then the total excess of the remaining blocks is at most $\epsilon - \frac{h}{6}\epsilon$. So the number of groups of non-heavy blocks will be at most $6 - h$. Now, the total number of paths used by this algorithm would be at most $6 - h + 3h \le 16$. With some more work, we can approximate O_1 using at most 7 r-paths (see [14]). Since O_2 can also be approximated by the same algorithm, we obtain the following.

Theorem 6. *Given an instance of minimum vehicle routing on general metrics, having an optimal solution that uses just one vehicle, there is a polynomial time algorithm that obtains a solution using at most 14 vehicles.*

Acknowledgements

We thank Shuchi Chawla for helpful discussions.

References

1. Esther M. Arkin, Refael Hassin, and Asaf Levin. Approximations for Minimum and Min-max Vehicle Routing Problems. *Journal of Algorithms*, 2005.
2. Esther M. Arkin, Joseph S. B. Mitchell, and Giri Narasimhan. Resource-constrained Geometric Network Optimization. *SCG '98: Proceedings of the Fourteenth Annual Symposium on Computational Geometry*, pages 307–316, 1998.

3. Nikhil Bansal, Avrim Blum, Shuchi Chawla, and Adam Meyerson. Approximation Algorithms for Deadline-TSP and Vehicle Routing with Time Windows. *Proceedings of the Thirty-sixth Annual ACM Symposium on Theory of Computing*, pages 166–174, 2004.
4. Reuven Bar-Yehuda, Guy Even, and Shimon (Moni) Shahar. On Approximating a Geometric Prize-Collecting Traveling Salesman Problem with Time Windows. *Proc. of ESA*, pages 55–66, 2003.
5. C. Bazgan, R. Hassin, and J. Monnot. Approximation Algorithms for Some Vehicle Routing Problems. *Discrete Applied Mathematics*, 146:27–42, 2005.
6. Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation Algorithms for Orienteering and Discounted-Reward TSP. *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, pages 46–55, 2003.
7. B. Carr and S. Vempala. Randomized meta-rounding. *32nd ACM Symposium on the Theory of Computing*, pages 58–62, 2000.
8. N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. *GSIA, CMU-Report 388*, 1977.
9. C.L.Li, D. Simchi-Levi, and M. Desrochers. On the distance constrained vehicle routing problem. *Operations Research*, 40:790–799, 1992.
10. M. Desrochers, J.Desrosiers, and M. Solomon. A New Optimization Algorithm for the Vehicle Routing Problem with Time Windows. *Operation Research*, 40:342–354, 1992.
11. M. Kantor and M. Rosenwein. The Orienteering Problem with Time Windows. *Journal of the Operational Research Society*, 43:629–635, 1992.
12. Samir Khuller, Azarakhsh Malekian, and Julian Mestre. To Fill or not to Fill: The Gas Station Problem. *Manuscript*, 2006.
13. A. Kohen, A. R. Kan, and H. Trienekens. Vehicle Routing with Time Windows. *Operations Research*, 36:266–273, 1987.
14. Viswanath Nagarajan and R. Ravi. Minimum Vehicle Routing with a Common Deadline. *https://server1.tepper.cmu.edu/gsiadoc/WP/2006-E53.pdf*, 2006.
15. M. Savelsbergh. Local Search for Routing Problems with Time Windows. *Annals of Operations Research*, 4:285–305, 1985.
16. K. C. Tan, L. H. Lee, K. Q. Zhu, and K. Ou. Heuristic Methods for Vehicle Routing Problems with Time Windows. *Artificial Intelligence in Engineering*, pages 281–295, 2001.