An Approximation Algorithm for Vehicle Routing with Compatibility Constraints[*]

Miao Yu, Viswanath Nagarajan, and Siqian Shen[†]

Department of Industrial and Operations Engineering,
University of Michigan at Ann Arbor

Abstract

We study a multiple-vehicle routing problem with a minimum makespan objective and compatibility constraints. We provide an approximation algorithm and a nearly-matching hardness of approximation result. We also provide computational results on benchmark instances with diverse sizes showing that the proposed algorithm (i) has a good empirical approximation factor, (ii) runs in a short amount of time and (iii) produces solutions comparable to the best feasible solutions found by a direct integer program formulation.

Keywords vehicle routing, minimum makespan, approximation algorithm

1 Introduction

Vehicle Routing Problems (VRPs) are classical and extensively studied combinatorial optimization problems, which aim to find the optimal routing decisions for one or multiple vehicles traveling from the depot(s) to serve demands at various locations. Depending on specific applications, various types of VRPs are formulated and solved by exact or heuristic approaches. We refer the interested readers to [31] and [23] for comprehensive surveys of models and algorithms for different VRPs, and review the ones that are the most relevant to this paper below.

A significant amount of VRP literature focus on single-vehicle VRPs, where only one vehicle is allowed. Without any additional constraints, a basic single-vehicle VRP problem is equivalent to the classic Traveling Salesmen Problem (TSP) [see, 2]. There exist many heuristic approaches, approximations, and exact algorithms for the TSP and $\frac{3}{2}$ is the best known approximation factor [14]. Some other single-VRPs are the orienteering problem [21, 13], TSP with time window [7, 6], Prize-collecting TSP [5] and k-TSP [19].

For multiple-vehicle VRPs, different variants are studied, including multiple TSP [26], capacitated VRP [10, 11], VRP with time windows [29], and the dial-a-ride problem [16]. The objective in all these studies is to minimize the total traveling cost of all vehicles.

However, with multiple vehicles, a natural objective is to minimize the makespan of the system, i.e., the maximum travel distance among all vehicles. This objective has received relatively less attention, see, e.g., [1, 18, 3, 24].

[*]A preliminary version of this paper appeared as [32].

[†]Corresponding Author: Prof. Siqian Shen, 2793 IOE, 1205 Beal Avenue, Ann Arbor, MI, USA 48109-2011, E-mail: siqian@umich.edu

In this paper, we study a multiple-vehicle routing problem with a minimum makespan objective where each vehicle can only serve a subset of the locations. Such "compatibility constraints" arise in applications such as medical home care delivery. Here one needs to dispatch shared vehicles to visit patients at their homes. Each patient may require different skill-sets from medical teams, who are conveyed by different vehicles, and we also need to balance the workload of different medical staff teams dispatched with the vehicles [28].

1.1 Problem Definition and Formulation

In this paper, we study the minimum makespan VRP with compatibility constraints (VRPCC). Given a graph $G = (V, E)$ with node set $V = \{0, 1, \dots, n\}$ and edge set $E = \{(i, j) : \ i \in V, \ j \in V\}$, we assume that the depot is located at node 0 and customers are located at the nodes in $V^+ = V \setminus \{0\}$. Each edge $(i, j) \in E$ has a non-negative length c_{ij}, which follows triangle inequality, i.e., $c_{ij} \le c_{il} + c_{lj}$ for all $i, \ j, \ l \in V$. We assume that the minimum distance is at least 1 and edge lengths are symmetric, i.e., $c_{ij} = c_{ji}$ for all $(i, j) \in E$. A fleet K of vehicles with $K = \{0, 1, \dots, m-1\}$ is located at the depot, and each can visit a subset of customers in V^+. (Our results also extend easily to the case of multiple depots, but we focus on the single depot case for simplicity.) We assume that each vehicle $k \in K$ can only visit a subset of nodes $V_k \subset V^+$, based on matches of vehicles and customers' service types. Our goal is to find a routing decision to assign each vehicle a route such that: (a) the nodes visited by vehicle $k \in K$ are in the set V_k; (b) each node must be visited exactly once; and (c) the maximum traveling cost over all vehicles is minimized.

Let $(x_{ij}^k, \ (i, j) \in E)$ be a binary vector of decision variables, such that $x_{ij}^k = 1$ if we assign vehicle $k \in K$ to visit node $j \in V$ right after node $i \in V$, and 0 otherwise. Let $(u_i^k, \ i \in V)$ be the indicator vector parameter, such that $u_i^k = 1$ if $i \in V_k$, and $u_i^k = 0$ otherwise. VRPCC can be formulated as the following integer program.

$$\textbf{MIP:} \ \underset{\mathbf{x}, \ \tau}{\text{minimize}} \quad \tau \tag{1}$$

$$\text{subject to} \ \sum_{(i,j) \in E} c_{ij} x_{ij}^k \le \tau \quad \forall k \in K \tag{2}$$

$$\sum_{(0,j) \in E} x_{0j}^k = 1 \quad \forall k \in K \tag{3}$$

$$\sum_{(i,v) \in E} x_{iv}^k - \sum_{(v,j) \in E} x_{vj}^k = 0 \quad \forall v \in V, \ \forall k \in K \tag{4}$$

$$\sum_{k \in K} \sum_{(i,j) \in E} x_{ij}^k = 1 \quad \forall j \in V^+ \tag{5}$$

$$x_{ij}^k \le u_j^k \quad \forall (i, j) \in E, \ j \in V^+, \ \forall k \in K \tag{6}$$

$$\sum_{i,j \in S, \ i \ne j} x_{ij}^k \le |S| - 1 \quad \forall S \subset V^+, \ \forall k \in K \tag{7}$$

$$x_{ij}^k \in \{0, 1\} \quad \forall (i, j) \in E, \forall k \in K. \tag{8}$$

In the above formulation, constraints (2) set τ as the maximum traveling cost among all vehicles; constraints (3) ensure that each vehicle leaves the depot; constraints (4) ensure that the same number of vehicles arrive at and depart from each customer node; constraints (5) ensure that each node in V^+ will be visited exactly once; constraints (6) ensure that each vehicle can only visit the nodes in its compatibility set; constraints (7) are sub-tour elimination constraints avoiding cycles that do not contain the depot; finally, $\mathbf{x}$ is a binary decision vector according to constraints (8).

Although there are exponentially many constraints in (7), this can be addressed by constraint generation as follows. We initially start with no sub-tour elimination constraints in the model.

When the solver finds a feasible solution satisfying all the current constraints, we detect cycles that do not pass the depot. (This can be found very efficiently by a simple graph search.) If we do not find such cycles then the MIP model (1)–(8) is already solved. Otherwise, we add the corresponding sub-tour elimination constraints from (7), and the solver continues solving this new modified model. We iterate this process until a valid solution is found.

In our computations, we solve this **MIP** model (1)–(8) using a state-of-the-art solver (Gurobi), and compare its performance to our approximation algorithm. We also use **MIP** to compute lower bounds for VRPCC.

Recall that an α-approximation algorithm for a minimization (resp. maximization) problem always produces a solution of objective at most (resp. at least) α times the optimal.

1.2 Main Results

In this paper, we focus on the approximability of the VRPCC.

Theorem 1. *VRPCC cannot be approximated to within a factor of* $(1-o(1))\cdot\ln n$*, unless* $\mathcal{NP}=\mathcal{P}$.

Theorem 2. *There is a* $2\lceil\ln n\rceil+1$*-approximation algorithm for VRPCC.*

Both the hardness result and the algorithm are based on relations to the set cover problem, which is known to have a tight approximability threshold of $\ln n$ [see, 15, 17]. Recall that the set cover problem involves selecting the smallest number of sets from a given collection so as to cover all elements of a ground set. The main idea for Theorem 1 is to reduce the min-sum objective in set-cover to the min-max objective in VRPCC. This is done by making several copies of the set-cover instance and "rotating" the set-element relations in each of these instances so as to balance the load across all sets used in an optimal solution. See Section 2 for details.

Theorem 2 is based on applying the set-cover greedy algorithm on an implicit set system: such an approach has been used in a number of approximation algorithms, e.g., [25, 30]. However, the "max coverage" subproblem that we need to solve is different in order to handle the min-max objective. This corresponds to the maximum coverage problem with group budgets [12, 27] (we will define it formally later), and we can apply their approach using an approximation algorithm for the orienteering problem [13]. We observe that using an approximation algorithm for the related k-TSP problem [20] leads to a slightly better constant (2 instead of $2+\epsilon$) in the approximation ratio, but more importantly this approach is far easier to implement. See Section 3 for details.

Our computational results show that the empirical approximation factor (solution value from the approximation algorithm divided by the best lower bound found) is much smaller than the theoretical one. Moreover, the running time of the approximation algorithm is much smaller than **MIP**: the time taken is less than 2 minutes on almost all instances. Also, the solution found by our approximation algorithm is very close (or better) than the best solution found by **MIP** even after 2 hours. See Section 4 for details.

1.3 Related Work

For VRPs with a min-max objective, most current literature focuses on heuristic methods [see, e.g., 9, 22] and approximation algorithms [see, e.g., 3, 18, 24]. In particular, constant-factor approximation algorithms are known for min-max (unrooted) path/tour cover [3], min-max (rooted) path/tour cover [18] and min-max TSP with non-uniform speeds [24]. To the best of our knowledge, approximation algorithms for min-max VRPs with compatibility constraints have not been studied, and this paper aims to fill this gap. Compared to these previous results, we show that the problem with compatibility constraints does not admit any constant-factor approximation.

In this paper, we use a set cover based technique to derive our algorithm. Set cover and maximum coverage problems are classic problems in combinatorial optimization with wide applications in various settings. A greedy algorithm that repeatedly picks the set covering the maximum number of uncovered elements, yields a $(\ln n)$-approximation for the set cover problem and $\frac{e}{e-1}$-approximation for the maximum coverage problem [see, 15]. This is also known to be best-possible unless $\mathcal{NP} = \mathcal{P}$ [17]. Our algorithm relies on the framework of maximum coverage with group budgets, introduced by [12]; the approximation ratio was recently improved in [27]. A crucial subroutine in implementing this framework for VRPCC is the k-TSP problem, which finds a rooted tour with minimum total cost covering k nodes in a given graph. A constant-factor approximation algorithm for k-TSP was given by [8] and later improved by [19], [4], and [20] to 3, $2 + \epsilon$ and 2, respectively.

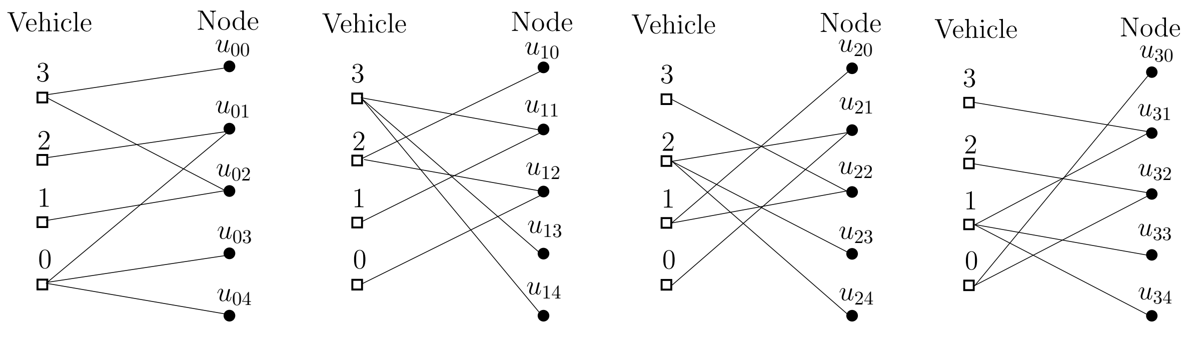
2 Hardness of Approximation for VRPCC

We prove the Theorem 1 by showing the set cover problem is polynomial-time reducible to VRPCC, and therefore VRPCC is at least as hard as the set cover problem.

Proof of Theorem 1. In a set cover instance, we are given a ground set $\mathcal{U}$ and a family $\mathcal{S}$ of subsets of $\mathcal{U}$. A cover is a subfamily $\mathcal{C} \subseteq \mathcal{S}$ of sets whose union is $\mathcal{U}$. The objective is to find a cover that uses the fewest sets in $\mathcal{S}$. Let $[t]$ denote the integer set $\{0, 1, \ldots, t-1\}$ for any $t \geq 1$.

Given an input $(\mathcal{U}, \mathcal{S})$ of set cover, we index elements in $\mathcal{U}$ as $0, 1, \ldots, n-1$ and let the collection of subsets be $\mathcal{S} = \{S_i : i \in [m]\}$. The reduction constructs a graph with $mn + 1$ nodes that are partitioned into a depot-node r and m disjoint groups $W_i = \{u_{ij}, j \in [n]\}$ for each $i \in [m]$. There are edges of cost 0 between each pair of nodes in the same group $\{W_i\}_{i \in [m]}$ and an edge of cost 0.5 between r and all other nodes. There are no edges between nodes of different groups. Let T denote this edge-weighted graph and $c_{a,b}$ the shortest path distance between nodes a and b in T.

The VRPCC instance has m vehicles, so $K = [m]$. The compatibility constraints $\{V_k : k \in [m]\}$ are based on "rotating" the sets in $\mathcal{S}$ and are defined as follows. For any vehicle $k \in [m]$, group $i \in [m]$ and $j \in [n]$, node $u_{ij} \in V_k$ if and only if $j \in S_{k'}$, where $k' = (k + i) \mod m$. Figure 1 shows an example for an instance with $m = 4$ and $n = 5$ where the relationship in W_0 represents the original collection $\mathcal{S}$. This reduction is clearly polynomial time in m and n.



(a) Group W_0 (b) Group W_1 (c) Group W_2 (d) Group W_3

Figure 1: An illustration of compatibility constraints on different groups.

We argue that solving the set cover instance $(\mathcal{U}, \mathcal{S})$ is equivalent to solving the above VRPCC instance. Let SC^* and CC^* denote corresponding optimal solutions to set cover and VRPCC, respectively. Let $c(SC^*)$ and $c(CC^*)$ denote corresponding optimal objectives.

We first show that $c(CC^*) \leq c(SC^*)$. Let C contain indices of all sets in SC^*. Then, we construct the following solution to VRPCC. We define $C_i := \{(c - i) \mod m \mid c \in C\}$ for each group $i \in [m]$. Note that by definition of the compatibility constraints in group i, the vehicles in C_i can cover all nodes in W_i. So the VRPCC solution involves routing each vehicle $k \in [m]$ to the groups $\{i \in [m] : k \in C_i\}$. The total cost for any vehicle $k \in [m]$ is then $|\{i \in [m] : k \in C_i\}| = |C|$. So the VRPCC objective is also $|C| = c(SC^*)$. This implies $c(CC^*) \leq c(SC^*)$.

Conversely, we show that $c(SC^*) \leq c(CC^*)$. Consider the optimal VRPCC solution CC^*. As $c(CC^*)$ is the maximum cost over all m vehicles, the total cost to cover all nodes is at most $m \times c(CC^*)$. This implies that there exists a group, say W_i, which is visited by *at most* $c(CC^*)$ vehicles. Let $D_i \subseteq [m]$ denote the set of vehicles that visit group W_i; note $|D_i| \leq c(CC^*)$. Then it follows (due to the compatibility constraints for W_i) that $D = \{(d + i) \mod m \mid d \in D_i\}$ forms a valid set-cover, i.e. $\cup_{\ell \in D} S_\ell = \mathcal{U}$. So the optimal set-cover value $c(SC^*) \leq |D| = |D_i| \leq c(SC^*)$.

Combining both, we conclude that $c(CC^*) = c(SC^*)$. The result in Theorem 1 now follows from the hardness for set-cover that there is no $(1 - o(1)) \cdot \ln n$ approximation algorithm unless $\mathcal{P} = \mathcal{NP}$ [17]. This completes the proof. □

3 An $O(\log n)$-Approximation Algorithm

In this section, we propose an approximation algorithm to VRPCC. Our algorithm starts with a "guess" B on the optimal value, which can be verified within a binary search scheme (see Algorithm 2). Then it iteratively picks a route for each vehicle that covers (approximately) the maximum number of new nodes. We iterate this process until all nodes are covered.

We start by introducing the maximum coverage problem with group budgets (MCG). Here, we are given a ground set for MCG, X, and a collection of subsets of X, $\mathcal{C} = \{S_1, S_2, \ldots, S_m\}$. We are also given a partition of $\mathcal{C}$ into k groups $G_1, G_2, \ldots, G_k$. A solution to MCG is a subset $H \subset \mathcal{C}$ such that $|G_i \cap H| \leq 1$ for all $i = 1, \ldots, k$, i.e. at most one set can be chosen from each part. The objective is to maximize the number of elements in X covered by H.

Example: Consider $X = \{1, 2, 3, 4, 5\}$, $S_1 = \{1, 2, 3\}, S_2 = \{4, 5\}, S_3 = \{1, 2, 5\}, S_4 = \{1, 5\}$ and $k = 2$. Suppose that $G_1 = \{S_1, S_3\}$ and $G_2 = \{S_2, S_4\}$. Then the optimal solution is $H = \{S_1, S_2\}$ which covers all 5 elements.

[12] gave a greedy algorithm to solve MCG by iteratively picking one set from each part that covers the maximum number of uncovered elements. Crucially, the algorithm works with an oracle model $\mathcal{O}$ that takes as input a ground set X' and an index i and outputs a set $S_j \in G_i$ such that $|S_j \cap X'|$ is maximized. They showed that this greedy algorithm is a $\frac{1}{1+\rho}$-approximation algorithm for MCG given a $\frac{1}{\rho}$-approximate oracle. Later, [27] proposed a better $(1 - e^{-1/\rho})$-approximation algorithm, given a $\frac{1}{\rho}$-approximate oracle, based on a linear program rounding algorithm. However, this algorithm relies on using the ellipsoid method to solve LPs, which is not practical for our purpose.

Specializing MCG to the VRPCC setting, we obtain the following.

Problem. *MCG-VRP*
Input: A node subset $X \subset V$, a fleet K of vehicles and a budget $B \geq 0$.
Output: Routes $\{A_i \subseteq V_i : i \in K\}$ for each vehicle with each route of cost at most B.
Objective: maximize $| \cup_{i \in K} A_i \cap X|$, the number of nodes covered.

In this case, the oracle $\mathcal{O}$ corresponds to the orienteering problem. Formally, $\mathcal{O}(Y, B, i)$ involves computing a route for vehicle $i \in K$ (originating from r) with cost at most B that covers the maximum number of nodes in Y. The compatibility constraints are enforced in the definition of this oracle instance. The complete algorithm is described in Algorithm 1.

```
Algorithm 1: Greedy Algorithm for MCG-VRP
  input  : A fleet $K$ of vehicles, a subset $X \subset V$ and a budget $B$
  output: A set $H$ of routes with cost at most $B$, one route for each vehicle
1 $X' \leftarrow X$
2 for $i \in K$ do
3     $A_i = \mathcal{O}(X' \cap V_i, B, i)$
4     $X' \leftarrow X' \setminus A_i$
5 end
6 return $H = \{A_i : i \in K\}$
```

Theorem 3 (Corollary 1 in [12])**.** *Algorithm 1 is a $\frac{1}{1+\rho}$-approximation algorithm for the MCG-VRP, assuming a $\frac{1}{\rho}$-approximate oracle $\mathcal{O}$.*

The current best approximation ratio for the orienteering problem is $(2+\epsilon)$ with running time $n^{O(1/\epsilon^2)}$ [13]; so this is polynomial time only for constant $\epsilon > 0$. However, this algorithm as well other constant-factor approximation algorithms for orienteering are rather complex to implement. To simplify the implementation, we instead use a $(1,\beta)$-*bicriteria* approximation algorithm for orienteering, which violates the cost by a factor $\beta \geq 1$ but covers the optimal number of nodes (for a cost B route). This corresponds to the k-TSP problem: given a graph with root r and target k, find a min-cost route (originating from r) that covers at least k nodes. Given a β-approximation algorithm for k-TSP, we can obtain a $(1,\beta)$-*bicriteria* approximation for orienteering by just running a binary search on k to output the route having highest k and cost at most βB. The best approximation ratio for k-TSP is $\beta = 2$ from [20]. This algorithm is also much more efficient than those for orienteering: in fact it is used as a subroutine in all algorithms for orienteering. Using Theorem 3, we obtain:

Corollary 4. *If we use a $(1,\beta)$-*bicriteria *approximation algorithm for oracle $\mathcal{O}$, then Algorithm 1 is a $(\frac{1}{2},\beta)$-*bicriteria *approximation algorithm for MCG-VRP.*

Finally, we use Algorithm 1 iteratively until all nodes are covered, and perform a binary search on the "guess" B. The details are displayed in Algorithm 2.

The following lemma shows VRPCC can be solved by iterating Algorithm 1 at most $\lceil \log_2 n \rceil$ times for the correct guess of B.

Lemma 5. *If we use a $(1,\beta)$-bicriteria approximation algorithm for oracle $\mathcal{O}$ then Algorithm 2 achieves a $(1+\epsilon)\beta\lceil \log_2 n \rceil$ approximation ratio for VRPCC.*

Proof. By the definition of upper/lower bounds (u and l) and the binary-search on B, it is clear that the parameter *Solve* is true (resp. false) at the end of the inner while-loop (line 9) when B is set to u (resp. l). Note also that for any B where *Solve* is true at the end of the inner while-loop, Algorithm 1 must have produced (in each iteration of the inner while-loop) a solution that covers at least half of the current set X: as each iteration adds makespan at most βB (by Corollary 4), the resulting VRPCC solution has makespan at most $(\beta\lceil \log_2 n \rceil) \cdot B$. In particular, for the choice $B = u$ at the end of the algorithm, we obtain that the makespan of Algorithm 2 is $ALG \leq \beta\lceil \log_2 n \rceil \cdot u$.

Suppose the optimal value for VRPCC is B^* and consider any $B \geq B^*$. Note that the optimal value of the MCG-VRP instance (X, B, K) is $|X|$ for any $X \subseteq V^+$: hence using Corollary 4,

Algorithm 2: Approximation Algorithm for VRPCC

```
input  : A network $G = (V, E)$, a fleet $K$ of vehicles
output: Routing assignment for each vehicle in $i \in K$
1  Initialize routes $\tau_i = \emptyset$ for all $i \in K$, $X \leftarrow V^+$.
2  Initialize upper-bound $u = 2\sum_{(i,j)\in E} c_{ij}$, lower-bound $l = 0$,  and a tolerance threshold $\epsilon$ for
     the binary search.
3  while $u - l \geq \epsilon$ do
4      $B \leftarrow \frac{u+l}{2}$, $Solve \leftarrow$ true
5      while  $X \neq \emptyset$ do
6          $\{A_i : i \in K\}$ = solution from Algorithm 1 for MCG-VRP with input $K, X, B$
7          if $|\cup_{i\in K} A_i \cap X| < |X|/2$ then $Solve \leftarrow$ false, break
8          Update $X \leftarrow X \setminus (\cup_{i\in K} A_i)$ and $\tau_k \leftarrow \tau_k \circ A_k$ for all $k \in K$
9      end
10     if $Solve$ then $u \leftarrow B$
11     else $l \leftarrow B$
12 end
13 return routes in $\tau_i$ for $i \in K$
```

Algorithm 1 covers at least half the nodes in X. So *Solve* is true at the end of inner while-loop for any $B \geq B^*$. As *Solve* is false at the end when $B = l$, we obtain $B^* \geq l$. So we have:

$$ALG \leq \beta\lceil\log_2 n\rceil \cdot u \leq \beta\lceil\log_2 n\rceil(l + \epsilon) \leq \beta\lceil\log_2 n\rceil(B^* + \epsilon) \leq (\beta\lceil\log_2 n\rceil)(1 + \epsilon)B^*.$$

The last inequality uses the fact that $B^* \geq 1$ as all distances are assumed to be at least one. Therefore, Algorithm 2 is a $(1 + \epsilon)\beta\lceil\log_2 n\rceil$-approximation algorithm. □

Using $\beta = 2$ and $\epsilon \leq \frac{1}{n}$, we obtain a $2\lceil\log_2 n\rceil+1$-approximation algorithm for VRPCC. Instead of the greedy approach from [12], if we used the LP-based approach in [27] to solve MCG-VRP, we obtain a $(2 \ln n+1)$-approximation algorithm. This completes the proof of Theorem 2.

For the computational results, we only tested the greedy approach which has a slightly worse approximation ratio, but is a lot simpler to implement.

The time complexity of Algorithm 2 depends on the complexity of the oracle $\mathcal{O}$. Suppose the complexity of $\mathcal{O}$ is $T(n)$, then the complexity of Algorithm 1 is $O(mT(n))$. For Algorithm 2, the inner while loop executes at most $\lceil\log n\rceil$ as we halve the size of X in each loop. The outer loop is used to conduct binary search for optimal budget, which takes $\log_2 \frac{C}{\epsilon}$ steps where $C = 2\sum_{(i,j)\in E} c_{ij}$. Therefore, the overall complexity of Algorithm 2 is $O\left(\log_2(\frac{C}{\epsilon}) \cdot \log_2(n) \cdot m \cdot T(n)\right)$.

4 Computational Results

To evaluate the performance of the proposed approximation algorithm, we conduct numerical studies on tailored instances generated from Solomon's benchmark for VRPs [29]. First, we compare the proposed approximation algorithm against the **MIP** on small instances with up to 25 customers. Then, we extend our experiments to the instances with up to 100 customers.

There are three categories of the original instances based on the nodes distribution: the R type where nodes are randomly distributed, the C type where nodes are distributed in clusters, and the RC type where some nodes are randomly distributed and the rest are clustered. We only use the node location from Solomon's benchmark and customize our instances as follows. We sample the desired number of nodes from the benchmark instances and compute the distance matrix. We consider two types of VRPCC instances: one with tight compatibility constraints where each node can be visited by few numbers of vehicles, and the other with relaxed compatibility constraints where each node can be visited by more number of vehicles. In our tailored instances, we randomly generate compatibility constraints such that we allow one vehicle to visit a node with 30% probability for tight instances and 70% probability for relaxed instances. Each instance is labeled by its type R/C/RC, number n of nodes and number k of vehicles.

To implement the approximation algorithm, we use the 5-approximation algorithm for k-TSP problem from [19] to serve as our oracle $\mathcal{O}$. Although the best known result is a 2-approximation from [20], the one that we use has an easier implementation while achieving satisfactory empirical results in our tests. We use $\epsilon = 10^{-3}$ in Algorithm 1. After solving the problem, we perform local search, including 2-opt [31] and relocation, to improve the solutions. The relocation procedure used in the local search works as follows. After we finish the 2-opt, we attempt to reassign the nodes in the most lengthy tour to another compatible vehicle and insert them to the best position of the tour. We repeat this process until no improvements can be made.

We code our algorithm in Java and execute the tests on a computer with an Intel Core i7-3770 CPU running at 3.4 GHz and 8 GB of RAM. We use Gurobi 7.5.1 as the mixed integer programming solver. We report the results with 10 minutes and 2 hours (120 minutes) time limits for **MIP** solver.

We evaluate the proposed algorithm against the **MIP** on small instances with up to 26 nodes and large instances with up to 101 nodes. For each test instance, we report the lower bounds, LB_1 (LB_2), upper bounds, UB_1 (UB_2), found by **MIP** within the 10 minutes (2 hours) time limit, the CPU time for **MIP** in seconds (Time) if the problem is solved within the time limit, the solution found by approximation algorithm (Obj), the CPU time for the approximation algorithm in seconds (Time), the empirical approximation ratio computed by $\frac{\text{Obj}}{\text{LB}_2}$, and the ratio between the objective from the approximation algorithm and the best upper bound from **MIP**, $\frac{\text{Obj}}{\text{UB}_2}$. In the second **MIP** run (2 hours limit), we highlight (with bold font) those lower/upper bounds that were improved from the first **MIP** run (10 minutes limit).

We first test our proposed algorithms on small instances with up to 25 customers (i.e., 26 nodes) and different compatibility constraints. We refer to Tables 1 and 2 in [33] for detailed results and summarize key observations below. First, solving VRPCC through **MIP** is challenging for the state-of-art solver and the instances with relaxed compatibility constraints are more difficult to solve. Out of twelve instances each, two and six instances cannot be solved to optimality within the time limit for instances with tight and relaxed compatibility constraints, respectively. Our proposed algorithm works well for these small instances as most of them can be solved within one second. Comparing the best lower bound found by **MIP**, our proposed algorithm yields good empirical approximation ratios: The ratios are within 1.16 for all instances with tight compatibility constraints and within 2 for ten out of twelve instances with relaxed compatibility constraints.

Tables 1 and 2 summarize the results for instances with up to 100 customers (i.e., 101 nodes) and different compatibility constraints. It becomes very difficult to solve for good lower bound by using the state-of-art solver for **MIP**, even given a two-hour time limit. Regarding the runtime, our proposed algorithm is capable of solving two types of instances with 101 nodes in 15 seconds and 270 seconds, respectively. The empirical approximation ratios are maintained within 8: we think that this is a pessimistic bound as the lower bounds from **MIP** seem poor. In addition, our results are comparable to the best **MIP** solutions found within the time limit as ratios $\frac{\text{Obj}}{\text{UB}_2}$, are close to 1: in fact in many cases the approximation algorithm finds a better solution.

Acknowledgement. S. Shen's work was partially supported by National Science Foundation

Table 1: Numerical results for instances with up to 101 nodes and tight compatibility constraints

Instance	MIP (10 minutes)			MIP (2 hours)			Approximation			
	LB_1	UB_1	Time (s)	LB_2	UB_2	Time (s)	Obj	Time (s)	$\frac{\text{Obj}}{LB_2}$	$\frac{\text{Obj}}{UB_2}$
C-n21-k6	87.30	87.30	5.79	87.30	87.30	6.47	87.30	0.01	1.00	1.00
C-n41-k10	109.70	114.30	600.03*	**114.30**	114.30	1772.48	114.50	0.09	1.00	1.00
C-n61-k14	55.07	–	600.15*	**55.35**	**102.70**	7200.15*	102.90	1.50	1.86	1.00
C-n81-k18	72.69	–	600.02*	**72.79**	**125.50**	7200.25*	117.40	2.82	1.61	0.94
C-n101-k22	32.13	–	600.02*	**32.39**	–	7200.22*	122.70	15.44	3.79	–
R-n21-k6	120.70	120.70	3.23	120.70	120.70	4.10	124.90	0.02	1.03	1.03
R-n41-k10	100.70	103.20	600.05*	101.90	**103.00**	7200.05*	149.70	0.28	1.47	1.45
R-n61-k14	77.80	121.10	600.04*	**81.20**	**120.10**	7200.08*	126.80	1.03	1.56	1.06
R-n81-k18	54.02	–	600.04*	**54.88**	–	7200.52*	117.60	3.73	2.14	–
R-n101-k22	51.80	–	600.05*	**56.40**	–	7200.26*	121.30	11.27	2.15	–
RC-n21-k6	138.80	138.80	0.80	138.80	138.80	1.01	138.80	0.03	1.00	1.00
RC-n41-k10	194.90	194.90	277.41	194.90	194.90	295.56	194.90	0.33	1.00	1.00
RC-n61-k14	108.40	257.30	600.03*	**114.00**	**146.30**	7200.11*	170.50	2.09	1.50	1.17
RC-n81-k18	49.81	–	600.03*	**50.34**	–	7200.17*	170.60	4.67	3.39	–
RC-n101-k22	47.08	–	600.04*	**47.61**	–	7200.97*	178.50	9.32	3.75	–

*: computation reaches time limit; –: no upper bounds found by **MIP** within time limit

Table 2: Numerical results for instances with up to 101 nodes and relaxed compatibility constraints

Instance	MIP (10 minutes)			MIP (2 hours)			Approximation			
	LB_1	UB_1	Time (s)	LB_2	UB_2	Time (s)	Obj	Time (s)	$\frac{\text{Obj}}{LB_2}$	$\frac{\text{Obj}}{UB_2}$
C-n21-k6	29.85	82.80	600.10*	**41.85**	**81.60**	7200.38*	80.60	0.06	1.93	0.99
C-n41-k10	18.25	129.20	600.08*	**18.44**	**86.50**	7200.18*	89.00	3.70	4.83	1.03
C-n61-k14	15.06	–	600.30*	**15.43**	–	7200.09*	90.40	32.31	5.86	–
C-n81-k18	13.36	–	600.07*	**15.48**	–	7200.39*	117.00	100.65	7.56	–
C-n101-k22	14.00	–	600.47*	14.00	–	7200.25*	117.00	262.38	8.35	–
R-n21-k6	56.58	79.90	600.03*	**65.09**	79.90	7200.05*	82.60	0.29	1.27	1.03
R-n41-k10	42.22	181.60	600.07*	**43.14**	**107.80**	7200.16*	97.80	3.08	2.27	0.91
R-n61-k14	35.83	–	600.24*	**36.48**	–	7200.09*	94.30	24.85	2.58	–
R-n81-k18	29.09	–	600.10*	**32.53**	–	7200.11*	103.90	89.01	3.19	–
R-n101-k22	26.18	–	600.19*	26.18	–	7200.05*	99.80	231.50	3.81	–
RC-n21-k6	90.20	90.20	141.53	90.20	90.20	142.03	90.40	0.27	1.00	1.00
RC-n41-k10	17.77	228.50	600.11*	**19.89**	**125.20**	7200.18*	160.50	4.41	8.07	1.28
RC-n61-k14	26.44	–	600.09*	**26.82**	–	7200.19*	117.50	31.39	4.38	–
RC-n81-k18	24.93	–	600.07*	**29.23**	–	7200.22*	121.00	114.88	4.14	–
RC-n101-k22	23.91	–	600.15*	23.91	–	7200.40*	124.20	268.86	5.19	–

*: computation reaches time limit; –: no upper bounds found by **MIP** within time limit

grants CMMI-1636876, 1727618. V. Nagarajan's work was supported in part by NSF CAREER grant CCF-1750127.

References

[1] D. Applegate, W. Cook, S. Dash, and A. Rohe. Solution of a min-max vehicle routing problem. *INFORMS Journal on Computing*, 14(2):132–143, 2002.

[2] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, 2006.

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

[4] S. Arora and G. Karakostas. A $2 + \epsilon$ approximation algorithm for the k-MST problem. *Mathematical Programming*, 107(3):491–504, 2006.

[5] E. Balas. The prize collecting traveling salesman problem. *Networks*, 19(6):621–636, 1989.

[6] N. Bansal, A. Blum, S. Chawla, and A. Meyerson. Approximation algorithms for deadline-tsp and vehicle routing with time-windows. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 166–174. ACM, 2004.

[7] J. Barrios and J. Godier. Fleet sizing for flexible carsharing systems: Simulation-based approach. *Transportation Research Record: Journal of the Transportation Research Board*, (2416): 1–9, 2014.

[8] A. Blum, R. Ravi, and S. Vempala. A constant-factor approximation algorithm for the k MST problem. In *Proceedings of the Twenty-eighth Annual ACM Symposium on Theory of Computing*, pages 442–448. ACM, 1996.

[9] J. Carlsson, D. Ge, A. Subramaniam, A. Wu, and Y. Ye. Solving min-max multi-depot vehicle routing problem. *Lectures on Global Optimization*, 55:31–46, 2009.

[10] P. Chalasani and R. Motwani. Approximating capacitated routing and delivery problems. *SIAM Journal on Computing*, 28(6):2133–2149, 1999.

[11] M. Charikar, S. Khuller, and B. Raghavachari. Algorithms for capacitated vehicle routing. *SIAM Journal on Computing*, 31(3):665–682, 2001.

[12] C. Chekuri and A. Kumar. Maximum coverage problem with group budget constraints and applications. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 72–83. Springer, 2004.

[13] C. Chekuri, N. Korula, and M. Pál. Improved algorithms for orienteering and related problems. *ACM Transactions on Algorithms (TALG)*, 8(3):23, 2012.

[14] N. Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.

[15] V. Chvatal. A greedy heuristic for the set-covering problem. *Math. Oper. Res.*, 4(3):233–235, 1979.

[16] J.-F. Cordeau and G. Laporte. The dial-a-ride problem: Models and algorithms. *Annals of Operations Research*, 153(1):29–46, 2007.

[17] I. Dinur and D. Steurer. Analytical approach to parallel repetition. In *Proceedings of the Forty-sixth Annual ACM Symposium on Theory of Computing*, pages 624–633. ACM, 2014.

[18] G. Even, N. Garg, J. Könemann, R. Ravi, and A. Sinha. Min–max tree covers of graphs. *Operations Research Letters*, 32(4):309–315, 2004.

[19] N. Garg. A 3-approximation for the minimum tree spanning k vertices. In *Proceedings of the 37th Annual Symposium on Foundations of Computer Science*, FOCS '96, pages 302–309, Washington, DC, USA, 1996. IEEE Computer Society.

[20] N. Garg. Saving an epsilon: A 2-approximation for the k-MST problem in graphs. In *Proceedings of the Thirty-seventh Annual ACM Symposium on Theory of Computing*, pages 396–402. ACM, 2005.

[21] B. L. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34 (3):307–318, 1987.

[22] B. L. Golden, G. Laporte, and É. D. Taillard. An adaptive memory heuristic for a class of vehicle routing problems with minmax objective. *Computers & Operations Research*, 24(5): 445–452, 1997.

[23] B. L. Golden, S. Raghavan, and E. A. Wasil. *The Vehicle Routing Problem: Latest Advances and New Challenges*, volume 43. Springer Science & Business Media, 2008.

[24] I. L. Gørtz, M. Molinaro, V. Nagarajan, and R. Ravi. Capacitated vehicle routing with nonuniform speeds. *Mathematics of Operations Research*, 41(1):318–331, 2016.

[25] P. N. Klein and R. Ravi. A nearly best-possible approximation algorithm for node-weighted steiner trees. *J. Algorithms*, 19(1):104–115, 1995.

[26] W. Malik, S. Rathinam, and S. Darbha. An approximation algorithm for a symmetric generalized multiple depot, multiple travelling salesman problem. *Operations Research Letters*, 35 (6):747–753, 2007.

[27] C. S. Martin and M. R. Salavatipour. Minimizing latency of capacitated k-tours. *Algorithmica*, pages 1–20, 2017.

[28] S. Salmond and P. E. Ropis. Job stress and general well-being: A comparative study of medical-surgical and home care nurses. *Medsurg Nursing*, 14(5):301, 2005.

[29] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[30] Z. Svitkina and É. Tardos. Min-max multiway cut. In *7th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX*, pages 207–218, 2004.

[31] P. Toth and D. Vigo. *Vehicle Routing: Problems, Methods, and Applications*, volume 18. SIAM, 2014.

[32] M. Yu, V. Nagarajan, and S. Shen. Minimum makespan vehicle routing problem with compatibility constraints. In *International Conference on AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, pages 244–253. Springer, 2017.

[33] M. Yu, V. Nagarajan, and S. Shen. An approximation algorithm for vehicle routing with compatibility constraints *Optimization Online* `http://www.optimization-online.org/DB_HTML/2018/06/6645.html`. 2018