An algorithm for the metric multiple depots capacitated vehicle routing problem with restocking and capacity two

Chao Xu* Yichen Yang† Qian Zhang‡

Abstract

The capacitated vehicle routing problem (CVRP) is one of the most well known NP-hard combinatorial optimization problem. Single depot CVRP with general metric is NP-hard even for fixed capacity $Q = 3$, while polynomial time solvable for fixed capacity $Q \leq 2$. We consider the variant of CVRP where restocking is available. We show that if there is a constant number of depots, then the problem can be solved in polynomial time when $Q = 2$. More generally, when there are p depots without a starting vehicle, there is a polynomial time algorithm for constant p.

1 Introduction

In this paper, we study a variant of the capacitated vehicle routing problem (CVRP). In the CVRP problem, there is a single depot, and multiple customers each have unit demand. Each vehicle has a capacity Q. Each vehicle travels from the depot, delivers the inventory to at most Q customers, and returns to the depot. We consider the variant that allows restocking. After completing a trip, the vehicle can return to a depot, and restock the inventory for more delivery. The objective is to minimize the total distance traveled over all vehicles.

We now introduce the case for multiple depots. In the *multiple depots capacitated vehicle routing with restocking problem* (MDCVRRP), there are k depots with enough inventory and n customers with unit demand. On a subset of depots, there are vehicles with capacity Q. The vehicles can restock at any depot before any further delivery. We are interested in finding a tour for each vehicle so that the demand of all the customers can be satisfied, each vehicle returns to the depot where it starts, and the total travel distance is minimized.

Since the traveling salesman problem (TSP) is a special case of MDCVRRP when the number of depot equals one and the capacity of the vehicle is unlimited, MDCVRRP is strongly NP-hard even in the Euclidean plane [7]. MDCVRRP is closely related to the classic CVRP. Almost all the special cases of CVRP studied in the literature are NP-hard. The only polynomial solvable special case of CVRP that relates to our result is the case when $Q \leq 2$ and the number of depots equals one [1]. However, for $Q \leq 2$, it is unknown if the multiple depot case can be solved in polynomial time. For a general metric, CVRP is shown to be APX-complete for any fixed $Q \geq 3$ [1].

Our Contribution We show that if there are a constant number of depots, then MDCVRRP with capacity $Q = 2$ is solvable in polynomial time. In fact, we prove a stronger result. Even if the number of depots is large, as long as the number of depots that does not have a starting vehicle is constant, then MDCVRRP with capacity $Q = 2$ can be solved in polynomial time.

*Yahoo! Research, New York, NY 10003, USA chao.xu@oath.com
†Department of Mathematics, East China University of Science and Technology, Shanghai, China agamemnon314@hotmail.com
‡School of Civil and Environmental Engineering, Cornell University, Ithaca, NY 14853, USA qz283@cornell.edu

Organization The rest of this paper is organized as follows. In section 2, we introduce some notations and notions we used in our work, and describe the problem . In section 3, we introduce the matching T-join, and solving the minimum cost matching T-join problem. In section 4, we describe an algorithm for MDCVRRP algorithm which runs in polynomial time for $Q = 2$ and constant number of depots without a starting vehicle.

2 Preliminaries

$G = (V, E)$ is a undirected complete graph, where V is partitioned into two sets of vertices D and U. We will abuse the notation and write $G = (D, U, E)$ to denote $G = (V, E)$, where $V = D \cup U$. The vertices in D are called *depots*. We also fix a set $S \subseteq D$, to be the set of depots with a starting vehicle. The set U represents the set of n *customers*. Each customer has a unit demand, and each vehicle has capacity Q, in our article, Q is 2, except specified otherwise. Let $c : E \to \mathbb{R}^+$ be a non-negative function over the edges, which we call the *cost*. The cost function c is a *metric*, if c is non-negative and satisfies the triangle inequality, that is, $c(uw) + c(wv) \geq c(uv)$ for every $u, v, w \in V$.

Let $F \subseteq E$ be a set of edges. We use $c(F) = \sum_{uv \in F} c(uv)$ to denote the total edge cost of F, and $V(F)$ to denote the vertex set of F. We use $F[S]$ to denote the set of edges in F that have both endpoints in S. Let G be a graph, then $G[S]$ is the *induced subgraph* whose vertex set is S and whose edge set is $E[S]$. If a statement applies to graphs, then we extend a statement to a set of edges F by considering the graph $(V(F), F)$. For any $F \subseteq E$, we use $Odd(F)$ to denote the set of odd degree vertices. A *forest* is a set of edges without cycles. We say a set of edges F *covers* a set of vertices U if $U \subseteq V(F)$. A maximal set of connected vertices is called a *component*. Note in particular, when we say components for a set of edges F, the isolated vertices are not counted, since they are not in $(V(F), F)$.

If uv and vw are edges, the *splitting-off* operation [6] on v removes the edge uv and vw, and add an edge uw. This operation maintains the parity of the degree of v, and preserves the degree of all other vertices. $\delta(X)$ is the set of edges with exactly one endpoint in X. $\delta(x)$ is defined to be $\delta(\{x\})$.

Now, we define the abstract problem we tries to solve in detail.

Problem 1 (Q-MDCVRRP(p))
INPUT: *A graph $G = (D, U, E)$, $S \subseteq D$ such that $|D \setminus S| \leq p$ and c a metric cost on E.*
OUTPUT: *A set of closed walks on G such that:*

1. *each closed walk in $\mathcal{W}$ contains a depot in S;*
2. *each closed walk in $\mathcal{W}$ contains at most Q successive vertices in U;*
3. *the total cost of $\mathcal{W}$ is minimum.*

Note the problem Q-MDCVRRP(p) captures the case when there are at most p empty depots. We define Q-MDCVRRP to be Q-MDCVRRP(∞). Namely, there is no bound on the size of $|D \setminus S|$.

Because the union of closed walks is a set of edges. Instead of a set of closed walk, we just have to find the set of edges used by the set of closed walk. This is what we mean a "solution" to the Q-MDCVRRP(p). Indeed, by the Eulerian theorem [3], there is a linear time algorithm to find the actual set of closed walks given a union of closed walks.

T-joins and matchings A *matching* is a set of disjoint edges. For a set $T \subseteq V$ of even cardinality. A set of edges $J \subseteq E$ is called a T*-join* if $|J \cap \delta(v)|$ is odd if and only if $v \in T$.

Lemma 2.1 (Korte & Vygen [5]) *Given an undirected graph $G = (V, E)$ and a set $T \subseteq V$ of even cardinality, there exists a T-join in G if and only if $|C \cap T|$ is even for each component C of G.*

Theorem 2.2 (Edmonds & Johnson [4]) *Given an undirected graph $G = (V, E)$ and a set $T \subseteq V$ of even cardinality, if the edge costs are nonnegative, the minimum cost T-join can be found in $O(|V|^3)$ time.*

3 Matching T-join

Consider a complete graph $G = (D, U, E)$ with a metric cost c. For $T \subseteq D$, a T-join J is a *matching T-join*, if J covers U and $J[U]$ is a matching.

Note in particular, each component in a matching T-join J contains at least one vertex in D. Indeed, consider a component of J that contains a vertex $u \in U$, because u has even non-zero degree, so it has degree at least 2. Because $J[U]$ is a matching, so u has a neighbor that is not in U, therefore must be an element in D. Since the cost is a metric, there always exists an minimum cost matching T-join where all vertices in U has degree 2. Indeed, this is because if U has larger degree, a splitting-off operation can be applied without increase the cost, and parity is preserved.

3.1 Auxiliary graph

We will show that finding the minimum cost matching T-join in G can reduce to solving a minimum cost T-join problem on the auxiliary graph of G. First, we describe the construction of the auxiliary graph G' of G:

1. for each $u \in U$, duplicate a vertex u'. Let $U' = \left\{u' | u \in U\right\}$.
2. let $V' = U \cup U' \cup D$;
3. let $E' = E \cup \left\{u'v | u' \in U', v \in D\right\}$;
4. let $G' = (D, U \cup U', E')$;

We consider a cost function $c' : E' \to \mathbb{R}$ as follows. If $e \in E$, then $c'(e) = c(e)$. If $uv \in E$ where $u \in U, v \in D$, then $c'(u'v) = c(uv)$.

Lemma 3.1 *For any T such that $U \cup U' \subseteq T$, the minimum cost T-join J^* in G' such that $|\delta(u) \cap J^*| = 1$ for all $u \in U \cup U'$ can be found in $O(|V|^3)$.*

Proof: Here we present a constructive proof. First, find a minimum cost T-join J^* in $O(|V|^3)$ time. If there exists $u \in U \cup U'$ such that $|\delta(u) \cap J^*| > 1$, we can apply splitting-off operation on u with respect to J^* until we get only one edge incident to u. Indeed, if $u \in U$, then it only incidents to vertices in $U \cup D$, and $G'[D \cup U]$ is a complete graph, so splitting-off is feasible. If $u \in U'$, then it only incidents to vertices in D, and $G'[D]$ is a complete graph, so splitting-off is also feasible. This operation can be applied at most $|E|$ times, as the number of edges decrease by one after each splitting-off operation. Hence, we can find the desired structure in $O(|V|^3)$ time. $\square$

Lemma 3.2 *The minimum cost matching T-join in G can be found in $O(|V|^3)$.*

Proof: Let $T' = T \cup U \cup U'$. Consider an arbitrary minimum cost T'-join $J^{*\prime}$ in G' such that $|\delta(u) \cap J^{*\prime}| = 1$ for all $u \in U \cup U'$. Shrink $J^{*\prime}$ to J by contracting each vertex $u \in U$ with the corresponding vertex $u' \in U'$. It is easy to check that J is a matching T-join in G with $c(J^{*\prime}) = c(J)$.

On the other hand, consider an arbitrary minimum cost matching T-join J^* in G, we can construct a matching T'-join J' in G' as follows:

1. start with $J' = J^*$;
2. for each vertex $u \in U$, find an edge $ud \in J'$ for some $d \in D$. Delete edge ud from J' and add edge $u'd$ to J'.

It is easy to check that J' is a T'-join in G' and $c(J') = c(J^*)$.

From the above, the minimum cost matching T-join can be found by solving the minimum cost T'-join problem on G', which implies the minimum cost matching T-join can be found in $O(|V|^3)$. $\square$

4 Algorithm for 2-MDCVRRP(p)

In this section we solve 2-MDCVRRP(p). Recall we have a graph $G = (D, U, E)$, with metric cost function c and $S \subseteq D$.

Definition A *constrained forest* F is a forest such that:

1. Each component of F has exactly one vertex in S;
2. For all $v \in U$ covered by F, we have $|\delta(v) \cap F| = 2$;
3. $F[U]$ is a matching;

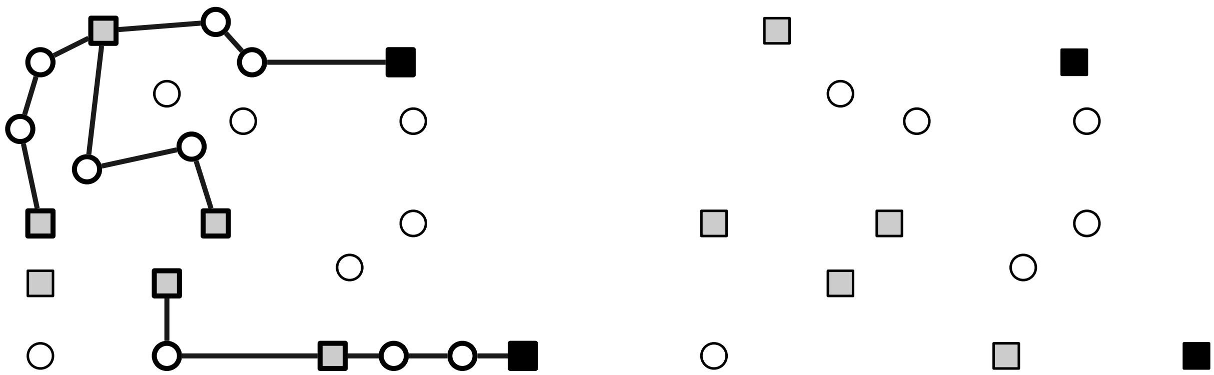

Figure 4.1. On the left, is an example of a constrained forest F for graph G. On the right is the corresponding vertices in the graph G_F. White circle are elements in U, black square are elements in S, and gray square are elements in $D \setminus S$.

See Figure 4.1 for an example of a constrained forest. Note that for a constrained forest F, $Odd(F) \subseteq D$. Now we define a new graph G_F. The new graph G_F is obtained from G by retaining the depot vertices in $V(F)$, and remove all covered U vertices in $V(F)$. Formally, let $D_F = D \cap V(F)$, $U_F = U \setminus V(F)$, $V_F = D_F \cup U_F$, $E_F = E[V_F]$. We define $G_F = (D_F, U_F, E_F)$. See Figure 4.1 for an example.

Given constrained forest F, if the optimal solution to 2-MDCVRRP(p) contains all the edges in F, and only uses the vertices in D_F, then such an optimal solution can be constructed in polynomial time. This is the main goal of the next theorem.

Lemma 4.1 *Let W^* be an optimal solution to 2-MDCVRRP(p) of $G = (D, U, E)$, $F \subseteq W^*$ a constrained forest with $V(F) \cap D = V(W^*) \cap D$. Let $T = Odd(F) \cup U_F$. If J is a minimum cost matching T-join of G_F, then $W = J \cup F$ is an optimal solution to 2-MDCVRRP(p) on G.*

Proof: By the definition of the constrained forest, $W^* \setminus F$ is a matching T-join of G_F. Hence we have $c(J) \leq c(W^*) - c(F)$. Note that $J \cup F$ is a solution to 2-MDCVRRP(p) of G. W is also a solution to 2-MDCVRRP(p) of G, therefore $c(W^*) \leq c(W)$. It follows that

$$c(W^*) \leq c(W) = c(J) + c(F) \leq c(W^*) - c(F) + c(F) = c(W^*)$$

Therefore we have shown W is an optimal solution. □

At this point, we obtain the algorithm in Figure 4.2. Next, we analyze its running time.

Theorem 4.2 *2-MDCVRRP(p) can be solved in $O(|V|^3 \cdot |\mathcal{F}|)$ time, where $\mathcal{F}$ is the set of constrained forests of G.*

```
CONSTRAINEDSOLUTION(G = (D,U,E),c)
  for F a constrained forest in G:
      D_F ← D ∩ V(F)
      U_F ← U \ V(F)
      T ← Odd(F) ∪ U_F
      G_F ← (D_F, U_F, E[D_F ∪ U_F])
      J ← minimum cost matching T-join of G_F
      W ← J ∪ F
      add W into the list of candidates
  return the minimum cost candidate
```

Figure 4.2. The algorithm

Proof: By Lemma 4.1, the optimal solution can be found by enumerating all the possible constrained forests and find minimum cost $(Odd(F) \cup U_F)$-join in G_F. Hence Figure 4.2 is correct. The for-loop ran $|\mathcal{F}|$ time. The running time inside the for-loop is finding the matching T-join, which takes $O(|V|^3)$ time. □

4.1 Number of constrained forests

In this section, we count the number of constrained forests. For each constrained forest, each component in the forest must contain at least one vertex in $D \setminus S$. Indeed, no vertex in U can be a leaf because it has degree 0 or 2. There are at least 2 leaves in each tree, and exactly one is in S, hence the other leaf has to be in $D \setminus S$. We proceed with a sequence of lemmas that eventually counts the number of constrained forests.

Lemma 4.3 *Let $Y_1, \ldots, Y_t$ be disjoint subsets of D such that each set contains exactly one vertex in S. Let $D' = \bigcup Y_i \setminus S$ and $q = |D'|$. The number of constrained forests F with components $C_1, \ldots, C_t$, such that $C_i \cap D = Y_i$ is at most $(q+1)^{q-t} n^{2q}$.*

Proof: Let F be a constrained forest with components $C_1, \ldots, C_t$ such that $C_i \cap D = Y_i$. We look at tree T_i induced on C_i. T_i contains $|Y_i|$ vertices in D, and at most $2(|Y_i|-1)$ vertices in U. Recall Cayley's formula, which states the number of labeled tree on n vertices is n^{n-2} [2]. We can fix a spanning tree on Y_i, and assign at most 2 vertices in U on each edge. Since each vertex in U has degree 2, we obtain a bijection from these labeled spanning trees to a component in a constrained forest. Therefore, there can be $|Y_i|^{|Y_i|-2}|U|^{2(|Y_i|-1)}$ possible trees that can act as the ith component of a constrained forest satisfies the property in the lemma. Each component is independent, hence we take their product.

$$\begin{aligned}
&\prod_{i=1}^{t} |Y_i|^{|Y_i|-2}|U|^{2(|Y_i|-1)} \\
\leq &\prod_{i=1}^{t} (q+1)^{|Y_i|-2} n^{2|Y_i|-1} \\
= &(q+1)^{\sum_{i=1}^{t}(|Y_i|-2)} n^{2\sum_{i=1}^{t}|Y_i|-1} \\
= &(q+1)^{q-t} n^{2q}.
\end{aligned}$$

□

Lemma 4.4 *For $X_1, \ldots, X_t$ a partition of $X \subseteq D \setminus S$. The number of constrained forests F with components $C_1, \ldots, C_t$, such that $C_i \cap (D \setminus S) = X_i$ is at most $k^t (q+1)^{q-t} n^{2q}$, where $q = |X|$.*

Proof: We pick a sequence of t distinct vertices in S, $s_1, \dots, s_t$, there are at most k^t choices. By the Lemma 4.3, where we take $Y_i = X_i \cup \{s_i\}$. We can show the number of constrained forests F satisfies this lemma to be $k^t(q+1)^{q-t}n^{2q}$. □

Lemma 4.5 *Let $X \subseteq D \setminus S$ a set of q elements. The number of constrained forests F such that $V(F) \cap (D \setminus S) = X$ is $k^q(q+1)^{2q}n^{2q}$.*

Proof: There can be no more than q^t ways to partition X to t classes. By Lemma 4.4, the total number of constrained forest is bounded by

$$\sum_{t=1}^{q} q^t k^t (q+1)^{q-t} n^{2q} = O(k^q(q+1)^q n^{2q}).$$

□

Lemma 4.6 *There are at most $O(k^p n^{2p})$ different constrained forests, where $p = |D \setminus S|$ and is a constant.*

Proof: Take X over all subsets of $D \setminus S$. Using Lemma 4.5, we have The number of constrained forests is therefore at most $\sum_{q=0}^{p} \binom{p}{q} O(k^q(q+1)^q n^{2q}) \leq O(2^p k^p (p+1)^p n^{2p}) = O(k^p n^{2p})$. □

Theorem 4.7 (Main Theorem) *2-MDCVRRP(p) can be solved in $O(|V|^3 \cdot k^p n^{2p})$ time, where k is the number of depots and n is the number of customers.*

Proof: By Lemma 4.6, there are at most $O(k^p n^{2p})$ different constrained forests. We apply Theorem 4.2 to obtain the desired running time. □

In particular, because $p \leq k-1$, this implies if the number of depots k is a constant, we also have a polynomial time algorithm.

Theorem 4.8 *2-MDCVRRP can be solved in $O(|V|^{2k+1})$ time if there is at most k depots.*

This matches the result when there is a single depot [1].

Note we can also show if the number of depots without a starting vehicle is unbounded, the problem is NP-hard.

Theorem 4.9 *2-MDCVRRP is NP-hard in general.*

Proof: Indeed, every TSP instance can transform to a 2-MDCVRRP instance. For each point in TSP, we create a depot and two customer at the same location. Let there be exactly one depot with a starting vehicle. It's easy to see the optimal 2-MDCVRRP tour would reflect an optimal tour in the TSP. □

References

[1] Tetsuo Asano, Naoki Katoh, Hisao Tamaki, and Takeshi Tokuyama. Covering points in the plane by k-tours: Towards a polynomial time approximation scheme for general k. In *Proceedings of the Twenty-ninth Annual ACM Symposium on Theory of Computing*, STOC '97, pages 275–283, New York, NY, USA, 1997. ACM.

[2] A. Cayley. A theorem on trees. *Quart. J. Math.*, 23:376–378, 1889.

[3] William J. Cook, William H. Cunningham, William R. Pulleyblank, and Alexander Schrijver. *Combinatorial Optimization*. John Wiley & Sons, Inc., 2011.

[4] Jack Edmonds and Ellis L. Johnson. Matching, euler tours and the chinese postman. *Mathematical Programming*, 5(1):88–124, Dec 1973.

[5] Bernhard Korte and Jens Vygen. *Combinatorial Optimization: Theory and Algorithms*. Springer, 2007.

[6] L Lovász. *Combinatorial Problems and Exercises*. Elsevier, 1994.

[7] Christos H. Papadimitriou. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237 – 244, 1977.