Multiple Resource Network Voronoi Diagram

Ahmad Qutbuddin and KwangSoo Yang

Abstract—Given a spatial network and a set of service centers from k different resource types, a Multiple Resource Network Voronoi Diagram (MRNVD) partitions the spatial network into a set of Service Areas that can minimize the total cycle-distances of graph-nodes to allotted k service centers with different resource types. The MRNVD problem is important for critical societal applications such as assigning essential survival supplies (e.g., food, water, gas, and medical assistance) to residents impacted by man-made or natural disasters. The MRNVD problem is NP-hard; it is computationally challenging due to the large size of the transportation network. Previous work proposed the Distance bounded Pruning (DP) approach to produce an optimal solution for MRNVD. However, we found that DP can be generalized to reduce the computational cost for the minimum cycle-distance. In this paper, we extend our prior work and propose a novel approach that reduces the computational cost. Experiments using real-world datasets from five different regions demonstrate that the proposed approach creates MRNVD and significantly reduces the computational cost.

Index Terms—Network Voronoi Diagram, Resource Allocation, Route Optimization

1 Introduction

GIVEN a spatial network and a set of service centers from k different resource types (e.g., gas stations, grocery stores, shelters, hospitals, etc.), a Multiple Resource Network Voronoi Diagram (MRNVD) partitions the spatial network into a set of Service Areas (SAs) that can minimize the total cycle-distances of graph-nodes to allotted k service centers with different resource types. Fig. 1a shows an example input of MRNVD consisting of a graph with 25 graph-nodes (i.e., $A, B, \dots, Y$) and service centers with four types (i.e., Type1:(I, P, Y), Type2:(N, Q, V), Type3:(B, G, X), and Type4:(C, M, W)). Fig. 1b shows an example output of MRNVD where the graph is partitioned such that every graph-node is allotted to four service centers with different types. The objective is to minimize the total cycle-distances of graph-nodes to allotted k service centers with different types. The MRNVD problem is NP-hard (a proof is provided in Section 1.3). Intuitively, the problem is computationally challenging because of the large size of the transportation network.

1.1 Application Domain

The MRNVD problem is important for critical societal applications such as assigning essential resources (e.g., food, water, gas, and medical assistance) to residents impacted by man-made or natural disasters. The objective of MRNVD is to minimize the total cycle-distances such that residents can quickly visit their allotted service centers and back to their original locations. MRNVD can help us to identify the most efficient route to visit all required service centers. The objective will help reduce traffic because it will encourage more people to use the shortest driving route. The multiple resource constraint also helps law enforcement agencies to manage an emergency easier and more effectively.

In addition, the simple format of the information is vital to communicate effectively during an emergency. MRNVD provides a compact and simple representation of Service Areas (SAs) that can mitigate panic and chaos, and allow for efficient delivery of critical information to the public. Planners in an emergency can allocate enough resources to service areas based on the number of residents and provide clear and concise explanations of service areas for the emergency first response. Policymakers can look at service areas and take actions to mitigate the potential traffic congestion. Due to the inherent risk of service disruptions or changes during an emergency, people may choose alternative routes for available service centers. MRNVD can help us quickly identify the nearest service areas that can provide better reliable services. Examples of the benefits of MRNVD Service Areas are provided in Table 1.

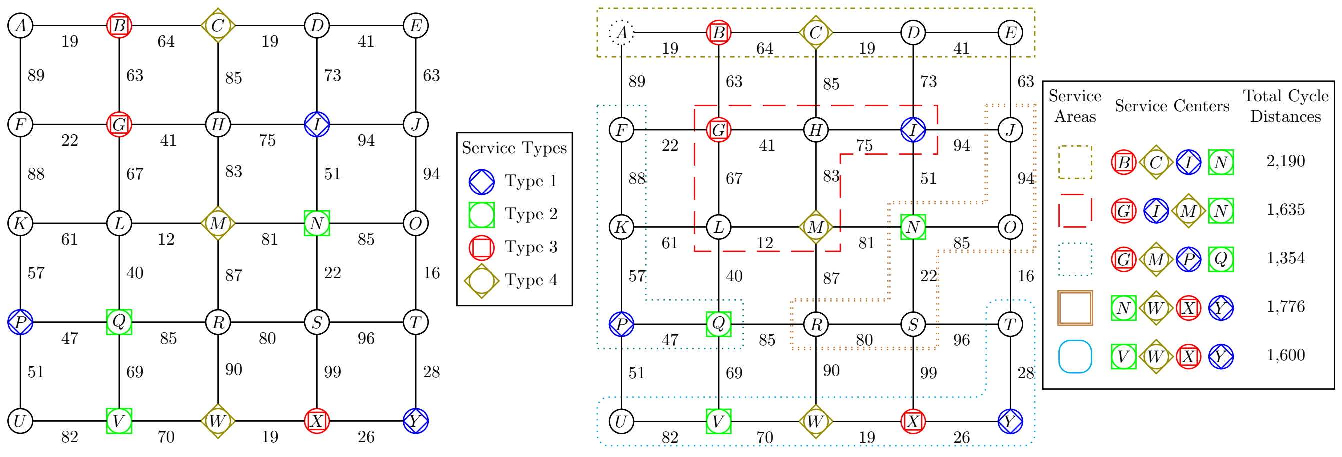
TABLE 1
Applications of MRNVD.

Applications	Benefit of MRNVD Service Areas
Emergency Resource Allocation	Develop an emergency plan to help citizens to minimize their travel times to obtain all required resources.
Store Choices	Provide an efficient route to save time and gas while shopping.
Tourist Site Selection	Recommend a tourist route that can visit attractions with different types.

The fast response is crucial to mitigating the impacts of a pandemic and improving the safety of the traveling public and emergency responders. Scalable algorithms for MRNVD can ensure the timely and accurate delivery of useful emergency information and response for disastrous events. Moreover, planners can recalculate MRNVD after unpredictable changes to the transportation network (e.g. an incident that makes a service center nonfunctional or a wildfire that hinders the accessibility through certain roads). A faster algorithm has an advantage over a slower one in adapting to such changes, and hence improves the emergency response.

The importance of transportation resilience grows as the frequency and magnitude of extreme events increases (e.g., hurricanes, wildfires, etc.) [1], [2], [3]. MRNVD can allocate multiple resources to enhance the resilience against

The authors are with the Department of Computer and Electrical Engineering and Computer Science, Florida Atlantic University, 777 Glades Road EE 403, Boca Raton, FL 33431. E-mail: aqutbuddin2017@fau.edu,yangk@fau.edu



(a) Input with four types of service centers.

(b) Output (Polygons show Service Areas).

Fig. 1. Example of Input and Output of MRNVD (Best in Colors).

disasters. These resource types could range from medical services (e.g. hospitals and pharmacies) to essential survival items (e.g. grocery stores and gas stations). Since MRNVDs enhance the efficiency of resource allocation in disaster situations, a fast algorithm that calculates MRNVD improves transportation resilience planning.

MRNVD is also useful for another societal application such as providing better store choices to reduce travel time for shoppers. It is most useful during seasonal sales (e.g. labor day and black Friday). Reduced travel time between shopping centers and retail stores translates to a higher probability of benefiting from offers and buying more items on sales. Tourists and travelers could also benefit from MRNVD if the available time for sightseeing is limited. They can utilize their limited time in exploring sites of different types (e.g. historic sites, museums, galleries, botanical gardens, zoos, ... etc.) in their area of interest. The shortest cycle-distance between the sites is more economic in terms of transportation cost. Furthermore, tourists will have a higher exposure to available activities in the visited place.

1.2 Problem Definition

In our formulation of the MRNVD problem, a transportation network is represented and analyzed as an undirected graph composed of graph-nodes and edges. Every graph-node represents a spatial location in geographic space (e.g., road intersections), which can be used as a proxy for locations of residents. Every edge between two graph-nodes represents a road segment and has a travel distance. Every service center has a resource type (e.g., water, food, gas, medicine, etc.). The $MRNVD(N, E, S, D)$ problem is defined as follows:

Input: A transportation network G with

- a set of graph-nodes N and a set of edges E,
- a set of service center locations with k different resource types $S \subset N$, and
- a set of nonnegative real distances of edges $D : E \rightarrow R_0^+$

Output: A Multiple Resource Network Voronoi Diagram (MRNVD)

Objective:

- Min-sum: Minimize the total cycle-distances of graph-nodes to their allotted k service centers with different types of resources.

Constraints:

- Service Area (SA) allotment must be k service centers with different types of resources.

Definition 1. ***Cycle-Distance:*** *Given a starting point and a set of k different service centers, the cycle-distance is the distance of the shortest route that visits k service centers and returns to the starting point.*

1.3 Problem Hardness

The NP-hardness of MRNVD follows from a well-known result about the NP-hardness of the traveling salesman problem.

Theorem 1. *The MRNVD problem is NP-hard.*

Proof. The NP-hardness of MRNVD can be proven by reduction from a well-known NP-hardness problem, the traveling salesman problem (TSP) [4], [5]. Given a starting point o and a set of service centers S, TSP finds the shortest cycle-distance of o. Let $A = (o, S)$ be an instance of TSP, where o is the starting point and S is a set of service centers. Let $B = (O, S)$ be an instance of the MRNVD problem, where O is a set of starting points and S is a set of service centers. Assume that every service center has a different type. Let $O = \{o\}$. Then the instance of TSP is a special case of MRNVD, where O is a set with a single element (i.e., o). Since A is constructed from B in polynomial-bounded time, the proof is complete. □

1.4 Our Contribution

Our previous work proposed the Distance bounded Pruning (DP) algorithm to produce the optimal solution of MRNVD, as reviewed in Section 2 [6]. DP consists of two main novel components: (a) Straight-Distance bounded Pruning (SDP) and (b) Triangle-Distance bounded Pruning. The core idea of DP is to prune the exponential search space of the optimal cycle-distances with the lower and upper bound constraint.

DP used the straight-distance bound and the triangle-distance bound to define the lower and upper bounds of the optimal cycle-distance. However, we found that the pruning efficiency of DP decreases as the number of service types increases. In this paper, we propose a novel Cycle-Distance bounded Pruning (CDP) and CDP with Neighboring Bounds (CDP-NB) algorithms to reduce the computational cost of the DP algorithm. In addition, we experimentally and theoretically evaluated all proposed algorithms.

In summary, our new contributions are as follows:

- We propose a novel Cycle-Distance bounded Pruning (CDP) algorithm to reduce the computational cost of MRNVD.
- We propose a more scalable solution that uses the allotments of neighboring nodes.
- We theoretically evaluate all proposed algorithms through a cost model and proofs of algorithm properties (e.g. termination, correctness).
- We experimentally evaluate all proposed algorithms using five different real-world road maps.

1.5 Related Work

Network Voronoi Diagram (NVD) is extensively used to identify the nearest service center [7], [8], [9], [10], [11]. Capacity Constrained Network Voronoi Diagram (CCNVD) honors the capacity constraint of each service center and creates contiguous service areas that can minimize the sum of the shortest distances of graph-nodes to their allotted service centers [12]. However, the application of NVD and CCNVD is limited to a single type of resource [13]. Consider the example of a resident who is looking for gas, water, and medicine at the same time. Both NVD and CCNVD cannot minimize the travel time to visit a series of service centers, one from each resource. Recently two-site network Voronoi diagrams were proposed to identify the best route for two different resources [14], [15]. The general idea is to find the minimum triangle-perimeter to partition the spatial network into a set of Service Areas. However, two-site network Voronoi diagrams cannot be generalized into MRNVD due to the hardness of the cycle-distance computation [16]. The Voronoi based k-nearest neighbor search for spatial network databases was proposed to identify k different nearest service centers [17], [18], [19], [20]. However, the Voronoi k-nearest neighbor cannot produce the minimum cycle-distance because it considers only the direct distance of the graph-node to service centers as shown in Fig. 2. Furthermore, none of these methods honor multiple resource types and guarantees the minimum cycle-distance for graph-nodes to allotted service centers. In this work, we propose a novel approach for creating the optimal MRNVD that can minimize the total cycle-distances of graph-nodes to their allotted k service centers with different types.

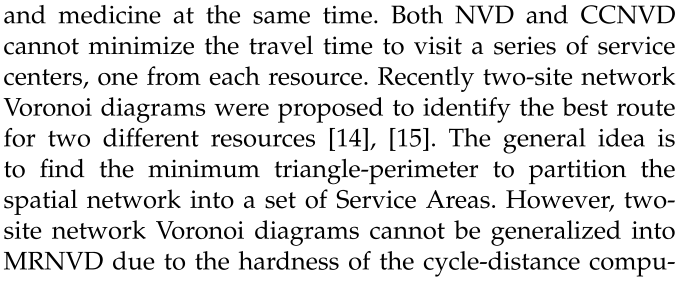


Fig. 2. Types of Network Voronoi Diagram.

The Multiple Resource Network Voronoi Diagram (MRNVD) problem is an extension of the Generalized Traveling Salesman Problem (GTSP). Given a weighted complete undirected graph G with a set of nodes N and a set of edges E, and a set of node clusters $N = N_1 \cup N_2 \dots N_k$, the objective of the GTSP problem is to find a minimum weight cycle containing exactly one node from each cluster [21]. Various approaches to GTSP have been studied in the literature. We can classify these approaches into three categories: (1) Graph Transformation methods, (2) Integer programming (IP) methods, and (3) Heuristic methods. The Graph Transformation methods convert the instance of GTSP to an equivalent instance of TSP and apply the traditional TSP solvers as black boxes to find the solution of GTSP [22], [23]. However, these transformation techniques increase the size of the input network and may yield the degeneracy of the TSP problem representation [24]. The GTSP problem can be formulated as an Integer Programming (IP) problem [23]. The Integer Programming (IP) methods utilize the Integer Programming (IP) solver with Lagrangian relaxation, branch-and-bound, or branch-and-cut techniques to find an optimal solution [23], [25], [26]. However, these methods require an exponential number of intermediate variables and constraints to shrink the feasible region and execute multiple IP solvers to obtain the feasible solutions [27]. This issue arises from a lack of a complexity analysis and comprehensive cost model for the IP solver [28]. The heuristic methods perform a predefined number of iterations of local improvement algorithms and produce the sub-optimal solution. Notable examples of heuristics include k-opt local search, genetic algorithm, and neighborhood search [29], [30], [31]. However, the heuristic methods mainly focus on the average empirical behavior of the local improvement algorithms, resulting in a lack of provable guarantees on the solution quality and optimality [16]. The MRNVD problem may be solved using the methods used for GTSP. However, these methods requires n number of IP instances for n graph-nodes, which is not scalable for a large-sized transportation network. In contrast with these methods, our methods utilizes the topological connectivity of the transportation network and minimizes the total cycle-distances of graph-nodes to their allotted service centers with different types of resources. The proposed approaches relies neither on the IP solvers nor the heuristic methods and finds the optimal solution for a large-sized transportation network.

1.6 Scope and Outline

In constructing our novel algorithms for MRNVD, we assume undirected edges. Additionally, in this work, the locations of service centers are known a priori and the goal is to create an MRNVD based on service center locations. Finding optimal locations for new or additional service centers (e.g. Facility Location Problems) [32], [33] is beyond the scope of the present research. Moreover, the capacity of service centers (e.g. CCNVD [12]) is not discussed in this research and might be addressed in future work.

The rest of the paper is organized as follows: Section 2.1 describes the Baseline approach for MRNVD. Section 2.2 explains the Distance bounded Pruning (DP) algorithm. Section 3 describes our proposed approaches. We provide correctness proofs of the proposed approaches in Section 4. Section 5 presents the experimental observations and results. Finally, Section 6 concludes the paper.

2 Baseline & Distance bounded Pruning

In this section, we first introduce our Baseline algorithm that can produce the optimal solution of MRNVD. We then describe the Distance bounded Pruning (DP) approach to reduce the computational cost for the MRNVD problem.

2.1 Baseline approach

In this subsection, we introduce the Baseline approach using the dynamic programming technique [34], [35]. The Baseline approach examines all graph-nodes $n \in N$ and finds the shortest cycle of graph-node n. It starts by (1) generating all service center combinations of k different service types, then (2) identifying a cycle that can produce the shortest cycle-distance of graph-node n, and (3) finally, assigning service centers on the shortest cycle to graph-node n. The core component of the Baseline approach is to construct a k-partite graph for service centers, materialize the shortest path distance on ordered k service centers, and reuse it for the cycle-distance computation.

Consider the example input of MRNVD in Fig. 1a. Let us identify the cycle-distance for graph-node F. First, the Baseline approach creates the k-partite graph for service centers and generates all service center combinations with k different types (see Fig 3).

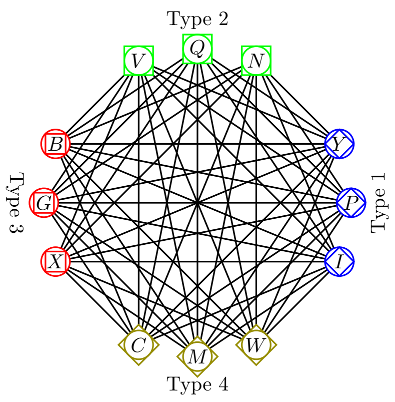


Fig. 3. k-partite graph for service center combinations (graph-node F)

Second, the Baseline approach uses the Held-Karp algorithm and computes the cycle-distance for graph-node F with every combination [35], [36]. In this example, the service center combination $\{G, M, Q, P\}$ can produce the shortest cycle with a distance of 345 for graph-node F (i.e., $F \to G \to M \to Q \to P \to F$). Therefore, the Baseline approach assigns service centers $\{G, M, Q, P\}$ to graph-node F. The path distance of $G \to M \to Q \to P$ can be materialized and reused for computing the cycle distance for other graph-nodes. The Baseline approach produces the optimal solution for MRNVD [6]. However, since the number of service center combinations is exponential in terms of the number of service centers and the number of resource types, the Baseline approach is challenging for a large-size transportation network [36].

2.2 Distance bounded Pruning Algorithm

The limitation of the Baseline approach is that the search space becomes exponential due to the large size of service center combinations. In this section, we describe the Distance bounded Pruning (DP) algorithm that can efficiently prune the search space [6]. The DP algorithm consists of two novel components: (a) Straight-Distance bounded Pruning (SDP) and (b) Triangle-Distance bounded Pruning (TDP). SDP identifies a subset of service centers that can produce the optimal cycle-distance. TDP can rule out the service center combinations that cannot produce the optimal solution. Both approaches can reduce the search space for finding the optimal cycle-distance without examining all possible combinations of service centers.

2.2.1 *Straight-Distance bounded Pruning*

In this subsection, we describe the Straight-Distance bounded Pruning (SDP) method that can prune the search space for service center combinations using a Set Window. A Set Window is defined as follows.

Definition 2. ***Set Window(S,n,t):*** *Given a set of service centers S, a graph-node n, and the number of service centers t, a Set Window (SW) is defined as a subset of service centers $SW \subseteq S$ that are the t nearest neighboring service centers from the given graph-node n.*

SDP finds the lower and upper bounds of the cycle-distance of a graph-node n and utilizes the two bounds to identify a subset of service centers that can produce the optimal cycle-distance of graph-node n. SDP expands the SW by changing the value of t and testing if the current SW meets the lower and upper bound constraint. If the SW violates the lower and upper bound constraint, SDP rules out service centers located outside of the SW. This approach can eliminate unnecessary service centers for finding the optimal cycle [6].

SDP follows three main steps: 1) constructing an initial SW, 2) expanding the SW until it violates the bound constraint, and 3) finding the optimal cycle-distance.

Definition 3. ***Initial Set Window:*** *Given a set of service centers S and a graph-node n, the Initial Set Window is defined as the minimum set of closest service centers to graph-node n that contain all types of service centers.*

SDP starts by creating an initial SW for each graph-node. Given a graph-node $n \in N$, SDP constructs an ordered-list of service centers based on the distance from graph-node n. Then, it identifies the minimum-sized SW that includes all types of service centers [6]. We refer to this as the initial SW. Since the initial SW contains k different service types, it produces the cycle-distance that is feasible but may not be optimal. Given a Set Window (SW), the lower-bound of the cycle-distance is obtained by doubling the distance of graph-node n to the farthest service center in the SW. The upper-bound of the cycle-distance is the minimum cycle-distance among all service center combinations in the SW [6]. SDP then expands the SW and updates the lower-bound of the optimal cycle-distance. If the lower-bound is greater than the upper-bound, SDP produces all possible combinations of service centers in the current SW. Finally, SDP identifies the minimum cycle-distance using the Held-Karp algorithm [35].

Property 1. ***Lower and Upper bounds Property:*** *As the size of a SW increases, the lower-bound of the SW monotonically increases and the upper-bound of the SW monotonically decreases.*

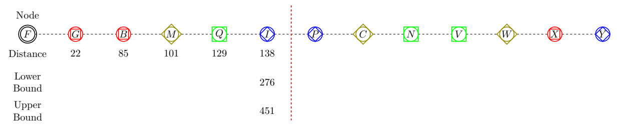


Fig. 4. Initial Set Window (SW) for graph-node F with bound constraints.

Consider again graph-node F in Fig. 1a. Fig. 4 shows an example of the initial SW for graph-node F. Given graph-node F, all service centers are ordered by the distance from graph-node F. The vertical bar splits the ordered-list into the left and right parts; The left part becomes the initial SW (i.e., $\{G, B, M, Q, I\}$) whose size is minimal and includes all types of service centers. SDP then computes the lower and upper bounds of the cycle-distance of graph-node F. Since service center I is the farthest service center from graph-node F in the SW, the lower-bound becomes 276. SDP examines all possible combinations of four different types and identifies the minimum cycle-distance among these combinations. In this example, SDP produces two service center combinations (i.e., $\{B, I, M, Q\}$ and $\{G, I, M, Q\}$) and computes the cycle-distance for each combination (see Table 2). The minimum cycle-distance is 451 and therefore it can serve as the upper-bound of the optimal cycle-distance.

TABLE 2
Cycle-Distances of graph-node F using the initial SW (The highlighted cell shows the minimum value).

Combinations	Shortest Cycle	Cycle-Distance
$\{B, I, M, Q\}$	$F \rightarrow B \rightarrow I \rightarrow M \rightarrow Q \rightarrow F$	554
$\{G, I, M, Q\}$	$F \rightarrow G \rightarrow I \rightarrow M \rightarrow Q \rightarrow F$	<u>451</u>

Since the upper-bound (i.e., 451) is greater than the lower-bound (i.e., 276), SDP can expand the SW until it violates the bound constraint (see Fig. 5). SDP then produces all possible service center combinations in the current SW and finds the optimal cycle-distance of graph-node F. In this example, the number of service center combinations is 24 ($2 \times 3 \times 2 \times 2 = 24$), and the optimal cycle-distance is 345 (i.e., $F \rightarrow G \rightarrow M \rightarrow Q \rightarrow P \rightarrow F$). Finally, SDP assigns service centers $\{G, M, Q, P\}$ to graph-node F for MRNVD.

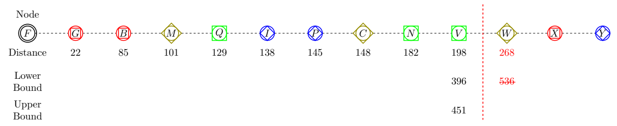


Fig. 5. Set Window (SW) for graph-node F with bound constraints.

2.2.2 Triangle-Distance bounded Pruning

In this subsection, we describe the Triangle-Distance bounded Pruning (TDP) method that can prune the search space for service center combinations using the max-min triangle-distance.

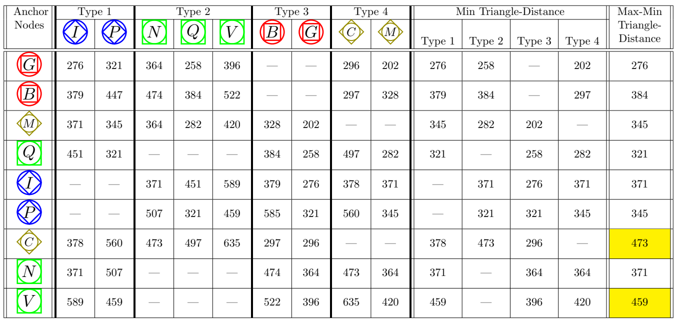
Definition 4. ***Triangle-Distance(n, s_a, s):*** *Given a starting graph-node n and two service centers s_a and s, the triangle-distance is defined as the cycle-distance of $n \rightarrow s_a \rightarrow s \rightarrow n$.*

Definition 5. ***Min Triangle-Distance(n, s_a, t):*** *Given a starting graph-node n, a service center s_a, and a service center type t, the min triangle-distance is defined as the minimum Triangle-Distance(n, s_a, s), where type(s_a) $\neq t$ and type(s) $= t$.*

Definition 6. ***Max-Min Triangle-Distance(n, s_a):*** *Given a starting graph-node n, a service center s_a, and a set of service center types T, the max-min triangle-distance is defined as the maximum of Min Triangle-Distance(n, s_a, $t \in T$).*

The core idea of TDP is that when Max-Min Triangle-Distance(n, s_a) is greater than the upper-bound of the optimal cycle-distance, the algorithm will not compute the cycle-distance of the service center combinations that include service center s_a [6]. We refer to s_a as the anchor-node.

The TDP method follows three main steps: 1) creating a triangle-distance table, 2) computing the max-min triangle-distance for every anchor-node, and 3) identifying anchor-nodes that violate the upper bound constraint and rule out the service center combinations that include these anchor-nodes.



Anchor Nodes	Type 1		Type 2			Type 3		Type 4		Min Triangle-Distance				Max-Min Triangle-Distance
	I	P	N	Q	V	B	G	C	M	Type 1	Type 2	Type 3	Type 4	
G	276	321	364	258	396	—	—	296	202	276	258	—	202	276
B	379	447	474	384	522	—	—	297	328	379	384	—	297	384
M	371	345	364	282	420	328	202	—	—	345	282	202	—	345
Q	451	321	—	—	—	384	258	497	282	321	—	258	282	321
I	—	—	371	451	589	379	276	378	371	—	371	276	371	371
P	—	—	507	321	459	585	321	560	345	—	321	321	345	345
C	378	560	473	497	635	297	296	—	—	378	473	296	—	473
N	371	507	—	—	—	474	364	473	364	371	—	364	364	371
V	589	459	—	—	—	522	396	635	420	459	—	396	420	459

Fig. 6. Triangle-Distance Table for graph-node F (The highlighted cells violate the upper-bound cycle-distance constraint).

Given a Set Window (SW), TDP starts by constructing a triangle-distance table for anchor-nodes (i.e., service centers) and computes the max-min triangle-distance for every anchor-node. Consider the Set Window (SW) in Fig. 5. First, TDP groups a set of service centers based on types and constructs the triangle-distance table for anchor-nodes (i.e., service centers G, B, M, Q, I, P, C, N, and V) (see Fig. 6). In this example, the group of type 1 is $\{I, P\}$, the group of type 2 is $\{N, Q, V\}$, the group of type 3 is $\{B, G\}$, and the group of type 4 is $\{C, M\}$. Next, TDP computes the min triangle-distance for every type of service centers. For instance, the min triangle-distance of the anchor-node B with Type 1 becomes 379. After that, TDP defines the max-min triangle-distance for all anchor-nodes.

TABLE 3
Cycle-Distances for Graph-node F using the Triangle Distance Table (The shaded rows show the new additions and the highlighted cell shows the minimum value).

Combinations	Shortest Cycle	Cycle-Distance
$\{B, I, N, M\}$	$F \rightarrow B \rightarrow I \rightarrow N \rightarrow M \rightarrow F$	474
$\{B, I, M, Q\}$	$F \rightarrow B \rightarrow I \rightarrow M \rightarrow Q \rightarrow F$	554
$\{B, N, M, P\}$	$F \rightarrow B \rightarrow N \rightarrow M \rightarrow P \rightarrow F$	617
$\{B, M, Q, P\}$	$F \rightarrow B \rightarrow M \rightarrow Q \rightarrow P \rightarrow F$	471
$\{G, M, N, I\}$	$F \rightarrow G \rightarrow M \rightarrow N \rightarrow I \rightarrow F$	371
$\{G, I, M, Q\}$	$F \rightarrow G \rightarrow I \rightarrow M \rightarrow Q \rightarrow F$	451
$\{G, M, N, P\}$	$F \rightarrow G \rightarrow M \rightarrow N \rightarrow P \rightarrow F$	507
$\{G, M, Q, P\}$	$F \rightarrow G \rightarrow M \rightarrow Q \rightarrow P \rightarrow F$	<u>345</u>

Note that the upper-bound of the optimal cycle-distance in SW is 451 (see Fig. 5). Since the max-min triangle-distances of anchor-nodes C and V are greater than the upper-bound of the optimal cycle-distance (i.e., 451), these anchor-nodes cannot be a part of the shortest cycle [6]. Therefore, TDP rules out the service center combinations containing service centers C or V because these combinations cannot produce the optimal cycle-distance. Finally, TDP computes the cycle-distance for the remaining service center combinations and finds the optimal cycle-distance of graph-node F. Table 3 shows the remaining service center combinations and the minimum cycle for these combinations. In this example, the number of service center combinations is 8. Since the service center combination $\{G, M, Q, P\}$ produces the minimum cycle-distance, TDP allots service centers $\{G, M, Q, P\}$ to graph-node F for MRNVD.

Algorithm 1 DP algorithm (Pseudo-code)

```
Inputs:
  - A transportation network $G(N, E)$ with graph-nodes $N$ and edges $E$.
  - A set of service center locations with $k$ different resource types $S \subset N$.
  - Every edge has a distance $d(e)$
Output: Multiple Resource Network Voronoi Diagram
Steps:
 1: Compute the distance matrix for graph-nodes in $G$.
 2: for graph-node $n \in N$ in $G(N, E)$ do
 3:     Construct the initial Set Window (SW) for $n$.
 4:     Compute the initial lower and upper bounds of the cycle-distance.
 5:     while the lower and upper bound constraint is not violated do
 6:         Increase the size of the SW by one.
 7:         Update the lower and upper bounds and prune search space.
 8:     end while
 9:     Identify the cycle-distance of $n$ and allot service centers to $n$.
10: end for
11: return MRNVD (i.e., allotment of graph-nodes to their service centers).
```

Algorithm 1 presents the pseudo-code for the Distance bounded Pruning (DP) algorithm. DP computes the distance matrix for graph-nodes in the transportation network G (Line 1). For each graph-node n, DP computes the cycle-distance of n (Line 2-10). First, it constructs the initial Set Window (SW) and computes the lower and upper bounds of the optimal cycle-distance (Line 3-4). Next, it changes the size of the SW and updates the lower-bound of the optimal cycle-distance (Line 5-8). When the SW violates the lower and upper bound constraint, DP rules out the non-optimal service center combinations using the TDP method (Line 7). Then DP finds the optimal cycle-distance of graph-node n and assigns service centers on the cycle to graph-node n (Line 9). This process continues until all graph-nodes are allotted (Line 2). Finally, MRNVD is returned (Line 11).

2.2.3 Limitations of Distance bounded Pruning

Even though the DP approach has a performance gain over the baseline approach, it has a limitation for two reasons. First, as the number of service centers increases, the size of the Triangle Distance Table becomes very large. Second, the effect of pruning on search space decreases as the number of service types increases. Therefore, the DP approach may be inapplicable for sizable road networks. Thus we propose more scalable algorithms that hierarchically construct the distance tables and efficiently reduce the search space to construct an MRNVD.

3 Proposed Approach

In this section, we introduce two novel approaches, namely (a) Cycle-Distance bounded Pruning (CDP) and (b) CDP with Neighboring Bounds (CDP-NB). CDP constructs and updates cycle-distance tables in a hierarchical fashion to reduce the computational cost of identifying the optimal cycle. CDP-NB uses a more strict upper bound based on neighboring allotments and further reduces the size of cycle distance tables.

3.1 Cycle-Distance bounded Pruning

The Cycle-Distance bounded Pruning (CDP) approach uses anchor-sets to construct hierarchical multi-level cycle-distance tables and efficiently rules out the number of service center combinations that cannot produce the optimal cycle-distance.

Definition 7. ***Anchor-Set(****S****,*** ***i*****):** *Given a set of service centers S and the size of anchor-set i, the anchor-set is a subset of S, where its cardinality is i and every element in the anchor-set has a unique type.*

The basic structure of the CDP algorithm involves major and minor iterations. The major iterations start with an initial SW and expand the SW until it violates the lower and upper bound constraint. Each major iteration updates the lower-bound of the current SW and the upper-bound of the next SW. The lower-bound of the cycle-distance is obtained by doubling the distance of graph-node n to the farthest service center in the SW. The upper-bound of the cycle-distance is the minimum cycle-distance in the SW. The algorithm terminates when the current SW violates the lower and upper bound constraint or the current SW includes all service centers.

Each major iteration requires minor iterations that construct multi-level cycle-distance tables based on the size of anchor-sets. If the max-min cycle-distance of an anchor-set does not meet the upper-bound constraint, CDP ignores all the supersets of the anchor-set when it creates or updates the next-level cycle-distance tables.

Definition 8. ***Cycle-Distance(****n****,*** ***S_A*****,** ***s*****):** *Given a starting graph-node n, an Anchor-Set S_A, and a service center $s \notin S_A$, the cycle-distance(n, S_A, s) is defined as the cycle-distance of graph-node n that visits every service center in $\{S_A \cup s\}$ and returns to the starting graph-node n.*

Definition 9. ***Min Cycle-Distance(****n****,*** ***S_A*****,** ***t*****):** *Given a starting graph-node n, an Anchor-Set S_A, and a service center type t, the min cycle-distance(n, S_A, t) is defined as the minimum cycle-distance(n, S_A, $s \notin S_A$), where $type(s) = t$ and $type(s_A \in S_A) \neq t$.*

Definition 10. ***Max-Min Cycle-Distance(****n****,*** ***S_A*****):** *Given a starting graph-node n, an Anchor-Set S_A, and a set of service center types T, the max-min cycle-distance is defined as the maximum of min cycle-distance(n, S_A, $t \in T$).*

Given an SW, CDP increases the size of an anchor-set (i.e., i), creates or updates cycle-distance tables in a hierarchical fashion, and rules out anchor-sets and service centers that violate the upper-bound constraint. The core idea of CDP is that it prunes the search space in two ways: 1) Vertical pruning and 2) Horizontal pruning. The Vertical pruning rules out service center combinations that include anchor-sets whose Max-Min Cycle-Distance is greater than the upper-bound of the optimal cycle-distance. The Horizontal pruning constructs a k-partite graph to eliminate unnecessary service centers for creating the next level cycle distance table.

Property 2. ***Monotone Property:*** *For all $n \in N$, if $S_{A_0} \subset S_{A_1}$, then Max-Min Cycle-Distance(n, S_{A_0}) $\leq$ Max-Min Cycle-Distance(n, S_{A_1}) (see Lemma 3).*

Property 3. ***Max-Min Cycle-Distance Principal:*** *If an anchor-set S_A cannot be a part of the optimal cycle of graph-node n in a SW, then the supersets of S_A cannot be a part of the optimal cycle in the SW.*

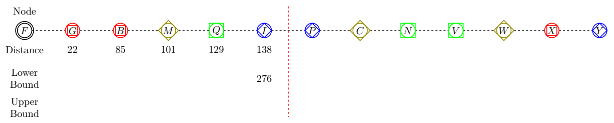


Fig. 7. Initial Set Window (SW) for graph-node F: iteration 1.

We illustrate the algorithm using the initial Set Window (SW) shown in Fig. 7 (reproduced from Fig. 4). The lower-bound of the optimal cycle-distance in the initial SW is 276. Since the number of service center types is 4, CDP changes the value of i from 1 to 3 and creates multi-level cycle-distance tables with $Anchor\text{-}Set(SW, i)$. Figs. 8–10 show the minor iterations that construct multi-level cycle-distance tables with the initial SW.

Anchor Sets	Type 1	Type 2	Type 3		Type 4	Min Cycle-Distance				Max-Min Cycle-Distance
	I	Q	B	G	M	Type 1	Type 2	Type 3	Type 4	
G	276	258	—	—	202	276	258	—	202	276
B	379	384	—	—	328	379	384	—	328	384
M	371	282	328	202	—	371	282	202	—	371
Q	451	—	384	258	282	451	—	258	282	451
I	—	451	379	276	371	—	451	276	371	451

Fig. 8. 1st-level Cycle-Distance Table using the initial SW.

Anchor Sets	Type 1	Type 2	Type 3		Type 4	Min Cycle-Distance				Max-Min Cycle-Distance
	I	Q	B	G	M	Type 1	Type 2	Type 3	Type 4	
G M	371	282	—	—	—	371	282	—	—	371
G Q	451	—	—	—	282	451	—	—	282	451
G I	—	451	—	—	371	—	451	—	371	451
B M	474	408	—	—	—	474	408	—	—	474
B Q	554	—	—	—	408	554	—	—	408	554
B I	—	554	—	—	474	—	554	—	474	554
M Q	451	—	408	282	—	451	—	282	—	451
M I	—	451	474	371	—	—	451	371	—	451
Q I	—	—	554	451	451	—	—	451	451	451

Fig. 9. 2nd-level Cycle-Distance Table using the initial SW.

Anchor Sets	Type 1	Type 2	Type 3		Type 4	Min Cycle-Distance				Max-Min Cycle-Distance
	I	Q	B	G	M	Type 1	Type 2	Type 3	Type 4	
G M Q	451	—	—	—	—	451	—	—	—	451
G M I	—	451	—	—	—	—	451	—	—	451
G Q I	—	—	—	—	451	—	—	—	451	451
B M Q	554	—	—	—	—	554	—	—	—	554
B M I	—	554	—	—	—	—	554	—	—	554
B Q I	—	—	—	—	554	—	—	—	554	554
M Q I	—	—	554	451	—	—	—	451	—	451

Fig. 10. 3rd-level Cycle-Distance Table using the initial SW.

Fig. 10 shows that the minimum max-min cycle-distance with $Anchor\text{-}Set(SW, 3)$ is 451, and it can serve as the upper-bound of the optimal cycle-distance. CDP then adds service center P to the SW, sets the lower-bound to 290 for the next SW, and performs the next major iteration.

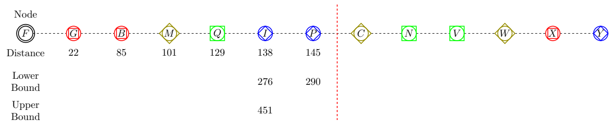


Fig. 11. Set Window (SW) for graph-node F : iteration 2.

In the second major iteration, CDP updates the multi-level cycle-distance tables for service center P. Let a be the new addition of a service center to the SW and let $\tilde{s_A}$ be an anchor-set. The update rule is following.

1) If the anchor set $\tilde{s_A}$ has no existing max-min cycle-distance, then CDP computes all possible min cycle-distances and adds the max-min cycle-distance for $\tilde{s_A}$ to the cycle-distance table.
2) If the anchor set $\tilde{s_A}$ produces a cycle-distance less than the already found max-min cycle-distance, then CDP updates min cycle-distances and the max-min cycle-distance for $\tilde{s_A}$.

Fig. 12 shows the updated 1st-level cycle-distance table after the addition of service center P. Gray-shaded fields represent the updated values. Since all anchor-sets meet the upper-bound constraint, CDP generates new anchor-sets with P and updates the 2nd-level cycle-distance table.

Anchor Sets	Type 1		Type 2	Type 3		Type 4	Min Cycle-Distance				Max-Min Cycle-Distance
	I	P	Q	B	G	M	Type 1	Type 2	Type 3	Type 4	
G	276	321	258	—	—	202	276	258	—	202	276
B	379	447	384	—	—	328	379	384	—	328	384
M	371	345	282	328	202	—	345	282	202	—	345
Q	451	321	—	384	258	282	321	—	258	282	321
I	—	—	451	379	276	371	—	451	276	371	451
P	—	—	321	447	321	345	—	321	321	345	345

Fig. 12. 1st-level Cycle-Distance Table after the addition of P.

Fig. 13 shows the updated 2nd-level cycle-distance table. At this level, CDP utilizes the special structure of a k-partite graph and applies the Horizontal pruning to remove the service centers that cannot be a part of the optimal cycle-distance. The creation of the k-partite graph takes the following steps (Lemma 1).

1) Creates a node for every service center and groups them based on their type.
2) Creates an edge between two different type service centers (i.e., s_a and s_b) when the minimum Max-Min Cycle-Distance(n, S_A), for all $\{s_a, s_b\} \subseteq S_A$, is less than the upper bound.
3) Removes a node that cannot connect to $k-1$ different type nodes and removes the edges of the node.

Consider the example 4-partite graph produced from the Cycle-Distance Table in Fig. 13. Every node represents a service center, and every edge connects two different type service centers. Every edge (i.e., $\overline{s_a s_b}$) has a value that represents the minimum Max-Min Cycle-Distance(n, S_A), $\forall\, S_A$ such that $\{s_a, s_b\} \subseteq S_A$. In this example, we do not create an edge for anchor-sets $\{B, I\}$,$\{B, M\}$, and $\{B, P\}$ because these anchor-sets violate the upper-bound constraint (see

Anchor Sets	Type 1		Type 2	Type 3		Type 4	Min Cycle-Distance				Max-Min Cycle-Distance
	I	P	Q	B	G	M	Type 1	Type 2	Type 3	Type 4	
G M	371	345	282	—	—	—	345	282	—	—	345
G Q	451	321	—	—	—	282	321	—	—	282	321
G I	—	—	451	—	—	371	—	451	—	371	451
G P	—	—	321	—	—	345	—	321	—	345	345
B M	474	471	408	—	—	—	471	408	—	—	471
B Q	554	447	—	—	—	408	447	—	—	408	447
B I	—	—	554	—	—	474	—	554	—	474	554
B P	—	—	447	—	—	471	—	447	—	471	471
M Q	451	345	—	408	282	—	345	—	282	—	345
M I	—	—	451	474	371	—	—	451	371	—	451
M P	—	—	345	471	345	—	—	345	345	—	345
Q I	—	—	—	554	451	451	—	—	451	451	451
Q P	—	—	—	447	321	345	—	—	321	345	345

Fig. 13. 2nd-level Cycle-Distance Table after the addition of P.

Fig. 14). Since service center B is not neighboring to three different type nodes, CDP ignores anchor-sets that contain B for the next-level cycle-distance table. (see Lemma 2).

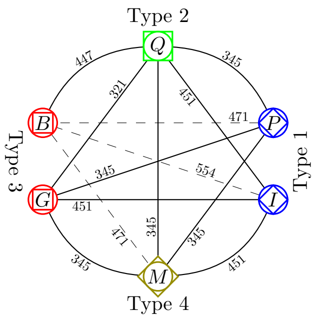


Fig. 14. k-partite graph for the Horizontal pruning.

Fig. 15 shows the updated 3rd-level cycle-distance table. The minimum max-min cycle-distance is 345, and therefore it serves as the upper-bound for the next major iteration.

Anchor Sets	Type 1		Type 2	Type 3	Type 4	Min Cycle-Distance				Max-Min Cycle-Distance
	I	P	Q	G	M	Type 1	Type 2	Type 3	Type 4	
G M Q	451	345	—	—	—	345	—	—	—	345
G M I	—	—	451	—	—	—	451	—	—	451
G M P	—	—	345	—	—	—	345	—	—	345
G Q I	—	—	—	—	451	—	—	—	451	451
G Q P	—	—	—	—	345	—	—	—	345	345
M Q I	—	—	—	451	—	—	—	451	—	451
M Q P	—	—	—	345	—	—	—	345	—	345

Fig. 15. 3rd-level Cycle-Distance Table after the addition of P.

In the third major iteration, CDP adds service center C to the SW and updates the lower and upper bounds for the optimal cycle-distance (see Fig. 16). Since the SW does not violate the lower and upper bound constraint, CDP updates the cycle-distance tables for the addition of service center C.

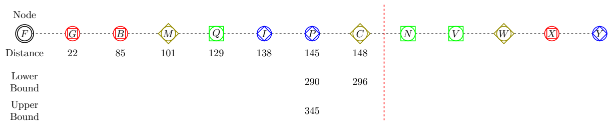


Fig. 16. Set Window (SW) for graph-node F: iteration 3.

Fig. 17 shows the update of the 1st-level cycle-distance table after the addition of service center C. It is important to note that service center C violates the upper bound constraint. Therefore, we do not need to update the next-level cycle-distance tables through the rest of the minor iterations.

Anchor Sets	Type 1		Type 2	Type 3		Type 4		Min Cycle-Distance				Max-Min Cycle-Distance
	I	P	Q	B	G	C	M	Type 1	Type 2	Type 3	Type 4	
G	276	321	258	—	—	296	202	276	258	—	202	276
B	379	447	384	—	—	297	328	379	384	—	297	384
M	371	345	282	328	202	—	—	345	282	202	—	345
Q	451	321	—	384	258	497	282	321	—	258	282	321
I	—	—	451	379	276	378	371	—	451	276	371	451
P	—	—	321	447	321	560	345	—	321	321	345	345
C	378	560	497	297	296	—	—	378	497	296	—	497

Fig. 17. 3rd-level Cycle-Distance Table after the addition of C.

Lastly, CDP adds service center N into the SW, but this addition violates the lower and upper bound constraint (see Fig. 18). Therefore, CDP assigns service centers $\{G, M, Q, P\}$ to graph-node F and terminates the major iteration.

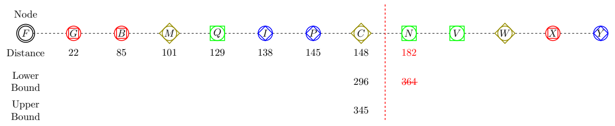


Fig. 18. Set Window (SW) for the graph-node F: iteration 4.

Algorithm 2 presents the pseudo-code for the Cycle-Distance bounded Pruning (CDP) algorithm. CDP computes the distance matrix for graph-nodes in the transportation network G (Line 1). For each graph-node n, CDP computes the cycle-distance of n (Line 2-11). First, it constructs the initial Set Window (SW) and initial cycle-distance tables and computes the lower and upper bounds of the optimal cycle-distance (Line 3-4). Next, it changes the size of the SW and updates the lower-bound of the optimal cycle-distance as well as cycle-distance tables (Line 6-7). It also updates the upper bound of the cycle-distance from the cycle-distance table with $Anchor\text{-}Set(SW, k-1)$ (Line 8). CDP uses the lower and upper bound constraint and the k-partite graph to rule out the non-optimal combinations for the cycle-distance computation. CDP then finds the optimal cycle-distance of graph-node n and assigns service centers on the cycle to graph-node n (Line 10). This process continues until all graph-nodes are allotted (Line 2). Finally, MRNVD is returned (Line 12).

3.2 CDP with Neighboring Bounds

In this section, we introduce the CDP with Neighboring Bounds (CDP-NB) approach that utilizes the allotments of neighboring nodes to find a more strict upper bound for the cycle distance computation. The upper-bound of the cycle-distance of n can be defined as:

$$UB(n) = \underset{a \in nbr(n)}{arg\,min} \quad 2 \cdot dist(n,a) + cycle\_dist(a), \tag{1}$$

where $nbr(n)$ is neighboring nodes of n, $dist(n,a)$ is the shortest distance between n and a, and $cycle\_dist(a)$ is the cycle-distance of a.

Consider graph-node F in Fig.1a. Graph-node F has three neighbors (i.e., A, G and K). Assume that we know

Algorithm 2 CDP algorithm (Pseudo-code)

Inputs:
- A transportation network $G(N, E)$ with graph-nodes N and edges E.
- A set of service center locations with k different resource types $S \subset N$.
- Every edge has a distance $d(e)$

Output: Multiple Resource Network Voronoi Diagram

Steps:

```
1:  Compute the distance matrix for graph-nodes in $G$.
2:  for graph-node $n \in N$ in $G(N, E)$ do
3:      Construct the initial SW and cycle-distance tables for $n$.
4:      Compute the initial lower and upper bounds of the cycle-distance.
5:      while the bound constraints are not violated do
6:          Increase the size of the SW by one and update the lower-bound.
7:          Update the cycle-distance tables and prune search space.
8:          Update the upper-bound.
9:      end while
10:     Identify the cycle-distance of $n$ and allot service centers to $n$.
11: end for
12: return MRNVD (i.e., allotment of graph-nodes to their service centers).
```

the cycle distances of the neighboring nodes of F. In this example, the cycle-distances of A, G, and K are 452, 327, and 345, respectively. According to Equation 1, the upper-bound is $min(630, 371, 521) = 371$.

Note that a more strict upper bound can reduce the size of multi-level cycle-distance tables. Consider the 1st-level cycle-distance table in Fig 8. Since only nodes G and M meet the upper-bound constraint (i.e., 371), the cycle-distance table cannot include four different type service centers. Therefore, we can skip the remaining minor iterations and move to the next major iteration. Consider again the 1st-level cycle-distance table after the addition of service center P to the SW (see Fig 12). Since service centers B and I violate the upper-bound constraint, we can exclude B and I to construct the 2nd-level cycle-distance table, as shown in Fig. 19.

Anchor Sets	Type 1	Type 2	Type 3	Type 4	Min Cycle-Distance				Max-Min Cycle-Distance
	P	Q	G	M	Type 1	Type 2	Type 3	Type 4	
G M	345	282	—	—	345	282	—	—	345
G Q	321	—	—	282	321	—	—	282	321
G P	—	321	—	345	—	321	—	345	345
M Q	345	—	282	—	345	—	282	—	345
M P	—	345	345	—	—	345	345	—	345
Q P	—	—	321	345	—	—	321	345	345

Fig. 19. 2nd-level Cycle-Distance Table after the addition of P.

Fig. 19 shows that no anchor-sets violate the upper bound constraint. CDP-NB then generates the anchor-sets with size of 3 and constructs the 3rd-level cycle-distance table (see Fig. 20). In this example, we can clearly see that CDP-NB reduces the size of the multi-level cycle distance tables.

Anchor Sets	Type 1	Type 2	Type 3	Type 4	Min Cycle-Distance				Max-Min Cycle-Distance
	P	Q	G	M	Type 1	Type 2	Type 3	Type 4	
G M Q	345	—	—	—	345	—	—	—	345
G M P	—	345	—	—	—	345	—	—	345
G Q P	—	—	—	345	—	—	—	345	345
M Q P	—	—	345	—	—	—	345	—	345

Fig. 20. 3rd-level Cycle-Distance Table after the addition of P.

4 Analysis on the Quality of the Proposed Approaches to MRNVD

In this section, we prove that the proposed approaches are correct, i.e., the algorithms create the optimal MRNVD. We also provide the algebraic cost model of the proposed approaches.

4.1 Analysis of CDP Approach to MRNVD

The following lemmas prove the correctness of the Cycle-Distance bounded Pruning algorithm.

Lemma 1. *If service center s is not connected to at least $k-1$ service centers, each from a different type, in the k-partite graph derived from the cycle-distance table with $Anchor\text{-}Set(SW, 2)$, it cannot be a part of the optimal cycle.*

Proof. We prove this lemma by contradiction. Assume that service center s is not connected to any center of type x in the k-partite graph and can be a part of the optimal cycle. The max-min cycle-distance of any anchor-set should be less than or equal to the optimal cycle according to the triangle inequality theorem. The service center combination that includes center s should include a service center of type x as well. If service center s is not connected to any center of type x in the k-partite graph, we need to include a deleted edge to connect service center s to a center of type x. Since the weight of the deleted edge is greater than the upper-bound of the cycle-distance, this contradicts the assumption. □

Lemma 2. *If service center s is not connected to at least $k-1$ service centers in the k-partite graph derived from a cycle-distance table with $Anchor\text{-}Set(SW, i > 2)$, it cannot be a part of the optimal cycle.*

Proof. We prove this lemma by contradiction. Let s be a service center in Set-Window SW. The k-partite graph is composed of nodes and weighted edges. The nodes are service centers in SW. An edge between two service centers s_1 and s_2 has the weight of the minimum of max-min cycle-distance for $Anchor\text{-}Set(SW, i) = S_A$, such that $\{s_1, s_2\} \subset S_A$.

Assume at all the edges connecting center s to all other centers of type x are removed because the weights of these edges are greater than the upper-bound cycle-distance. Therefore, we have to include deleted edges to produce service center combinations that have service center s and a service center of type x. Since the weight of the deleted edge is greater than the upper-bound of the cycle-distance, this contradicts the assumption. □

Consider, for example, the 4-partite graph in Fig. 21. The graph is derived from the cycle-distance table in Fig. 15. Two anchor-sets are supersets of $\{G, Q\}$, which are $\{G, Q, I\}$ and $\{G, Q, P\}$. The max-min cycle-distance for both Anchor-Sets are 451 and 345, respectively. Therefore, the edge weight between G and Q is 345. Furthermore, all edges incident to service center I have a cost greater than the upper bound of the next major iteration (i.e., 345). Therefore, we are sure that service center I cannot be part of the optimal cycle distance in the SW shown in Fig. 16 with the updated upper bound.

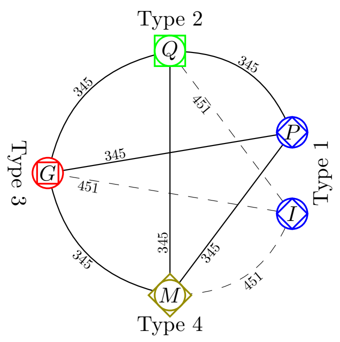


Fig. 21. k-partite graph from Cycle-Distance Table in Fig. 15

Lemma 3. *If $S_{A_0} \subset S_{A_1}$, then Max-Min Cycle-Distance(n, S_{A_0}) $\leq$ Max-Min Cycle-Distance(n, S_{A_1}).*

Proof. Let $S_{A_1 \setminus A_0}$ be $S_{A_1} \setminus S_{A_0}$. Assume $s_n \in S_{A_1 \setminus A_0}$. Let S be $S_{A_0} \cup \{s_n\}$. Then $Min\,Cycle\text{-}Distance(n, S, t)$ for every type $t \in T$ should be greater or equal to $Min\,Cycle\text{-}Distance(n, S_{A_0}, t)$. Therefore, Max-Min Cycle-Distance(n, S_{A_0}) $\leq$ Max-Min Cycle-Distance(n, S_{A_n}). Since the max-min cycle-distance monotonically increases by adding $s_n \in S_{A_1 \setminus A_0}$, we complete the proof by induction. □

Lemma 4. *The CDP approach to the MRNVD problem creates the optimal solution.*

Proof. CDP starts with the initial SW that can produce the feasible solution. CDP then expands the SW and updates the lower and upper bounds for the SW. CDP removes anchor-sets and service centers that cannot produce the optimal cycle-distance (Lemma 2). Since CDP terminates when the lower and upper bound constraint is violated or all service centers are included in SW, it produces the optimal solution. □

4.2 Algebraic Cost Model of Proposed Approaches to MRNVD

The goal of this subsection is to present cost models for our proposed approaches. Let n be the number of graph-nodes, let k be the number of types in service centers, let c be the maximum number of service centers for a service type.

4.2.1 Baseline Approach

The Baseline approach starts by generating all possible combinations of k different service centers. This takes $O(c^k)$. Given a combination, the cost of computation for the cycle-distance is $2^k \cdot k^2$ [16]. Since the number of combinations is bounded by $O(c^k)$, the minimum cycle-distance for a graph-node can be obtained by the cost of $O(c^k \cdot 2^k \cdot k^2)$. The number of graph-nodes is n. Therefore, the Baseline approach takes $O(c^k \cdot 2^k \cdot k^2 \cdot n)$.

4.2.2 Distance Bounded Pruning (DP) Approach

The Distance bounded Pruning (DP) approach starts by sorting service centers based on the distance from the starting graph-node. This takes $O(c \cdot k \cdot \log(c \cdot k))$ [37]. Next, SDP creates an initial Set Window (SW) and expands the SW. The size of the SW is bounded by $O(c \cdot k)$. During this incremental process, the cost for computing the lower-bounds is $O(c \cdot k)$ and the cost for computing the upper-bound is $O(c^k \cdot 2^k \cdot k^2)$ [16]. TDP creates a triangle-distance table to compute the max-min triangle-distances. This takes $O(c^2 \cdot k^2)$. Thus, the total cost of the allotment for each graph-node is $O(c{\cdot}k + c^2{\cdot}k^2 + c^k{\cdot}2^k{\cdot}k^2) \approx O(c^k{\cdot}2^k{\cdot}k^2)$. Since the number of graph-nodes is n, DP takes $O(c^k \cdot 2^k \cdot k^2 \cdot n)$. In the worst case, the cost model of DP is the same as the cost model of the Baseline approach. However, DP can rule out the non-optimal combinations and reduce the number of computations of the cycle-distances using SDP and TDP.

4.2.3 Cycle-Distance bounded Pruning (CDP) Approach

The Cycle-Distance bounded Pruning (CDP) approach starts by sorting service centers based on the distance from the starting graph-node. This takes $O(c \cdot k \cdot \log(c \cdot k))$ [37]. Next, it creates an initial Set Window (SW) and initial cycle-distance tables. CDP expands the SW and updates the cycle-distance tables. During this incremental process, the size of the SW is bounded by $O(c \cdot k)$. The cost for computing the lower-bound for the SW is $O(c \cdot k)$. The size of the cycle-distance tables is bounded by $O(c^{2k})$ if $c > 1$. The cost for updating a cycle-distance table entry is bounded by $O(2^k \cdot k^2)$ [16]. The total cost of the allotment for each graph-node is $O(c{\cdot}k + c^{2k}{\cdot}2^k{\cdot}k^2) = O(c^k{\cdot}2^k{\cdot}k^2)$. Since the number of graph-nodes is n, CDP takes $O(c^k \cdot 2^k \cdot k^2 \cdot n)$. In the worst case, the cost model of CDP is the same as the cost model of our preliminary work [6]. However, CDP can rule out the non-optimal anchor-sets and service center combinations, and significantly reduce the number of computations of the cycle-distances using cycle-distance tables and k-partite graphs.

4.2.4 CDP with Neighboring Bounds (CDP-NB) Approach

The main difference with CDP is to use the cycle-distance of neighboring nodes and apply a more strict bound for computing the cycle-distance. Therefore, the cost model of CDP-NB is the same as the cost model of CDP.

TABLE 4
Algebraic Comparison of Computational Cost

Algorithm	Computational Cost
Baseline	$O(c^k \cdot 2^k \cdot k^2 \cdot n)$
DP	$O(c^k \cdot 2^k \cdot k^2 \cdot n)$
CDP	$O(c^k \cdot 2^k \cdot k^2 \cdot n)$
CDP-NB	$O(c^k \cdot 2^k \cdot k^2 \cdot n)$

4.3 Storage Model for MRNVD

Trip and route planning information is a crucial component for resource allocation. In our study, we use the dictionary-based compression technique to store and provide the personalized route to individual drivers. MRNVD partitions the spatial network into a set of Service Areas that share the same service centers. MRNVD models the date hierarchy that plays a key role in processing and monitoring time-critical information. We can organize the clustered index based on Service Areas to fast access to individual route information in data pages. A data page stores the detailed routing information including intersections, roadway, traffic signal, etc. This hierarchy allows us to analyze and simplify huge volumes of traffic data for emergency preparedness.

5 Experimental Evaluation

We conducted experiments to evaluate the performances of Cycle-Distance bounded Pruning (CDP) and CDP with Neighboring Bounds (CDP-NB) approaches and compare them to the Distance bounded Pruning (DP) from our preliminary work [6]. The overall goal was to show the performance improvements to create an MRNVD that can be obtained by the CDP and CDP-NB approaches. We wanted to answer four questions: (1) What is the effect of the number of service types? (2) What is the effect of the number of service centers? (3) What is the effect of the size of the network (i.e., number of graph-nodes)? (4) Are CDP and CDP-NB algorithms correct, and is the solution quality preserved?

5.1 Experiment Layout

Fig. 22 shows our experimental setup. We chose five different municipal areas in the U.S. from OpenStreetMap [38] (as shown in Table 5). We retrieved the real locations of service centers and created a Multiple Resource Network Voronoi Diagram (MRNVD). We tested three approaches: (1) Distance bounded Pruning (DP), (2) Cycle-Distance bounded Pruning (CDP), and (3) CDP with Neighboring Bounds (CDP-NB) approaches.

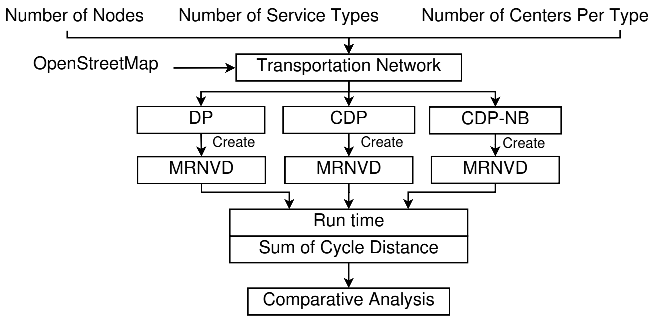


Fig. 22. Experiment Layout

TABLE 5
Transportation Network Datasets (Source: OpenStreetMap [38])

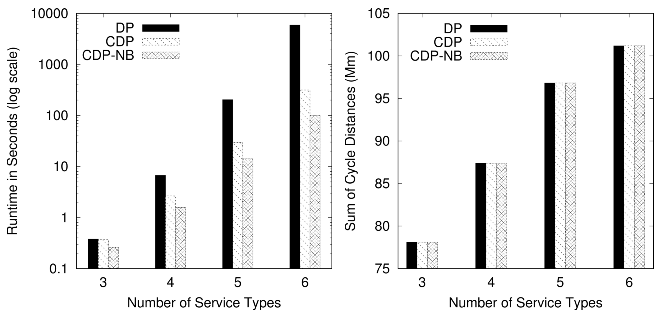
Area	No. of Graph-nodes ($\|N\|$)	No. of Edges ($\|E\|$)
Fort Lauderdale, FL	30,668	84,598
Miami, FL	40,484	114,822
Cape Cod, MA	42,104	107,566
Boston, MA	44,298	125,040
New Orleans, LA	55,107	166,184

5.2 Experiment Results and Analysis

We experimentally evaluated the proposed algorithms by comparing the impact on the performance of (1) the number of service types, (2) the number of centers per service type, and (3) the size of the transportation network. We used Dijkstra's algorithm to compute the shortest distance between nodes [37]. We extracted the locations of service centers from OpenStreetMap datasets and then chose a set of service centers from extracted ones to vary the number of service centers. The algorithms were implemented in Java 1.8 with a 32 GB memory run-time environment. All experiments were performed on an Intel Core i5 machine running Linux with 32 GB of RAM.

5.2.1 *Effect of Number of Service Types*

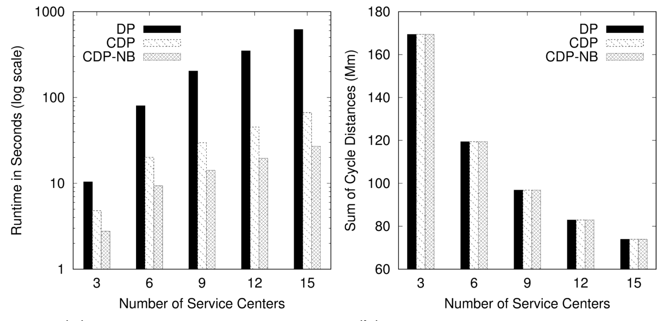
The first set of experiments evaluated the effect of the number of service types on the performance of the algorithms. We fixed the number of graph-nodes to $10,000$ and the number of service centers to 9. The average number of edges is $28,654$. We varied the number of service types from 3 to 6. We chose the locations of service centers (e.g., gas stations, grocery stores, shelters, hospitals, etc.) from OpenStreetMap and constructed 20 test cases. Performance measurements were execution time and the sum of the cycle-distances. The performance measurements were averaged over 100 test runs. Fig. 23a gives the execution times. As can be seen, the CDP approach outperforms the DP approach. This is because the number of combinations for the cycle-distance computation increases as the number of service types increases. CDP-NB outperforms other approaches because the upper-bound based on allotments of neighboring nodes can reduce the size of multi-level cycle-distance tables. When comparing the sum of the cycle-distances, we see that both CDP and CDP-NB approaches produce the optimal solution (see Fig. 23b). This means that our proposed approaches do not affect the solution quality. As the number of service types increases, the sum of the cycle-distances increases.



(a) Run Time Comparison (b) Comparison of Solution Quality
Fig. 23. Effect of number of service types ($n = 10,000$, $c = 9$)

5.2.2 *Effect of Number of Service Centers*

The second set of experiments evaluated the effect of the number of service centers on the performance of the algorithms. We fixed the number of graph-nodes to $10,000$ and the number of service types to 5. The average number of edges is $28,654$. We varied the number of centers per service type from 3 to 15. Locations of service centers were chosen from OpenStreetMap to construct 25 test cases. Performance measurements were execution time and the sum of the cycle-distances. The performance measurements were averaged over 125 test runs. Fig. 25a gives the execution times. As can be seen, CDP approach significantly outperforms the DP approach. This is because the number of combinations for the cycle-distance computation increases as the number of service centers increases. CDP-NB approach outperforms other approaches because it can reduce the size of multi-level cycle-distance tables. Fig. 24b shows that both CDP and CDP-NB approaches perform the same as the DP approach, and do not affect the solution quality. As the number of service centers increases, the sum of cycle-distances decreases.

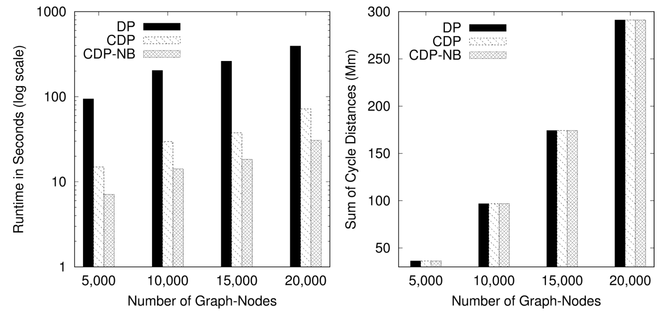


(a) Run Time Comparison (b) Comparison of Solution Quality

Fig. 24. Effect of number of service centers ($n = 10,000$, $k = 5$).

5.2.3 Effect of Network Size

The third set of experiments evaluated the effect of the network size on algorithm performance. We fixed the number of service types to 5 and the number of centers per service type to 9. We varied the number of nodes from $5,000$ to $20,000$ (i.e., $5,000$, $10,000$, $15,000$ and $20,000$). The average number of edges are $14,291$, $28,654$, $43,141$, and $57,468$, respectively. Service center locations were chosen from OpenStreetMap to construct 20 test runs for each road network. Performance measurements were execution time and the sum of the cycle-distances. The performance measurements were averaged over 100 test runs. Fig. 25a gives the execution times. We can see that the CDP approach significantly outperforms the DP approach. This is because CDP prunes the search space for service center combinations more efficiently than DP. CDP-NB outperforms other approaches because it can utilize the allotment of neighboring nodes to reduce the size of cycle-distance tables. Fig. 25b shows that the performances of CDP and CDP-NB are identical to the performance of DP. This means that CDP and CDP-NB do not affect the solution quality. As the number of graph-nodes increases, the sum of cycle-distances increases.



(a) Run Time Comparison (b) Comparison of Solution Quality

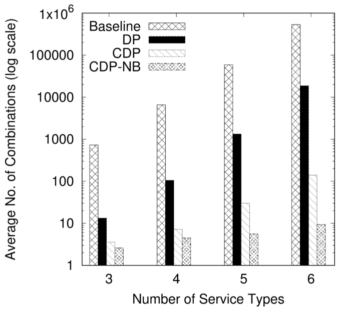
Fig. 25. Effect of network size ($k = 5$, $c = 9$).

5.2.4 Number of Cycle-Distance Computations of MRNVD Approaches

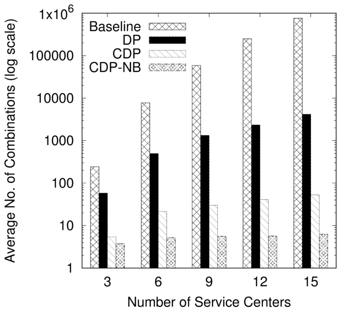
The fourth set of experiments measured the efficiency of pruning algorithms in reducing the number of service center combinations used for the cycle-distance computation. We tested four approaches: (1) Baseline, (2) Distance bounded Pruning, (3) Cycle-Distance bounded Pruning and (4) CDP with Neighboring Bounds. For performance measurement, we averaged the number of cycle-distance computations for an input graph over the number of graph-nodes. We used five road maps (see Table 5) to perform the pruning efficiency analysis. Service center locations were chosen from OpenStreetMap [38]. First, we fixed the number of graph-nodes (i.e., n) to $10,000$ and the number of centers per service type (i.e., c) to 9. We then varied the number of service types from 3 to 6. Fig. 26a shows that CDP uses a lower number of service center combinations for cycle-distance calculation than the Baseline and DP approaches. CDP-NB outperforms other approaches because it reduces the size of multi-level cycle-distance tables. As the number of service types increases, the average number of cycle-distance computations increases for all approaches. Second, we fixed the number of graph-nodes (i.e., n) to $10,000$ and the number of service types (i.e., k) to 5. We then varied the number of centers per service type (i.e., c) from 3 to 15. Fig. 26b shows that CDP-NB uses lower number of service center combinations for cycle-distance calculation than the Baseline, DP, and CDP approaches. As the number of centers per service type increases (i.e., c), the average number of cycle-distance calculations increases for all approaches. Lastly, we fixed the number of service types (i.e., k) to 5 and the number of centers per service type (i.e., c) to 9. We then varied the number of graph-nodes from $5,000$ to $20,000$. Fig. 26c shows that on average, CDP-NB uses a lower number of combinations for the cycle-distance calculation than the Baseline, DP, and CDP approaches. As the number of graph-nodes increases, the average number of combinations does not change. The results of the average number of cycle-distance computations analysis clearly shows that CDP efficiently prunes the search space of service center combinations. Furthermore, the CDP-NB approach utilizes the allotment of neighboring nodes to further prune the search space and reduce the number of service center combinations. Pruning service center combinations yields a reduction in the cost of the cycle-distances computation as the number of graph-nodes (or service centers) increases.

5.2.5 Comparison with Integer Programming Method

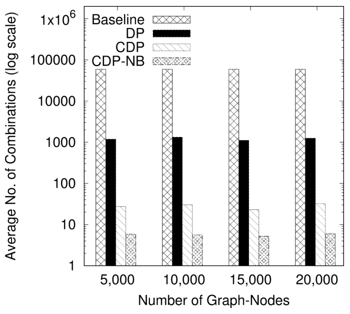
The fifth set of experiments compared our proposed approaches with existing approaches for GTSP. We built an IP formulation of MRNVD based on the GTSP approaches [25], [26] and run the IP solver using OjAlgo [39]. We fixed the number of service types to 5 and the number of centers per service type to 9. We varied the number of graph-nodes from 100 to 500. Service center locations were chosen from OpenStreetMap to construct 15 test runs for each road network. Performance measurements were execution time and the sum of cycle-distances. The performance measurements were averaged over 75 test runs. Fig. 27a gives the execution times. As can be seen, both CDP and CDP-NB outperforms others. The IP formulation of MRNVD uses Lagrangian relaxation, branch-and-bound, and branch-and-cut techniques to find the optimal cycle distance [25], [26]. However, it cannot utilize the topological connectivity of road maps and prune the search space for service center combinations. The results show that the IP method was not scalable for sizable road maps. The performance gap increases as the number of graph-nodes increases. When comparing the sum of the cycle-distances, we see that all approaches produce the optimal solution (see Fig. 27b).



(a) Effect of No. of Service Types ($n = 10,000$, $c = 9$)

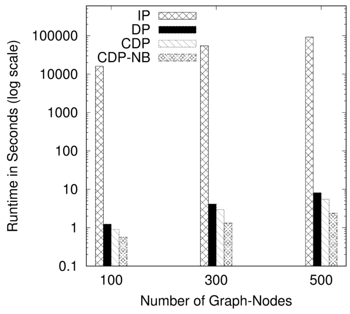


(b) Effect of No. of Centers per Service Type ($n = 10,000$, $k = 5$)

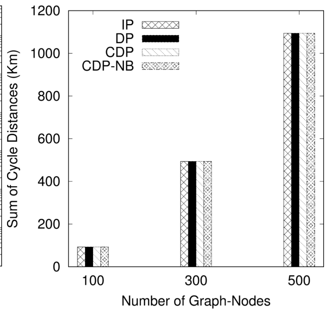


(c) Effect of No. of Graph-nodes ($k = 5$, $c = 9$)

Fig. 26. Average Number of generated service center combinations.



(a) Run Time Comparison



(b) Comparison of Solution Quality

Fig. 27. Performance comparison with the GTSP approach ($k = 5$, $c = 9$).

5.2.6 Discussion

The proposed CDP approach achieves a significant computation performance gain over our preliminary work [6]. This improvement was obtained by using three key components: 1) Set-Window, 2) Multi-level Cycle-Distance Tables, and 3) Vertical and Horizontal pruning. The DP approach uses only two service centers to define the lower and upper bound of the optimal cycle-distance. The DP constructs Triangle-Distance Tables to identify the optimal cycle distance. However, the size of Triangle-Distance Tables becomes very large as the size of network increases. This limitation reduces the efficiency of pruning for service center combinations. To remedy this issue, CDP uses multi-level Cycle-Distance Tables to eliminate unnecessary service center combinations and reduces the search space based on Vertical and Horizontal pruning. This novel approach significantly reduces the computational cost because in each iteration, CDP eliminates unnecessary service centers to reduce the size of multi-level Cycle-Distance Tables and generates a set of service center combinations that are smaller than those of DP. We also proposed the CDP-NB approach that can utilize the allotments of neighboring nodes to find a more strict upper bound for the cycle distance computation. The experimental results show that the CDP-NB approach reduced the size of multi-level Cycle-Distance Tables and improved the efficiency of Vertical and Horizontal pruning. We also see that the output (i.e., MRNVD) of CDP and CDP-NB is optimal and is the same as that of the DP approach.

6 Conclusion and Future work

We presented the problem of creating a Multiple Resource Network Voronoi Diagram (MRNVD). An important societal application of MRNVD is promoting transportation resilience before or after a disaster. The MRNVD problem is challenging due to multiple different types of resources. Traditional Network Voronoi Diagram (NVD) uses the shortest distances and divides the region based on the closest service center. However, the application of NVDs is limited to a single type of resource. In this paper, we describe our novel Cycle-Distance bounded Pruning (CDP) approach for creating an MRNVD that can minimize the total cycle-distances of graph-nodes to allotted k service centers. In addition, we proposed the CDP with Neighboring Bounds (CDP-NB) approach that can utilize the allotments of neighboring nodes to reduce the computation cost of CDP. Experiments using five different road maps demonstrated that our proposed algorithms significantly reduce the computational cost against our prior work.

In future work, we will develop a parallel formulation of the proposed approaches to handle continental-sized transportation networks. We will also investigate the effect of applying spatial filters on reducing the size of the network and improving the performance of MRNVD. MRVND with Monte Carlo simulation may solve the facility location problem [32], [33]. We will study new methods that determine the near-optimal positions of service facilities. Lastly, we plan to design a new MRNVD problem that includes the capacity constraint for each service center and the directional constraint based on directed graphs.

Acknowledgments

The authors would like to thank the US National Science Foundation under Grant No. 1844565.

References

[1] ABC News (2020), "2020 may be 'one of the busiest' hurricane seasons on record: Noaa," https://abcnews.go.com/US/2020-busiest-hurricane-seasons-record-noaa/story?id=72213828, Retrieved Sep. 2020.
[2] CBS News (2020), "California wildfires reach devastating milestone, scorching more than 4 million acres," https://www.cbsnews.com/news/california-wildfires-break-record-scorch-more-than-4million-acres-2020-10-04/, Retrieved Oct. 2020.
[3] Homeland Security, "National Preparedness Goal – Second Edition," https://www.fema.gov/sites/default/files/2020-06/national_preparedness_goal_2nd_edition.pdf, Retrieved Oct. 2020.
[4] G. B. Dantzig and J. H. Ramser, "The truck dispatching problem," *Management science*, vol. 6, no. 1, pp. 80–91, 1959.

[5] D. J. Rosenkrantz, R. E. Stearns, and P. M. Lewis, II, "An analysis of several heuristics for the traveling salesman problem," *SIAM journal on computing*, vol. 6, no. 3, pp. 563–581, 1977.

[6] A. Qutbuddin and K. Yang, "Multiple Resource Network Voronoi Diagram," in *11th International Conference on Geographic Information Science (GIScience 2021) - Part I*, vol. 177. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2020, pp. 11:1–11:16.

[7] M. Erwig, "The graph voronoi diagram with applications," *Networks: An International Journal*, vol. 36, no. 3, pp. 156–163, 2000.

[8] A. Okabe, T. Satoh, T. Furuta, A. Suzuki, and K. Okano, "Generalized network voronoi diagrams: Concepts, computational methods, and applications," *International Journal of Geographical Information Science*, vol. 22, no. 9, pp. 965–994, 2008.

[9] A. Okabe, B. Boots, and K. Sugihara, "Nearest neighbourhood operations with generalized voronoi diagrams: a review," *International Journal of Geographical Information Systems*, vol. 8, no. 1, pp. 43–71, 1994.

[10] S. Nutanong, E. Tanin, M. E. Ali, and L. Kulik, "Local network voronoi diagrams," in *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2010, pp. 109–118.

[11] A. Mehta, K. Malik, V. M. Gunturi, A. Goel, P. Sethia, and A. Aggarwal, "Load balancing in network voronoi diagrams under overload penalties," in *International Conference on Database and Expert Systems Applications*. Springer, 2018, pp. 457–475.

[12] K. Yang, A. H. Shekhar, D. Oliver, and S. Shekhar, "Capacity-constrained network-voronoi diagram," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 11, pp. 2919–2932, 2015.

[13] A. Okabe and K. Sugihara, *Spatial analysis along networks: statistical and computational methods*. John Wiley & Sons, 2012.

[14] M. T. Dickerson and M. T. Goodrich, "Two-site voronoi diagrams in geographic networks," in *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, 2008, pp. 1–4.

[15] M. T. Dickerson, M. T. Goodrich, T. D. Dickerson, and Y. D. Zhuo, "Round-trip voronoi diagrams and doubling density in geographic networks," in *Transactions on Computational Science XIV*. Springer, 2011, pp. 211–238.

[16] D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook, *The traveling salesman problem: a computational study*. Princeton university press, 2006.

[17] M. Kolahdouzan and C. Shahabi, "Voronoi-based k nearest neighbor search for spatial network databases," in *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, 2004, pp. 840–851.

[18] M. R. Kolahdouzan and C. Shahabi, "Alternative solutions for continuous k nearest neighbor queries in spatial network databases," *GeoInformatica*, vol. 9, no. 4, pp. 321–341, 2005.

[19] G. Zhao, K. Xuan, D. Taniar, M. Safar, M. Gavrilova, and B. Srinivasan, "Multiple object types knn search using network voronoi diagram," in *International Conference on Computational Science and Its Applications*. Springer, 2009, pp. 819–834.

[20] Q. T. Tran, D. Taniar, and M. Safar, "Reverse k nearest neighbor and reverse farthest neighbor search on spatial networks," in *Transactions on large-scale data-and knowledge-centered systems I*. Springer, 2009, pp. 353–372.

[21] M. Fischetti, J.-J. Salazar-González, and P. Toth, "The generalized traveling salesman and orienteering problems," in *The traveling salesman problem and its variations*. Springer, 2007, pp. 609–662.

[22] Y.-N. Lien, E. Ma, and B. W.-S. Wah, "Transformation of the generalized traveling-salesman problem into the standard traveling-salesman problem," *Information Sciences*, vol. 74, no. 1-2, pp. 177–189, 1993.

[23] G. Laporte and Y. Nobert, "Generalized travelling salesman problem through n sets of nodes: an integer programming approach," *INFOR: Information Systems and Operational Research*, vol. 21, no. 1, pp. 61–75, 1983.

[24] G. Laporte and F. Semet, "Computational evaluation of a transformation procedure for the symmetric generalized traveling salesman problem," *INFOR: Information Systems and Operational Research*, vol. 37, no. 2, pp. 114–120, 1999.

[25] C. E. Noon and J. C. Bean, "A lagrangian based approach for the asymmetric generalized traveling salesman problem," *Operations Research*, vol. 39, no. 4, pp. 623–632, 1991.

[26] M. Fischetti, J. J. Salazar González, and P. Toth, "A branch-and-cut algorithm for the symmetric generalized traveling salesman problem," *Operations Research*, vol. 45, no. 3, pp. 378–394, 1997.

[27] L. A. Wolsey, *Integer programming*. Wiley Online Library, 1998, vol. 42.

[28] M. R. Garey and D. S. Johnson, "Computers and intractability," *A Guide to the Theory of NP-Completeness*, 1979.

[29] L. V. Snyder and M. S. Daskin, "A random-key genetic algorithm for the generalized traveling salesman problem," *European journal of operational research*, vol. 174, no. 1, pp. 38–53, 2006.

[30] S. L. Smith and F. Imeson, "GLNS: An effective large neighborhood search heuristic for the generalized traveling salesman problem," *Computers & Operations Research*, vol. 87, pp. 1–19, 2017.

[31] J. Renaud and F. F. Boctor, "An efficient composite heuristic for the symmetric generalized traveling salesman problem," *European Journal of Operational Research*, vol. 108, no. 3, pp. 571–584, 1998.

[32] M. Daskin, *Network and discrete location: models, algorithms, and applications*, ser. Wiley-Interscience series in discrete mathematics and optimization. Wiley, 2013.

[33] Z. Reza and H. Masoud, *Facility Location: Concepts, Models, Algorithms and Case Studies*. Springer Dordrecht Heidelberg London New York, 2009.

[34] R. Bellman, "Dynamic programming treatment of the travelling salesman problem," *Journal of the ACM (JACM)*, vol. 9, no. 1, pp. 61–63, 1962.

[35] M. Held and R. M. Karp, "A dynamic programming approach to sequencing problems," *Journal of the Society for Industrial and Applied mathematics*, vol. 10, no. 1, pp. 196–210, 1962.

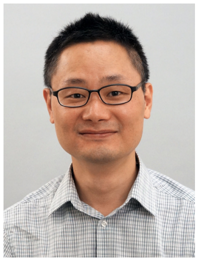
[36] G. J. Woeginger, "Exact algorithms for np-hard problems: A survey," in *Combinatorial optimization—eureka, you shrink!* Springer, 2003, pp. 185–207.

[37] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin, *Network flows*. Cambridge, Mass.: Alfred P. Sloan School of Management, Massachusetts, 1988.

[38] OpenStreetMap, http://goo.gl/Hso0, Retrieved Feb. 2020.

[39] "oj! Algorithms," http://ojalgo.org/, Retrieved Apr. 2021.

Ahmad Qutbuddin is a PhD candidate in Computer Engineering at Florida Atlantic University. He received the BS degree in Computer Engineering from King Fahd University of Petroleum and Minerals in 2010, and the master's degree in Computer Engineering from the University of Central Florida in 2017. He worked as a Lecturer for Umm Al-Qura University in Makkah, Saudi Arabia from 2010 to 2014. His research interests are spatial network databases and Network Voronoi Diagram.



KwangSoo Yang is an assistant professor in the Department of Computer and Electrical Engineering and Computer Science at Florida Atlantic University. He received the BS degree in electrical engineering from Yonsei University in 1998, the master's degree in computer science from the University of Minnesota in 2010, and the PhD degree in computer science from the University of Minnesota in 2015. He was a software engineer for LG CNS in Seoul, Korea, from 2001 to 2008. He has received an NSF CAREER Award in 2019. His research interests include spatio-temporal network databases, spatio-temporal networks, and evacuation routing problems.