Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 06-017

Capacity Constrained Routing Algorithms for Evacuation Route Planning

Qingsong Lu, Betsy George, and Shashi Shekhar

May 04, 2006

Dagstuhl Seminar Proceedings 10121
Computational Transportation Science
http://drops.dagstuhl.de/opus/volltexte/2010/2721

Capacity Constrained Routing Algorithms for Evacuation Route Planning

Qingsong Lu, Betsy George, Shashi Shekhar

This work was supported by Army High Performance Computing Research Center contract number DAAD19-01-2-0014 and the Minnesota Department of Transportation contract number 81655. The content of this work does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. Access to computing facilities was provided by the AHPCRC and the Minnesota Supercomputing Institute.

Abstract

Evacuation route planning identifies paths in a given transportation network to minimize the time needed to move vulnerable populations to safe destinations. Evacuation route planning is critical for numerous important applications like disaster emergency management and homeland defense preparation. It is computationally challenging because the number of evacuees often far exceeds the capacity, *i.e.* the number of people that can move along the road segments in a unit time. Linear Programming(LP) based methods using time expanded networks can take hours to days of computation for metropolitan sized problems. In this paper, we propose a new approach, namely a capacity constrained routing planner which models capacity as a time series and generalizes shortest path algorithms to incorporate capacity constraints. We characterize the design space for reducing the computational cost. Analytical cost model and experiment results show that the proposed algorithm is faster than the LP based algorithms and requires less memory. Experimental evaluation using various network configurations also shows that the proposed algorithm produces solutions that are comparable to those produced by LP based algorithms while significantly reducing the computational cost.

Index Terms

evacuation planning, routing and scheduling, transportation network

I. INTRODUCTION

Many disasters, natural or man-made, can lead to situations where people need to be moved from impacted areas to safe destinations. In such scenarios, it is critical to identify routes such that evacuation can be completed in the shortest possible time. Evacuation route planning aims at finding routes in the given transportation network that would minimize the evacuation time. This is a critical step in disaster emergency management and homeland defense preparation. The recent catastrophes caused by hurricanes on the Gulf coast underscore the importance of evacuation planning. Route planning in these circumstances is challenging because of the capacity constraints *i.e.* the limit on the number of people that can move along the road segments in unit time. Effective evacuation route planning that honors the capacity constraints of the transportation network has the potential to reduce congestion during large scale evacuations. A comprehensive approach which addresses capacity constraints and their time-dependence is critical for the effectiveness of any evacuation plan.

Previous approaches [12], [19], [20], [23], [27], [28] to evacuation route planning use linear programming (LP) based methods to generate evacuation plans. These methods incorporate capacity constraints by using time expanded networks and require a user-provided upper bound on the total evacuation time. Although these evacuation planning algorithms generate optimal plans, they are expensive with respect to memory and take a long time (order of hours to days)

to solve problems of the sizes usually encountered in urban evacuation scenarios. An overview of LP based methods is given in Appendix III.

There is an immediate need for a scalable algorithm that quickly generates high quality evacuation plans for metropolitan sized networks. This paper presents a new approach, namely a Capacity Constrained Routing (CCRP) approach, to evacuation route planning. The proposed approach makes use of well known shortest path algorithms and extends them by incorporating capacity constraints. It models capacity as a time series to account for the time dependent nature of the networks. It uses only the original evacuation network instead of the time-expanded network used by the LP based approach and thus requires less memory.

In this paper, we characterize the design space available in the context of the Capacity Constrained Route Planner (CCRP) algorithm and evaluate the performance of the CCRP algorithm for each dimension in the design space. The paper presents analytical cost models for the various design options. Performance evaluation of CCRP was done by conducting experiments on various network configurations. Analytical evaluation and experimental results show that the proposed CCRP algorithm produces high quality solutions, and significantly reduces the computational cost compared to the LP-based approach, which produces optimal but expensive solutions.

A. Application Domain

Evacuation route planning has been identified as a critical step in emergency management. A recent Executive Summary [15] issued by the US Homeland Security Council listed 15 kinds of scenarios, ranging from natural disasters to terrorist attacks, for which government agencies are urged to develop emergency plans and most of these scenarios would require evacuation plans to evacuate large populations to safe areas. Currently, local emergency management authorities often identify evacuation routes by hand using a committee of experts. They do not have computerized tools to consider capacity constraints of the transportation network and thus seldom avoid congestion during evacuation. For example, when Hurricane Andrew was approaching Florida in 1992 (see Figure 14), the lack of effective planning caused tremendous traffic congestion, general confusion and chaos (see Figure 15) [1]. This experience was echoed in the words of Mayor Tim Lott of Morgan City, Louisiana when Hurricane Andrew later headed for that state: ”We packed up Morgan City residents to evacuate in the a.m. on the day that Andrew hit coastal Louisiana, but in early afternoon the majority came back home. The traffic was so bad that they couldn’t get

through Lafayette." [1]. These events illustrate the complexity of evacuation route planning and the fact that the problem extends beyond computing the shortest routes from evacuation points to safe destinations. A comprehensive approach which includes capacity constraints and their time-dependence is critical for the effectiveness of the solution. In very recent times, Hurricane Katrina and Hurricane Rita (see Figure 16) caused similar problems [21]. Figure 17 shows the traffic congestion caused by Hurricane Rita during the Houston evacuation on highway I-45.

Other types of disasters, such as accidents or terrorist attacks (e.g. bio-chemical attack) may also result in the need for massive and rapid evacuations of people from metropolitan areas [10], [11], [15], [17]. In other cases, a disaster may require the evacuation of large buildings (e.g. Pentagon, the Sears Tower).

Thus, efficient tools are needed to produce evacuation plans that identify routes and schedules to quickly evacuate affected populations to safety in the event of natural disasters, terrorist attacks or other types of large-scale emergencies.

B. Problem Formulation

We formulate the evacuation route planning problem as follows:

Given: A transportation network with non-negative integer capacity constraints on nodes and edges, non-negative integer travel times on edges, the total number of evacuees and their initial locations, and locations of evacuation destinations.

Output: An evacuation plan consisting of a set of origin-destination routes and a scheduling of evacuees on each route. The scheduling of evacuees on each route should observe the capacity constraints of the nodes and edges on this route.

Objective: (1) Minimize the evacuation egress time, which is the time elapsed from the start of the evacuation until the last evacuee reaches the evacuation destination. (2) Minimize the computational cost of producing the evacuation plan.

Constraint: (1) Edge travel time preserves the FIFO (First-In First-Out) property. (2) Edge travel time reflects delays at intersections. (3) Limited amount of computer memory.

Example 1- An Evacuation Network: Figure 1 shows an example evacuation network. Each node is shown by an ellipse and has two attributes: maximum node capacity and initial node occupancy. For example, at node N1, the maximum capacity is 50, which indicates that

this node can hold at most 50 evacuees at any time instant. The initial occupancy is shown to be 10, which means there are 10 evacuees at this node when the evacuation starts. In Figure 1, each edge, shown as an arrow, represents a link between two nodes. Each edge also has two attributes: maximum edge capacity and travel time. For example, at edge N4-N6, the maximum edge capacity is 5, which means at each time point, at most 5 evacuees can start to travel from node N4 to N6 through this link. The travel time of this edge is 4, which means it takes 4 time units to travel from node N4 to N6. This approach of modeling an evacuation scenario to a capacitated node-edge graph is similar to those presented in Hamacher [20], Kisko [28] and Chalmet [12].

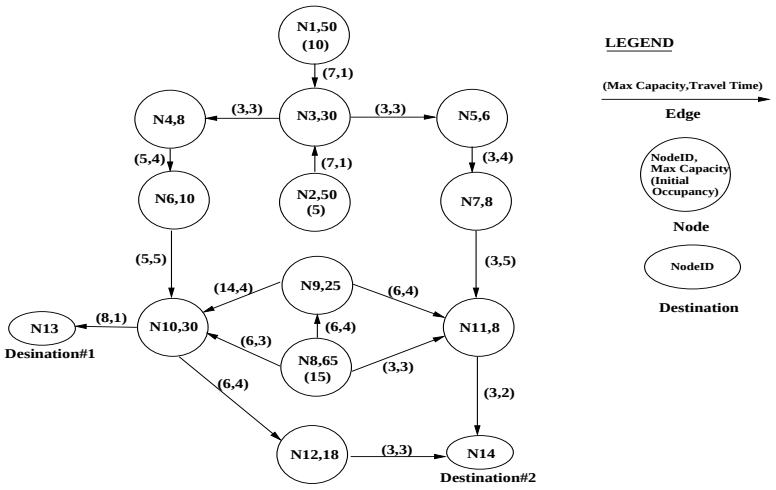


Fig. 1. Node-Edge Graph Model of Example Evacuation Network

As shown in Figure 1, suppose we initially have 10 evacuees at node N1, 5 at node N2, and 15 at node N8. The task is to generate an evacuation plan that evacuates the 30 evacuees to the two destinations (node N13 and N14) using the least amount of time.

Example 2- An Evacuation Plan: Table I shows an example evacuation plan for the evacuation network in Figure 1. In this table, each row shows one group of evacuees moving together during the evacuation with a group ID, source node, number of evacuees in this group, the evacuation route with time schedule, and the destination time. The route is shown by a series of node numbers and the time schedule is shown by a start time associated with each node on the route. Take source node N8 for example; initially there are 15 evacuees at N8. They are

divided into 3 groups: Group A with 6 people, Group B with 6 people and Group C with 3 people. Group A moves from node N8 at time 0 to node N10, then moves from node N10 at time 3 to node N13, and reaches destination N13 at time 4. Group B follows the same route as group A, but has a different schedule due to capacity constraints of this route. This group moves from N8 at time 1 to N10, then moves from N10 at time 4 to N13, and reaches destination N13 at time 5. Group C takes a different route. It moves from N8 at time 0 to N11, then moves from N11 at time 3 to N14, and reaches destination N14 at time 5. The procedure is similar for other groups of evacuees from source node N1 and N2. The whole evacuation egress time is 16 time units since the last groups of people (Groups H and I) reach destinations at time 16. This evacuation plan is an optimal plan for the evacuation scenario shown in Figure 1.

Alternate problem formulations of the evacuation problem are available by changing the objective

TABLE I
EXAMPLE EVACUATION PLAN

Group of Evacuees			Route with Schedule	Dest. Time
ID	**Source**	**No. of Evacuees**		
A	N8	6	N8(T0)-N10(T3)-N13	4
B	N8	6	N8(T1)-N10(T4)-N13	5
C	N8	3	N8(T0)-N11(T3)-N14	5
D	N1	3	N1(T0)-N3(T1)-N4(T4)-N6(T8)-N10(T13)-N13	14
E	N1	3	N1(T0)-N3(T2)-N4(T5)-N6(T9)-N10(T14)-N13	15
F	N1	1	N1(T0)-N3(T1)-N5(T4)-N7(T8)-N11(T13)-N14	15
G	N2	2	N2(T0)-N3(T1)-N5(T4)-N7(T8)-N11(T13)-N14	15
H	N2	3	N2(T0)-N3(T3)-N4(T6)-N6(T10)-N10(T15)-N13	16
I	N1	3	N1(T1)-N3(T2)-N5(T5)-N7(T9)-N11(T14)-N14	16

of the problem. The main objective of our problem formulation is to minimize the evacuation egress time. Two alternate objectives are: (1) Maximize the number of evacuees that reach the destination for each time unit; (2) Minimize the average evacuation time for all evacuees. Jarvis and Ratliff presented and proved the *triple optimization theorem* [25], which illustrates the properties of the solutions that optimize the above objectives of the evacuation problem.

C. Related Work and Our Contribution

The previous approach for evacuation route planning uses a linear programming (LP) based method. It models the evacuation problem as a network flow problem [6], [18] and finds the optimal solution using LP based method solvers. Hamacher and Tjandra [20] gave an extensive

literature review of the models and algorithms used in these LP based methods. Based on the triple-optimization results by Jarvis and Ratliff [25], the LP based method for evacuation route planning works as follows. First, it models the evacuation network into a graph, as shown by network G in Figure 19, and it requires the user to provide an estimated upper bound T of the evacuation egress time. Second, it converts evacuation network G to a time-expanded network*, by duplicating the original evacuation network G for each discrete time unit t = 0, 1, ..., T. Then, it defines the evacuation problem as a minimum cost network flow problem [6], [18] on the time-expanded network G_T. Finally, it feeds the expanded network G_T to minimum cost network flow solvers, such as NETFLO [26], to find the optimal solution. For example, EVACNET [12], [19], [27], [28] is a computer program based on this approach which computes egress time for building evacuations. It uses NETFLO code to obtain the optimal solution. Hoppe and Tardos [23], [24] gave a polynomial time bounded algorithm by using the ellipsoid method of linear programming to find the optimal solution for the minimum cost flow problem. Theoretically, the ellipsoid method has a polynomial bounded running time.

Limitations of Related Work: The LP based approaches can produce optimal solutions for evacuation route planning. It is useful for evacuation scenarios with small size networks(several hundreds of nodes and edges), such as building evacuation. However, this approach has the following limitations. First, it significantly increases the problem size because it requires time-expanded network G_T to produce a solution. As can been seen in Figures 19 and 20, if the original evacuation network G has n nodes and the time upper bound is T, the time-expanded network G_T will have at least $(T + 1)n$ nodes. This approach may not be able to scale up to large size (tens of thousands of nodes and edges) transportation networks in urban evacuation scenarios due to high computational run-time caused by the tremendously increased size of the time-expanded network. Second, the LP based approach requires the user to provide an upper bound T of the evacuation time in order to generate the time-expanded network. It is difficult, however, to precisely estimate the evacuation time for an urban scenario where the number of evacuees is large and the transportation network is complex. An under-estimated time bound T will result in failure of finding a solution. In this case, the user will have to increase the value of

*Details of time-expanded network are available in Appendix II-A.

T and re-run the algorithm until a solution can be reached. On the other hand, an over-estimated T will result in an over-expanded network G_T and hence lead to unnecessary storage and run-time and would adversely affect the scalability to large networks.

Our Contributions: To begin to address the limitations of the previous methods, Lu, Huang and Shekhar [31] proposed a heuristic capacity constrained routing algorithm CCRP_03 (formerly called MRCCP in [31]) for evacuation route planning. It has a computational complexity $O(p \cdot n^2 logn)$ (where n the is number of nodes and p is the number of evacuees). Lu, George and Shekhar [30] presented an improved algorithm CCRP_05, which reduced the run-time to $O(p \cdot nlogn)$ by optimizing the shortest path search in CCRP_03. In this paper, we propose an improved heuristic algorithm (CCRP_06), based on CCRP_05 [30], by exploring available design decisions for CCRP_05. We characterize the design space available in the context of the CCRP algorithms and evaluate the performance of the CCRP algorithms for each of the design decisions. Since the shortest path computation is the bottleneck step in CCRP, a wide range of shortest path algorithms and related data structures [13], [14], [16], [34], [36] are explored. Experiment results show that Dijkstra's algorithm with double-bucket data structure gives the best performance for CCRP. We prove that CCRP_06, which uses Dijkstra's algorithm with double-bucket, has an improved run-time of $O(p \cdot (m + 2Cn))$, which is faster than the LP based method in real evacuation scenarios. We also show that CCRP_06 requires less memory than the LP based algorithm. Experimental evaluation of CCRP_06 was conducted using various network configurations to test the performance and the solution quality under different network parameters. Results show that CCRP_06 produces high quality solutions and is much more computationally efficient than the LP based algorithm. It is also shown that CCRP_06's advantage over the LP based algorithm increases with increase in the number of destination nodes in the network.

We also developed an optimal algorithm using A* search [33], [35]. This algorithm addresses the limitations of the LP based approach by using only the original evacuation network to find the optimal solution and it does not require a user-provided upper bound on evacuation time. We provide the proof of monotonicity and admissibility of this A* search algorithm. However, our experiments showed that this method is not scalable to large size networks. For interested readers, we have included the details of this approach in Appendix VI.

D. Scope and Outline of the Paper

The main focus of the paper is on the analysis of a heuristic algorithm which effectively extends a shortest path algorithm to account for the capacity constraints of a road network and thus provides a simpler and computationally efficient solution to evacuation route planning. In this framework, the evacuation network will be modeled as a graph and the capacities of the edges and nodes will be modeled using time series. In our problem formulation, we allow time dependent node capacity and edge capacity, but we assume that edge capacity does not depend on the actual flow amount in the edge. We also allow time dependent edge travel time, but we require that the network preserve the FIFO (First-In First-Out) property. Though the model cannot handle travel times represented as continuous functions of time, a dynamic model represented by a discrete function can be easily incorporated in this model.

Outline of the Paper: The rest of the paper is organized as follows. Section 2 presents our heuristic approach to the problem. This section explains the heuristic algorithm and lists the various design choices available in the context of the heuristic algorithm. Sections 3 and 4 deal with the evaluations of the various design choices; Section 3 gives an analytical evaluation of various candidates pertaining to every design decision relevant to the performance of the evacuation route planning algorithm. In Section 4, we present an experimental study to assess the relative merits of various options available in every design decision. We give the conclusions and discuss future work in Section 5.

II. Proposed Heuristic Approach

As discussed in Section 1, the LP based methods to solve the evacuation route planning problem use time expanded networks that require a large amount of memory; these methods also require a prior knowledge of the upper bound of evacuation time. We formulated the evacuation route planning problem as a search problem implemented as an A* search as a new approach to generate optimal solution without using time-expanded networks (See Appendix VI). Though this method finds optimal routes, its performance evaluation raises some questions about its scalability to metro-sized networks. We do not expect any drastic change in scalability unless we can formulate another heuristic which would require less computation and memory. The urgent need for high quality, scalable solutions in evacuation route planning is thus the motivation behind the exploration of heuristic methods in evacuation. This section discusses in detail the

CCRP algorithms, which have demonstrated very high scalability.

A. Algorithm Framework

The algorithm discussed in this section uses a heuristic method to solve the evacuation route planning problem. The basic idea behind the heuristic is to send the largest possible number of evacuees on the shortest route to the nearest destination. Though the method relies on shortest path algorithms to accomplish this, it contributes considerably in terms of extension of these algorithms to account for the capacity constraints encountered in real world evacuation networks. Though the algorithm does not always yield an optimal solution (with minimum evacuation time), the scalability and time complexity of this method show drastic improvement over the optimal methods.

B. Representation of the Temporal Network

In this representation, the edge capacity and node capacity are modeled as a time series instead of fixed numbers. This time series stores the available capacity at each time instant for a given edge or node. The next section discusses a heuristic approach which uses this representation to extend the shortest path algorithms [14], [16] to account for capacity constraints of the network. This representation is clearly illustrated in Tables V and VI (Appendix II)which show the time series representation of node and edge capacities of the network shown in Figure 19.

C. Heuristic Approach - CCRP Algorithms

In this section, we present a generic description of the Capacity Constrained Route Planner (CCRP). CCRP is a heuristic algorithm which is based on an extension of shortest path algorithms [14], [16] to account for capacity constraints of the network.

The CCRP algorithm uses an iterative approach. In each iteration, the algorithm first searches for route R with the earliest destination arrival time from any source node to any destination node, taking previous reservations and possible waiting time into consideration. Next, it computes the actual number of evacuees that will travel through route R. This number is affected by the available capacity of route R and the remaining number of evacuees. Then, it reserves the node and edge capacity on route R for those evacuees. The algorithm continues to iterate until all evacuees reach the destination. An outline of the algorithm is shown in Algorithm 1. The detailed pseudo-code and algorithm description are given in Appendix IV.

Algorithm 1 Capacity Constrained Route Planner (CCRP)

Input:
1) $G(N, E)$: a graph G with a set of nodes N and a set of edges E;
define type nn non-negative integer
Each node $n \in N$ has two properties:
$Maximum\_Node\_Capacity(n)$, $Initial\_Node\_Occupancy(n)$: nn
Each edge $e \in E$ has two properties:
$Maximum\_Edge\_Capacity(e)$, $Travel\_time(e)$: nn
2) S: set of source nodes, $S \subseteq N$;
3) D: set of destination nodes, $D \subseteq N$;
Output: Evacuation plan : Routes with schedules of evacuees on each route
Method:

```
(1)while any source node s ∈ S has evacuee do {
(2)              closest_pair_shortest_path();
(3)              compute_flow();
/* k is the number of nodes on the shortest path */
(4)              for i = 0 to k − 1 do {
(5)                     reserve_flow();
      }
}
```

Output evacuation plan with routes and schedules of evacuees on each route;

The CCRP algorithm keeps iterating as long as there are still evacuees left at any source node (line 1). Each iteration starts by finding the route R with the earliest destination arrival time from any source node to any destination node based on the current available capacities (line 2). This is done by generalizing Dijkstra's shortest path algorithm [14], [16] to work with the time series node and edge capacities and edge travel time. Route R is the route that starts from a source node and gets to a destination node in the least amount of time, and available capacity of the route allows at least one person to travel through route R to a destination node. Given the evacuation network in Figure 1, the example execution trace of CCRP is as follows:

Example 3- CCRP Execution Trace: At the very first iteration, route R will be N8-N10-N13. Evacuees from source node N8 can take this route to reach destination N13 at time 4 using the time schedule N8(T0)-N10(T3)-N13. At algorithm line 3, the actual number of evacuees that will travel through route R is determined by taking the smallest number among the number of evacuees at the source node and the available capacities of each nodes and edges on route R based on the time schedule that evacuees will travel through each node and edge. Thus, at the first iteration, this flow amount of R will be 6, which is the available edge capacity of edge N8-N10 at time 0. The next step is to reserve capacities for the evacuees on each node and edge of route R based on the time schedule(lines 4-7). At the first iteration, the algorithm makes a reservation for

the 6 evacuees by reducing the available capacity of each node and edge at corresponding time points. This means that available capacities are reduced by 6 for edge N8-N10 at time 0, for node N10 at time 3, and for edge N10-N13 at time 3. The 6 evacuees arrive at destination N13 at time 4. Then, the algorithm goes back to line 1 for the next iteration(line 8). The iteration terminates when the occupancy of all source nodes is reduced to zero, which means all evacuees have been sent to destination nodes. Line 9 outputs the evacuation plan, as shown in Table I. A more detailed illustration of the iterations of the algorithm on the network is shown in in Appendix V.

D. Design Decisions in the CCRP Algorithm

The CCRP algorithm uses shortest path computation as one of its key steps to generate the evacuation plan. This section evaluates the choices that are available in the context of this computation. For details on the design space in the context of evacuation planning algorithms, the reader can refer to Appendix II. This section also lists the design options available specifically in the context of the proposed CCRP algorithm.

1) Choice of Algorithm to Identify Closest Source-Destination Pair: The performance of the heuristic algorithm depends heavily on the efficiency in computing the shortest paths from source nodes to destination nodes. The run time increases in proportion to the number of runs of the algorithm. The CCRP_06 algorithm makes a major improvement in the algorithm used to find the quickest route between the closest source-destination pair. In CCRP_05, finding the quickest route R is done by running generalized shortest path searches from each source node. Each search is terminated when any destination node is reached. In CCRP_06, this step is improved by adding a super source node s_0 to the network and connecting s_0 to all source nodes. This allows us to complete the search for route R by using only one single generalized shortest path search, which takes the super source s_0 as the start node. This search terminates when any destination node is reached. Since the super source s_0 is connected to each source nodes by an edge with infinite capacity and zero travel time, it can be easily proved that the shortest route found by this search is the route R that we need. This improvement significantly reduces the computational cost of the algorithm by one degree of magnitude compared with CCRP_05.

2) Algorithms Used in the Shortest Path Computation: The most computationally intense task in the CCRP algorithms is the computation of the shortest path from a source to destinations.

The performance of the CCRP algorithm depends significantly on the shortest path algorithm used. Although a number of evaluations of the existing shortest path algorithms are available, there is no clear answer as to which algorithm would perform the best in our case. In this section we explore a set of shortest path algorithms that belong to the groups of label setting and label correcting algorithms [13] and try to evaluate them in the context of evacuation route planning. These two types of algorithms differ in the criteria used in the selection of nodes for scanning. This leads to a difference in the ways they update the estimate of the shortest path distance(label) associated with each node and in the ways in which they converge to the optimal shortest path distance.

Dijkstra's algorithm is one of the most widely used label setting algorithm. This paper evaluates the performance of the evacuation route planning algorithm when Dijkstra's algorithm is used to compute the shortest path. Since this algorithm selects the node with the shortest distance estimate as the next node to be scanned, the algorithm performs well when the destination node is "close" to the source node. Label correcting algorithms, when implemented with suitable data structures can outperform Dijkstra's implementations when the destination nodes are "far away" from the source nodes. The performance of these algorithms depends on the number of destination nodes in the problem formulation and the length of the shortest path from the source to the destination relative to the longest, shortest path in the network. Since we do not always know these parameters in advance, the performance of label correcting algorithms, in addition to label setting algorithms, needed to be evaluated in the context of the CCRP algorithm.

3) Data Structures Used in the Shortest Path Computation: Here we describe two different versions of Dijkstra's algorithm and two versions of a label correcting algorithm. These algorithms are reported [40] to give the best performance among all shortest path algorithms on road networks. They differ in the data structures used to maintain the set of labeled nodes.

The double bucket implementation of Dijkstra's algorithm is a modified version of Dial's implementation. The details of this implementation are given in Appendix II-E.The complexity of the algorithm is $O(m + n(\Delta + C/\Delta))$. This implementation of Dijkstra's algorithm is especially suitable for networks with non-negative arc lengths, which is the case for road network used in evacuation route planning. Since the CCRP algorithms use the shortest path algorithm to find the shortest path from a single source to a single destination, there is a likelihood that the double bucket implementation would outperform any label setting algorithm since it can

terminate as soon as the destination node is reached. The difference in the running times of various implementations of the label setting algorithms is due to the difference in the computation involved in selecting a labeled node with the minimum label. In contrast to dense graphs, where this computation is small compared to the work involved in node scans, the selection process would be a significant part in sparse graphs. Bucket implementations appear to be cheaper than heap implementations since heap operations are expensive unless the number of nodes in the heap is small. Among the bucket implementations, double bucket implementation is preferred since it uses less memory than Dial's implementation. Road networks are generally sparse ($m \approx 4n$) and hence the bucket implementation of the algorithm would be efficient. Dijkstra's algorithm using Fibonacci heaps qualifies as a candidate because of its best, worst-case complexity. The asymptotic complexity of the algorithm is $O(m + nlogn)$.

Despite the worse asymptotic performance ($O(n^2m)$ of the Two-Q algorithm, it has the potential of performing better if the destination node is sufficiently far away from a given source node. The Two-Q algorithm is a good choice as a candidate algorithm since at each iteration of the CCRP algorithms we do not have the prior knowledge about the shortest path distance from the(super)source to a destination node relative to the longest shortest path distance in a shortest path tree rooted at the source node. But, it must be noted that in some iterations, we would have cases where the shortest path distance from the source to the destination is a small fraction of the longest shortest path distance in the shortest path tree. This prompts the choice of Dijkstra's algorithm in addition to the label correcting algorithm. Also, since the CCRP algorithm computes the shortest path from a single source to multiple destinations, the relative performance of the shortest path algorithms can depend on the number of destination nodes relative to the total number of nodes. The label correcting methods may have an edge over the label setting methods in scenarios where the number of destinations is a significant fraction of the total number of nodes.

III. Analytical Evaluation of CCRP Design Decisions

In this section, we give an analytical evaluation of the different options available for each of the design decisions of the CCRP algorithm. Each subsection evaluates the options for one CCRP design decision. The options are listed in Figure 21.

A. Heuristic vs Optimal Algorithms

Here, we compare the computational cost of optimal algorithms and heuristic algorithms by providing analytical evaluation of both methods.

Optimal methods using Linear Programming:

The computational cost of the LP approach depends on the method used to solve the minimum cost flow problem. Hoppe and Tardos [23] showed that this problem can be solved using the ellipsoid method, which is theoretically polynomial time bounded. However, the computational complexity of the ellipsoid method is at least $O(N^6)$ [9](where N is the number of nodes in the network). Since the LP approach requires a time-expanded network, in which N equals $(T+1)n$ (where n is the number of nodes in the original evacuation network and T is the user-provided evacuation time upper bound), the optimal algorithm using LP based runs in at least $O((T \cdot n)^6)$ time.

Heuristic Method:

We now provide the algebraic cost model for the computational cost of the heuristic algorithm presented in Lu, George, and Shekhar [30]. The CCRP algorithm is an iterative approach. In each iteration, the route for one group of people is chosen and the capacities along the route are reserved. The total number of iterations equals the number of groups generated. In the worst case, each individual evacuee forms one group. Therefore, the upper bound of the number of groups is p, i.e. the number of iterations is $O(p)$. In each iteration, the computation of the route R with earliest destination arrival time is done by running one generalized Dijkstra's shortest path search. The worst case computational complexity of Dijkstra's algorithm is $O(n^2)$ for dense graphs [14]. Various implementations of Dijkstra's algorithm have been developed and evaluated extensively [6], [13], [40]. Many of these implementations can reduce the computational cost by taking advantage of the sparsity of the graph. Transportation road networks are very sparse graphs with a typical edge/node ratio around 3. In this paper, we implement the shortest path search in the CCRP_06 algorithm using Dijkstra's algorithm with double bucket data structures, which runs in $O(m + n(\Delta + C/\Delta))$ time [13], where Δ is the bucket size and C is the maximum edge weight. In our implementation, Δ is set to the biggest power of two that is less than $\sqrt{C}$ [13].

The time complexity is hence $O(m+2Cn)$. The generalization of Dijkstra's algorithm to account for capacity constraints affects only how the shortest distance to each node is defined. It does not affect the computational complexity of the algorithm. Therefore, the search for route R in CCRP_06 takes $O(m + 2Cn)$ time. The reservation step is done by updating the node and edge capacities along route R, which has a cost of $O(n)$. Each iteration of the CCRP_06 algorithm is done in $O(m + 2Cn)$ time. It takes $O(p)$ iterations to complete the algorithm. The cost model of the algorithm is $O(p \cdot (m + 2Cn))$. The LP based approach produces optimal solutions but suffers from high computational cost. A heuristic method reduces the computation cost though it produces a sub-optimal solution.

Lemma 1: CCRP_06 is asymptotically faster than LP based algorithm when $p < \frac{T^6}{3+2C}n^5$.

Proof: The cost model for CCRP_06 is in $O(p \cdot (m + 2Cn))$. Transportation road networks are very sparse graphs with a typical edge/node of less than 3, i.e. $m \leq 3n$. This means CCRP_06 runs in $O(p \cdot (3 + 2C)n)$ time on road networks. The optimal algorithm using LP runs in at least $O((T \cdot n)^6)$ time. Therefore, CCRP_06 is asymptotically faster than LP algorithm when $p < \frac{T^6}{3+2C}n^5$. This condition is almost always true in a real evacuation scenario, in which n (number of nodes in the network) ranges from hundreds to millions and T (upper-bound on evacuation time) can be hundreds of minutes.

B. Temporal Network Framework - Time Expanded Network vs Time Series Representation

Another design decision is to choose a framework to represent the temporal network. The two available choices are time expansion and time series representation. In this section, we compare the two temporal network frameworks by providing an analytical evaluation of the memory requirements.

Time Expanded Network:

Let G be the original graph that represents the evacuation network.

The size of the time expanded network $G_T = (N_T, A_T)$ would be as follows:

Number of nodes $|N_T| = (T + 1)n$, where n is the number of nodes in the original network.

Number of edges $|A_T| = T(m + d)$, where m is the number of edges in G, and d is the number of destination nodes.

According to the analysis in [28], the minimum memory requirement (number of bytes) for the

time expanded network is at least $(52+36T)n+(20+12T)m$. Since T is always a large number (at least hundreds) in real evacuation scenarios, it can be simplified as $T(36n+12m)$.

Time Series Representation:
Let graph G be the evacuation network, n the number of nodes in G, and m the number of edges in G. Instead of the time expanded network G_T used in time expansion, the time series representation used by the CCRP_06 algorithm needs to work only with the original network G. CCRP_06 needs data structures to store a network with n node and m edges. In our implementation, the number of bytes used to store the network is $8n+12m$.

In addition, the time series representation incorporates capacity constraints by building a time series for each node and each edge to keep track of the available node capacity and edge capacity at each time instant during the evacuation. In our implementation, the number of bytes used for the time series is $4tn+4tm$, where t is the evacuation egress time.

Therefore, the memory requirement (number of bytes) for the CCRP_06 algorithm is $(8+4t)n+(12+4t)m$. As T is always a large number (at least hundreds) in real evacuation scenarios, we can simplify it as $4t(n+m)$.

Lemma 2: The time series representation used in CCRP_06 requires less memory than the time expanded network used in LP algorithm if $t<3T$.

Proof: The number of bytes required by the time expanded network and time series representation are $T(36n+12m)$ and $4t(n+m)$. Therefore, the time series representation used in CCRP_06 requires less memory than the time expanded network used in the LP algorithm if $t<3T$. Our experiments show that evacuation time t produced by the CCRP_06 algorithm is within ten percent of the optimal evacuation time (see details in Section IV-A.3), which means t is within five percent larger than T. Therefore, the condition is almost always true.

C. Choice of Algorithm to Identify Closest Source-Destination Pair

The most critical step in the heuristic algorithm presented in Section II-C is the computation of the shortest paths between all source-destination pairs. This step is key in determining the route each group would be assigned to minimize the evacuation time. Since there are various ways to formulate the ”closest pair problem”, there is a need to evaluate the performance with respect to the choices listed in Section 2.4.

TABLE II

COMPARISON OF COMPUTATIONAL COSTS (n: NUMBER OF NODES, p: NUMBER OF EVACUEES, T: USER-PROVIDED UPPER-BOUND ON EVACUATION TIME, C: MAXIMUM EDGE WEIGHT); SOURCE [30]

Algorithm	**Computational Cost**
CCRP_06	$O(p \cdot (m + 2Cn))$
CCRP_05	$O(p \cdot n^2 logn)$
Linear Programming Approach	at least $O((T \cdot n)^6)$

Table II provides a comparison of the LP based approach and the heuristic algorithm with k shortest path computations and single shortest path computation. CCRP_05 is the version of the heuristic algorithm which runs the shortest path algorithm multiple times and CCRP_06 runs the shortest path algorithm just once in an iteration. As can be seen, the LP-based approach produces optimal solutions but suffers from high computational cost. Both versions of the heuristic algorithm reduce the computation cost.

Lemma 3: CCRP_06 is strictly faster than CCRP_05.

Proof: CCRP_06 runs in $O(p \cdot (m + 2Cn))$ time and CCRP_05 runs in $O(p \cdot n^2 logn)$ time (II. Transportation networks are sparse and the number of edges(m) is generally a linear factor of the number of nodes(n) (usually $m \approx 3n$). Therefore, it is easy to see that CCRP_06 is strictly faster than CCRP_05.

D. Shortest Path Algorithms/Data Structures

Valuable insight into the performance of the candidate algorithms is provided by Table III, which lists the asymptotic complexities of the algorithms when used in conjunction with various data structures. The implementations of Dijkstra's algorithm using various data structures have the best asymptotic complexities. In the double bucket implementation of the algorithm, if the bucket size (Δ) is set to the biggest power of two less than $\sqrt{C}$, where C is the maximum edge weight, the time complexity of this implementation would be $O(m + 2Cn)$. In a transportation network, since the edge weight represents the travel time, C is small (of the order of tens of units). Since in a metropolitan sized network the factor $n \ln n$ would be larger than $2Cn$, it can be concluded that the double bucket implementation of the Dijkstra's algorithm would perform better compared to the other implementations. Despite a worse asymptotic performance

TABLE III

ASYMPTOTIC COMPLEXITIES (n: NUMBER OF NODES, m: NUMBER OF EDGES, Δ: BUCKET SIZE, C: MAXIMUM EDGE WEIGHT) ; SOURCE [13]

Algorithm	**Dijkstra-binary-heap**	**Dijkstra-Fibonacci heap**	**Dijkstra-Double-bucket**	**Two_Q**
Asymptotic Complexity	$O(m \log n)$	$O(m + n \log n)$	$O(m + n(\Delta + C/\Delta))$	$O(mn^2)$

compared to Dijkstra's algorithm, the Two_Q algorithm has the potential of performing better if the closest destination node is sufficiently far away from the source node [13]. Since the shortest path distance is not known in advance in a transportation network, the Two_Q algorithm also qualifies to be a candidate algorithm.

E. Solution Quality of CCRP

Since CCRP is a heuristic algorithm, it does not produce optimal solutions for all evacuation scenarios. Experiments show that the evacuation time produced by CCRP is slightly (within 10%) longer than the optimal evacuation time in all test cases (detailed results given in Section IV-A.3).

However, it can be shown that, under certain conditions, CCRP can produce optimal solutions. We define the bottleneck capacity of the evacuation network as the number of evacuees which can travel simultaneously using shortest paths without any wait during their entire travel between respective source-destination pairs. For example, a trivial though very loose lower bound on bottleneck capacity is the minimum of the maximum edge capacity and maximum node capacity.

Lemma 4: CCRP produces an optimal solution if the number of evacuees is less than or equal to the bottleneck capacity of the network.

Proof: It is easy to see that when the total number of evacuees is no more than the bottleneck capacity of the network, there will be no wait time for any evacuees traveling along a route because waiting only occurs when the number of evacuees need to use a route is greater than the maximum node or edge capacity on this route. In this case, the evacuees from each source can be sent through the quickest route to a destination without any delay on the route. This means that the problem is reduced to first finding the shortest path from each source node to any destination node and then sending all evacuees from each source node as one group to a destination using the shortest path found. In this case, all the routes used are the shortest

path from each source to a destination and there is no delay along the routes. Therefore, the evacuation plan found must be the optimal plan.

IV. Experimental Evaluation of CCRP_06 Design Decisions

Performance evaluation of CCRP_06 design decisions consisted of the following tasks: 1) Compare the algorithm run-time and solution quality of the CCRP_06 algorithm and the LP based algorithm, 2) Compare two versions of the the CCRP algorithms, namely CCRP with multiple shortest search (CCRP_05) and CCRP with single shortest path search (CCRP_06) and 3) Compare the performance of different implementations of CCRP_06 using different shortest path search algorithms.

A. Comparison of CCRP_06 and Linear Programming Approach

The purpose of this section is to compare the performance of the heuristic CCRP_06 algorithm, with the optimal LP based algorithm. The linear programming software used in this experiment was RelaxIV [8], which is widely considered as one of the fastest minimum cost flow solvers. The experiment was done by comparing the algorithm run-time of CCRP_06 and RelaxIV by using various network configurations. It should be noted that the CCRP_06 algorithm used in this experiment was implemented with Dijkstra's shortest path algorithm using the double-bucket data structure. The reason for choosing Dijkstra's algorithm with double-bucket is that experiments show that it results in best CCRP_06 performance among available shortest path algorithms. We present a detailed analysis of this choice in Section IV-B.2.

1) Experiment Design: Figure 2 illustrates the experiment design to compare the performance of CCRP_06 and RelaxIV. First, NETGEN [29] was used to generate evacuation networks with capacity constraints and evacuees. NETGEN is a software that generates transportation networks with capacity constraints and initial supplies based on a set of input parameters. In our experiments, the following four were selected as independent parameters to test their impacts on the performance of the algorithms: 1) network size represented by number of nodes; 2) number of evacuees initially in the network; 3) number of source nodes; and 4) number of destination nodes. Number of edges is treated as a dependent parameter. We set the number of edges to be equal to 3 times the number of nodes because the typical edge/node ratio for real transportation road networks is around 3. Next, the evacuation network generated by NETGEN was fed to the CCRP_06. Before feeding the network to RelaxIV, we needed to use a network converter to

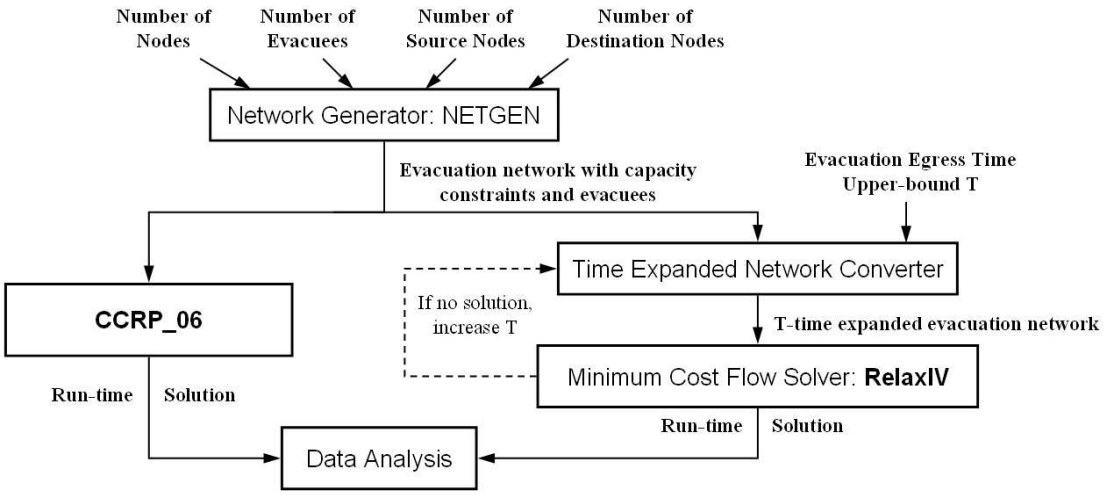


Fig. 2. Experiment Design

transform the evacuation network into a time-expanded network, which is required by minimum cost flow solvers (such as RelaxIV) to solve evacuation problems [12], [20]. This conversion requires an input parameter T, which is an estimated upper-bound on the optimal evacuation egress time. If the evacuation cannot be completed by time T, RelaxIV will return no solution. In this case, T needs to be increased to create a new time-expanded network and to run RelaxIV again until a solution can be reached. In the experiments, we avoided under-estimation of T by setting T equal to the egress time produced by CCRP_06. Since CCRP_06 is a heuristic algorithm, its evacuation egress time can be used as an upper-bound of the optimal solution. After CCRP_06 and RelaxIV produced solutions for each test case, the algorithm run-times were collected and analyzed in the data analysis module. This same experiment design was also used to evaluate the solution quality of CCRP_06; we present the results and analysis in Section IV-A.3.

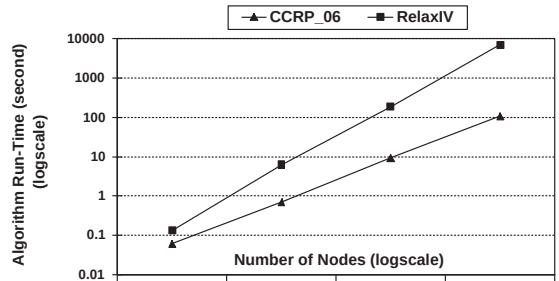
The experiments were conducted on a workstation with Intel Pentium 4 2.8GHz CPU, 2GB RAM and Linux operating system. Each experimental result reported in the following sections is the average over 5 experiment runs with networks generated using the same input parameters.

2) Experiment Results for Algorithm Run-time: We wanted to answer four questions: (1) Are the algorithms scalable to the size of the network, particularly will they handle large size transportation networks as in urban evacuation scenarios? (2) How does the number of evacuees

affect the performance of the algorithms? (3) How does the number of source nodes affect the performance of the algorithms? (4) How does the number of destination nodes affect the performance of the algorithms?

a) Experiment 1: Are the algorithms scalable to the size of the network?

In this experiment, we evaluated how the network size affects the performance of the algorithms. We fixed the other three independent parameters and varied the network size to observe the run-time of the algorithms. The experiment was done using networks with 5000 evacuees, 20 source nodes, and 10 destination nodes. We varied the number of nodes in the work from 50 to 50000. Figure 3 shows the run-times of the two algorithms with an accompanying data table. Both the



	50	500	5000	50000
CCRP_06	0.06	0.69	9.38	108.16
RelaxIV	0.13	6.39	189.51	6971.05

Fig. 3. Run-time (**log-scale**)With Respect to Network Size. (Note: Both x-axis and y-axis are in logarithmic scale.)

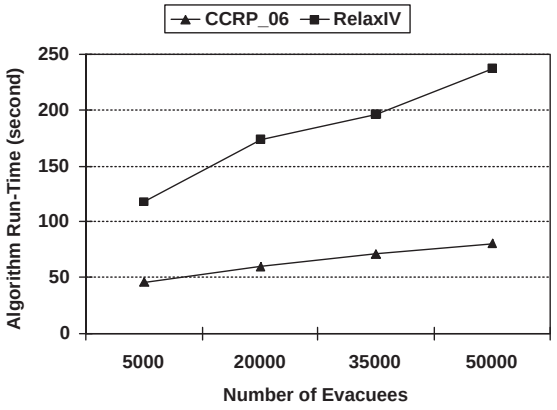


Fig. 4. Run-time With Respect to Number of Evacuees

x-axis(number of nodes) and y-axis(run-time) of Figure 3 are on a logarithmic scale rather than linear. It can be seen that the CCRP_06 algorithm runs in time that is proportional to a small polynomial in the size of the network while the run-time of RelaxIV grows much faster. This shows that CCRP_06 is much more computationally efficient than LP RelaxIV. This experiment also shows that the run-time of CCRP_06 is scalable to the size of the network.

b) Experiment 2: How does the number of evacuees affect the performance of the algorithms?

The purpose of this experiment was to evaluate how the number of evacuees affects the performance of the algorithms. We fixed the other independent parameters and varied the number of evacuees to observe the algorithm run-time of CCRP_06 and RelaxIV.

The experiment was done using networks with 5000 nodes, 2000 source nodes, and 10

destination nodes. We varied the number of evacuees from 5000 to 50000. Figure 4 shows the run-times of the two algorithms. As can be seen, in each test case, the run-time of CCRP_06 remains less than half that of RelaxIV. In addition, the CCRP_06 run-time is scalable to the number of evacuees while the run-time of RelaxIV grows much faster. This experiment shows: (1) CCRP_06gives much less run-time than that of RelaxIV. (2) The run-time of CCRP_06 is scalable to the number of evacuees.

c) Experiment 3: How does the number of source nodes affect the performance of the algorithms?

In this experiment, we evaluated how the number of source nodes affects the performance of the algorithms. We fixed the other three independent parameters and varied the number of source nodes to observe the algorithm run-time. In this experiment setup, by varying the number of source nodes, we actually create different evacuee distributions in the network. A higher number of source nodes means that the evacuees are more scattered in the network.

The experiment was done using networks with 5000 nodes, 5000 evacuees, and 10 destination nodes. We varied the number of source nodes from 1000 to 4000. As shown in Figure 5, the run-times of both algorithms are scalable to the number of source nodes. However, in all test cases, the run-time of CCRP_06 remains less than half of the run-time of RelaxIV. This experiment also shows that the run-time of CCRP_06 is scalable to the number of source nodes.

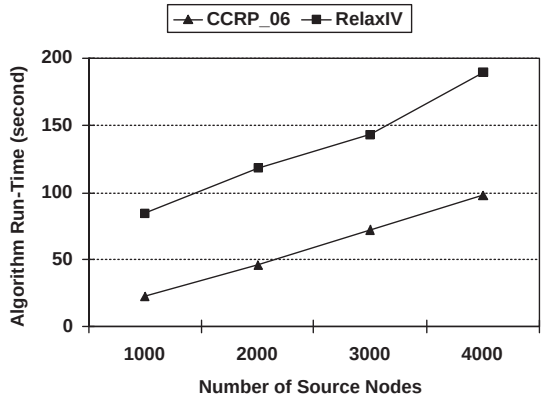


Fig. 5. Run-time With Respect to Number of Source Nodes

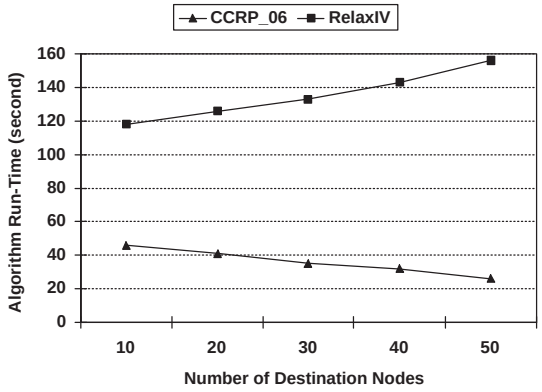


Fig. 6. Run-time With Respect to Number of Destination Nodes

d) Experiment 4: How does the number of destination nodes affect the performance of the

algorithms?

In this experiment, we evaluated how the number of destination nodes affects the performance of the algorithms. We fixed the other three independent parameters and varied the number of destination nodes to observe the algorithm run-time. The experiment was done using networks with 5000 nodes, 5000 evacuees, and 2000 source nodes. We varied the number of destination nodes from 10 to 50. Figure 6 shows the run-times of the two algorithms.

As can be seen, the run-time of the CCRP_06 algorithms actually decreases as the number of destination nodes grows, while the run-time of RelaxIV increases. This is due to the fact that CCRP_06 uses shortest path searches in each iteration to find the quickest route from any source node to any destination node and we implemented the shortest path search with Dijkstra's algorithm [16]. It is known that Dijkstra's algorithm finds the shortest path from the source to any node as soon as the node is permanently labeled [14]. In CCRP_06, this means that the quickest route is found as soon as any destination node is reached and Dijkstra's algorithm can terminate. This property enables the CCRP_06 algorithm to take advantage of more destination nodes because more destinations result in less time for Dijkstra's algorithm to reach a destination node and hence it reduces the CCRP_06 run-time when the number of destination nodes increases. By contrast, the RelaxIV algorithm, which does not uses Dijkstra's algorithm, cannot take advantage of more destination nodes. As Figure 6 shows, more destination nodes make the problem harder for RelaxIV to solve because its run-time actually increase as the number of destination nodes grows.

This experiment shows that the run-time of CCRP_06 decreases as the number of destination nodes grows, while the run-time of RelaxIV increases. The CCRP_06 algorithm has a clear advantage over RelaxIV on algorithm run-time when there is need to add more destination nodes to an evacuation scenario.

3) Experiment Results for Quality of Solution: In this experiment, we used the same experiment design as shown in Figure 2 After CCRP_06 and RelaxIV produced solutions for each test case, the solution quality of the two algorithm were collected and analyzed in the data analysis module.

We wanted to compare the solution quality of the CCRP_06, which is a heuristic algorithm, with that of the RelaxIV, which produces optimal solutions. We conduct the comparison by examining how the following four parameters affect the solution quality of CCRP_06: (1) network

size represented by number of nodes in the network; (2) number of evacuees; (3) number of source nodes; and (4) number of destination nodes.

a) Experiment 1: How does the network size affect the the solution quality of CCRP_06?

In this experiment, we evaluated how the network size affects the performance of the algorithms. We fixed the other three independent parameters and varied the network size to observe the quality of solutions.

The experiment was done using networks 5000 evacuees, 20 source nodes, and 10 destination nodes. We varied the number of nodes in the work from 50 to 50000. Figure 7 shows the solution quality represented by evacuation egress time.

In each of the test cases, CCRP_06 produced high quality solutions (within 10 percent longer than optimal evacuation time) and the solution quality of CCRP becomes very close to the optimal solution produced by RelaxIV as the network size increases. This means CCRP_06 can produce close-to-optimals solutions for large size networks.

This experiment shows: (1) The solution quality of CCRP_06 increases as the network size grows, (2) CCRP_06 produces close-to-optimal solution for large size networks (e.g. network with more than 5000 nodes).

These findings indicate that CCRP_06 has an advantage over the RelaxIV algorithm when producing plans for urban evacuation scenarios where the road network is complex. In these cases, CCRP_06 can provide high quality solutions with much less running time than the optimal solution algorithm as we showed in the previous experiments. More importantly, the findings suggest that it is often not necessary to obtain the optimal plan in a real evacuation scenario. Instead, it is critical to be able to produce a number of high quality plans efficiently so that officials can revise the plan based on the changing situation and make decisions in a timely manner.

b) Experiment 2: How does the number of evacuees affect solution quality of CCRP_06

In this experiment, we fixed the other independent parameters and varied the number of evacuees to observe the quality of the solution and the run-time of CCRP_06 and RelaxIV.

The experiment was done using networks with 5000 nodes, 2000 source nodes, and 10 destination nodes. We varied the number of evacuees from 5000 to 50000. Figure 8 shows the solution quality represented by evacuation egress time. One exception is that the data point with 50 evacuees has only 25 source nodes, we included this setup in order to test whether

CCRP can an produce optimal solution when the number of evacuees is no greater than the bottleneck capacity (50 in this test case) of the network, as stated in Lemma 4.

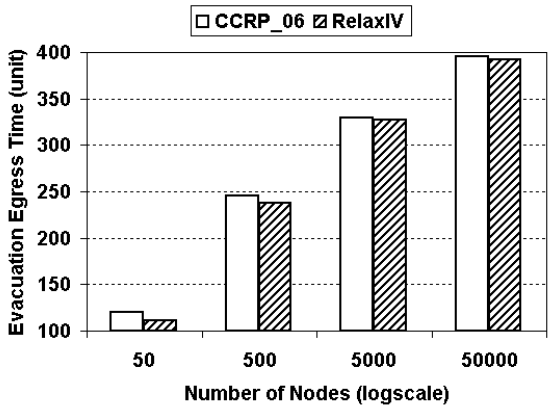


Fig. 7. Quality of Solution With Respect to Network Size (unit for y-axis is the same as input unit for travel time,typically in minutes)

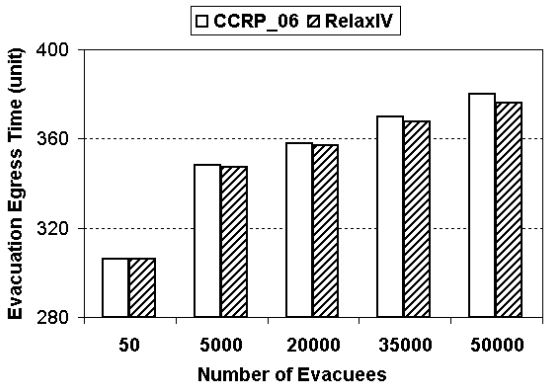


Fig. 8. Quality of Solution With Respect to Number of Evacuees (unit for y-axis is the same as input unit for travel time,typically in minutes)

The experiment results show that: 1) In each test case, CCRP_06 produced very high quality solutions compared with the optimal solutions produced by RelaxIV. 2) At the data point with 50 evacuees, CCRP produced the same evacuation time (306 time units)as RelaxIV produced. In this test case, the number of evacuees is less than the bottleneck capacity of the network. Therefore, CCRP produces the optimal solution as we stated in Lemma 4. Its solution quality does drop slightly though, as the the number of evacuees grows.

c)Experiment 3: How does the number of source nodes affect the solution quality of CCRP_06? In this experiment, we evaluated how the number of source nodes affects the solution quality of the algorithms. We fixed the other three independent parameters and varied the number of source nodes to observe the quality of the solution. In this experiment setup, by varying the number of source nodes, we actually create different evacuee distributions in the network. A higher number of source nodes means that the evacuees are more scattered in the network.

The experiment was done using networks with 5000 nodes, 5000 evacuees, and 10 destination nodes. We varied the number of source nodes from 1000 to 4000. Figure 9 shows the solution quality represented by evacuation egress time.

In all test cases, CCRP_06 produced high quality solutions (within 5 percent longer than the optimal evacuation time) and the number of source nodes has little effect on the solution quality

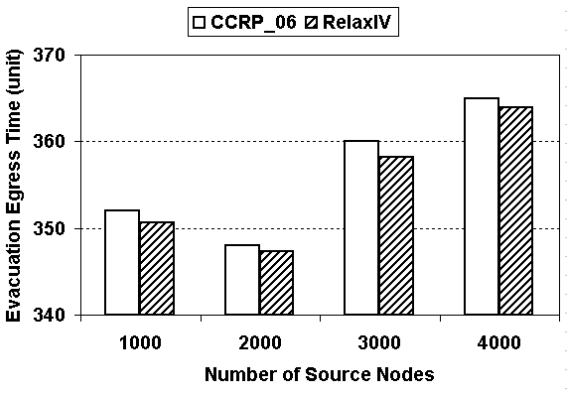


Fig. 9. Quality of Solution With Respect to Number of Source Nodes (unit for y-axis is the same as input unit for travel time,typically in minutes)

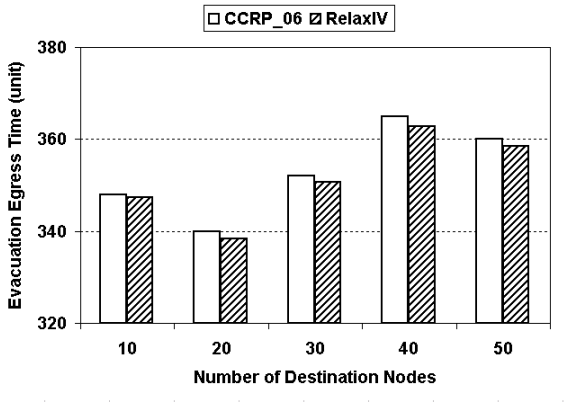


Fig. 10. Quality of Solution With Respect to Number of Destination Nodes (unit for y-axis is the same as input unit for travel time,typically in minutes)

of CCRP_06. It is also interesting to note that the evacuation egress time is non-monotonic with respect to the number of source nodes. This means that when the number of evacuees is fixed, adding more source nodes does not necessarily increase or decrease the evacuation egress time. In this case, the location of the newly added source nodes have much more impact on the evacuation time. For example, adding source nodes closer to the destinations will likely decrease the evacuation time, while adding source nodes further away from the destinations will likely increase the evacuation time.

This experiment shows: (1) CCRP_06 produces high quality solutions in all test cases. (2)The solution quality of CCRP_06 is not affected by the number of source nodes.

d) Experiment 4: How does the number of destination nodes affect the the solution quality of CCRP_06?

In this experiment, we evaluated how the number of destination nodes affects the solution quality of the algorithms. We fixed the other three independent parameters and varied the number of destination nodes to observe the quality of the solution.

The experiment was done using networks with 5000 nodes, 5000 evacuees, and 2000 source nodes. We varied the number of destination nodes from 10 to 50. Figure 10 shows the solution quality represented by evacuation egress time.

In all test cases, CCRP_06 produced high quality solutions (within 5 percent longer than optimal evacuation time) and the number of destination nodes has little effect on the solution

quality of CCRP_06. Similar to the previous experiment on the number of source nodes, it is also noted that the evacuation egress time is non-monotonic with respect to the number of destination nodes. This means that adding more destination nodes to an evacuation scenario does not necessarily reduce the evacuation egress time. Instead, the location of the added destination nodes and the capacity of the roads leading to these nodes may play a much more important role.

This experiment shows: (1) CCRP_06 produces high quality solutions in all test cases. (2)The solution quality of CCRP_06 is not affected by the number of destination nodes.

B. CCRP Design Decisions

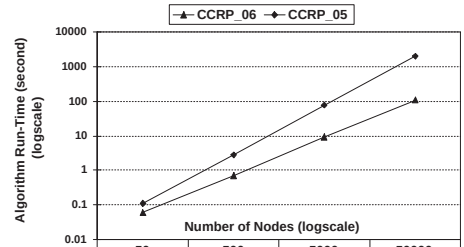
In this section, we present the experimental evaluation of two design decisions to improve the performance of the CCRP algorithm.

1) Choice of Algorithm to Identify Closest Source-Destination Pair: CCRP_05 is an earlier algorithm based on the capacity constrained routing approach. Major improvements in the new version CCRP_06, lie in the algorithm used to find the quickest route between the closest source-destination pair. In CCRP_05, finding quickest route R is done by running one generalized shortest path search from each source node to all destination nodes. Each search is terminated when any destination node is reached. If there are x source nodes in the network, CCRP_05 algorithm requires x shortest path searches (one per source node) to be done in each iteration in order to find route R.

In CCRP_06, one important design decision was made to improve the step of finding route R. The improvement is to replace the x shortest path searches in CCRP_06 with only one shortest path search. This was done by adding a super source node s_0 to the network and connecting s_0 to all source nodes. The super source s_0 is connected to each source node by an edge with infinite capacity and zero travel time. This allows us to complete the search for route R by using only one single shortest path search, which takes the super source s_0 as the start node. The search terminates when any destination node is reached. It can be easily proved that the shortest route found by this search is the route R we need in line 2. This improvement significantly reduces the computational cost of the algorithm by one degree of magnitude compared with CCRP_05.

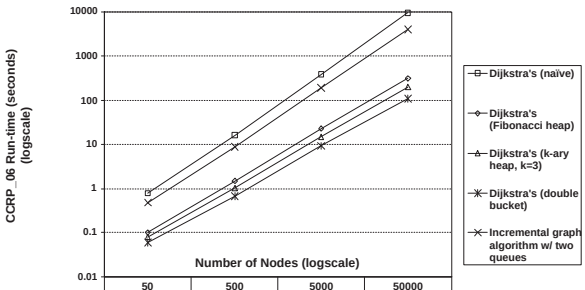
Since CCRP_05 and CCRP_06 use the same heuristic method to find a solution, it is expected that CCRP_05 and CCRP_06 would produce solutions with the same evacuation egress time for

each test case. To observe the difference between the actual run-time of CCRP_05 and CCRP_06, we conducted the following experiment. NETGEN was used to generate evacuation networks with 5000 evacuees, 20 source nodes, 10 destination nodes, and number of nodes varying from 50 to 50,000. It should be noted again that the CCRP_05 and CCRP_06 algorithms used in this experiment were implemented with Dijkstra's shortest path algorithm using double-bucket data structure. The reasons for this decision are presented in Section IV-B.2. Figure 11 shows the run-times of CCRP_05 and CCRP_06 with respect to different network sizes, with an accompanying data table.



	50	500	5000	50000
CCRP_06	0.06	0.69	9.38	108.16
CCRP_05	0.11	2.85	77.43	1971.84

Fig. 11. Run-time **(log-scale)** With Respect to Network Size. (Note: Both x-axis and y-axis are in logarithmic scale.)



	50	500	5000	50000
Dijkstra's (naïve)	0.8	16.26	382.84	9532.8
Dijkstra's (Fibonacci heap)	0.1	1.5	23.1	316.44
Dijkstra's (k-ary heap, k=3)	0.08	1.06	15.14	200.49
Dijkstra's (double bucket)	0.06	0.69	9.38	108.16
Incremental graph algorithm w/ two queues	0.49	9.02	190.73	3983.69

Fig. 12. CCRP_06 Run-time (**log-scale**) With Respect to Network Size. (Note: Both x-axis and y-axis are in logarithmic scale.)

Both the x-axis(number of nodes) and y-axis(run-time) of Figure 11 are on a logarithmic scale. It can be seen that, in all test cases, CCRP_06 run-time was much faster than that of CCRP_05. For small networks with 50 nodes, CCRP_06 out-performed CCRP_05 by a factor of about 2 and this factor became more significant as the network size increases. For large networks with 50,000 nodes, CCRP_06 was faster than CCRP_05 by a factor of 10.

This experiment shows that, compared to CCRP_05, CCRP_06 significantly improves the performance of the capacity constrained routing algorithm, especially for evacuation scenarios with large size networks.

2) Comparison of different implementations of the CCRP_06 algorithm: Another important design decision for CCRP_06 is the choice of shortest path algorithm used to find the quickest route R. Shortest path algorithms and their implementations have been developed and evaluated

extensively [6], [13], [40]. Many of these algorithms can reduce the computational cost by taking advantage of certain properties of the graph network. We chose to look specifically at Dijkstra's algorithm and the Two-Q based on the following reasoning.

Evacuation networks have a few important properties. First, most evacuation networks are transportation road networks; as such they are sparse networks because most road networks have an edge/node ratio that is less than 3. Second, in our problem formulation, we defined the travel time of edges as non-negative integers, which means the network has non-negative and integral edge weights.

Many shortest path algorithms have proved to be able to reduce computational cost with networks of such properties. One of the most comprehensive reviews of shortest path algorithms was done by Cherkassky, Goldberg, and Radzik [13]. Cherkassky et al. [13] suggested that Dijkstra's algorithm has the best performance for networks with non-negative edge weights. Among the various implementations of Dijkstra's algorithm, Dijkstra's using binary heap and Dijkstra's using double bucket gave better performance for sparse networks. In addition, it has been shown that Two-Q algorithm [34] also performed well on some problems with road networks [40].

In order to test the performance of the CCRP_06 algorithm with different shortest path algorithms, we chose the following four algorithms as candidates to implement the shortest path search in CCRP_06: incremental graph algorithm with two queues, Dijkstra's using binary heap, Dijkstra's using double bucket, and Dijkstra's using Fibonacci heaps. Dijkstra's algorithm using Fibonacci heaps was chosen because it has the best theoretical worst case complexity on sparse graphs [13] and we wanted to see how its actual performance compare with others.

In this experiment, NETGEN was used to generate evacuation networks with 5000 evacuees, 20 source nodes, 10 destination nodes, and number of nodes varying from 50 to 50,000. The purpose was to compare the performance of CCRP_06 with each implementation on evacuation networks with different sizes. Figure 12 shows the run-times of the candidate algorithms with respect to different network sizes. Dijkstra's algorithm with naive implementation, which is known to perform poorly, was added in the experiment as a reference. As can be seen, the three implementations of Dijkstra's algorithms (Dijkstra's using binary heap, Dijkstra's using double bucket and Dijkstra's using Fibonacci heaps.) gave much better performance than Two-Q algorithm. Among the three, Dijkstra's using double bucket performed the best mainly because

it is known to be able to take advantage of non-negative integral edge weights. By contrast, Dijkstra's algorithm using Fibonacci heaps was the slowest since it does not take advantage of these network properties. This result also means that an algorithm with the best theoretical computational cost (such as Dijkstra's algorithm using Fibonacci heaps) does not necessarily give the best performance.

The Two-Q algorithm performed very poorly, which is only faster than Dijkstra's algorithm with naive implementation. Previously, Two-Q algorithm was shown to perform well on some road networks problems with one-to-all shortest path search [40]. However, the shortest path search in the CCRP_06 algorithm is a one-to-some shortest path search because it only needs to find the best route from the source to any one of the destination nodes. The Two-Q algorithm cannot take advantage of this because it has to complete the search to all nodes before it terminates. By contrast, Dijkstra's algorithm can terminate as soon as one of the destination nodes is reached. This is the main reason that Two-Q algorithm performed slower than all the three candidates of Dijkstra's algorithm in the experiment.

Overall, this experiment shows that Dijkstra's algorithm with double bucket implementation gives the best performance among all candidates. Therefore, Dijkstra's algorithm with double bucket is our choice for the design decision of implementing the shortest path search in CCRP.

C. A Case Study

In this section we report the results of experiments conducted on a real evacuation scenario. As shown in Figure 13, the Monticello nuclear power plant is about 40 miles to the northwest of the Twin Cities of Minneapolis-St.Paul. Evacuation plans need to be in place in case of accidents or terrorist attacks. The evacuation zone is a 10-mile radius around the nuclear power plant as defined by Minnesota Homeland Security and Emergency Management [3]. A hand-drafted evacuation route plan was developed to evacuate the affected population to a high school. However, this plan did not consider the capacity of the road networks and put high loads on two highways.

We conducted an experiment using the CCRP algorithm. The experiment was done using the road network around the evacuation zone provided by the Minnesota Department of Transportation [2], and the Census 2000 population data for each affected city (circles in Figure 13). The total number of evacuees is about 42,000. As can be seen in Figure 13, our algorithm

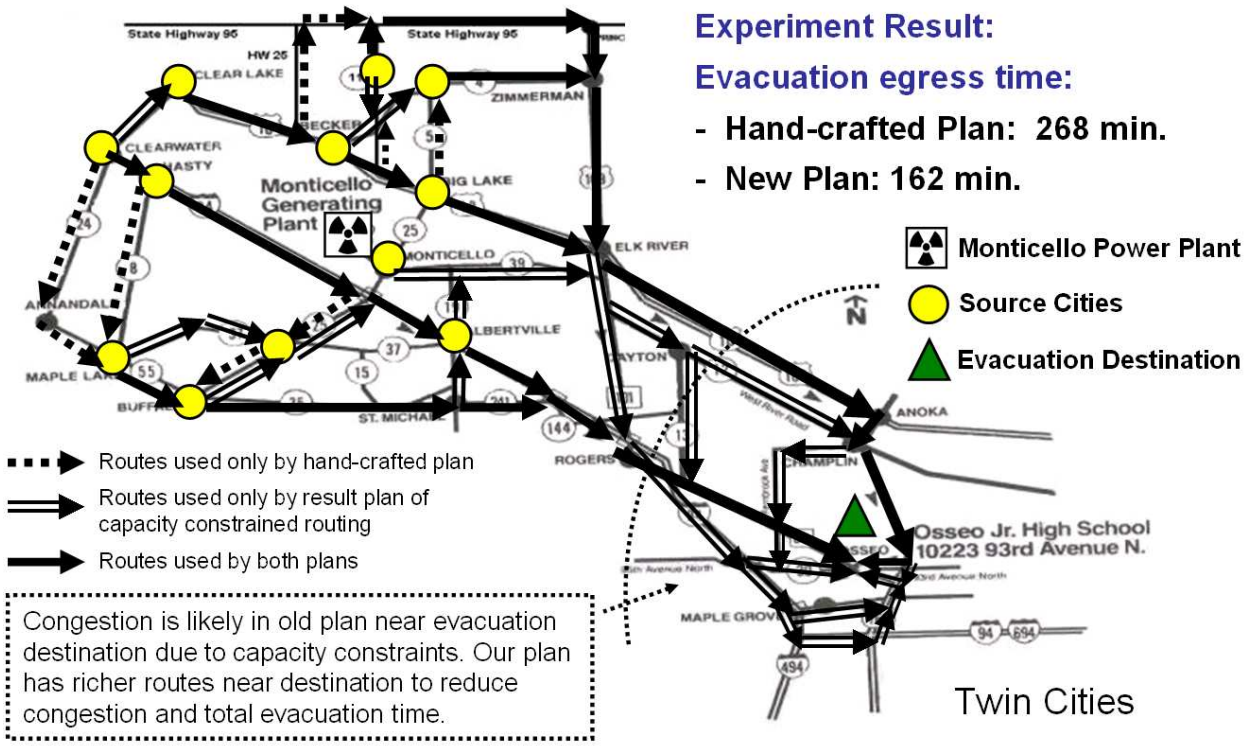


Fig. 13. Overlay of Result Routes for Monticello Power Plant Evacuation Route Planning

gives a much better evacuation route plan by selecting shorter paths to reduce evacuation time and utilizing richer routes (routes near evacuation destination) to reduce congestion. The old evacuation plan has an evacuation egress time of 268 minutes. The CCRP algorithm produced a much better plan with an evacuation time of only 162 minutes. This experiment shows that our algorithm is effective in real evacuation scenarios to reduce evacuation time and improve existing plans.

Our approach was presented at the Congressional Breakfast Program on Homeland Security [37] held by the University Consortium for Geographic Information Science (UCGIS), and also reported in the Minnesota Homeland Security and Emergency Management newsletter [39]. It was also selected by the Minnesota Department of Transportation to be used in the evacuation planning project for the Twin Cities Metro Area, which involves a road network of about 250,000 nodes and a population of over 2 million people. In this project, the CCRP algorithm was tested on five pre-defined scenarios and some randomly selected locations. Transportation professionals evaluated the quality of the solutions and found them to be highly satisfactory. An article in

St. Paul Pioneer Press [5] discussed some salient features of this project. The project also won the Research Partnership Award from the Center for Transportation Studies(CTS) [4] as a recognition for making significant impacts on transportation.

V. Conclusions and Future Work

Prior approaches to evacuation route planning relied on LP based methods to generate optimal evacuation plans. These methods suffer from high computational cost and memory requirement. We addressed the need for a computationally efficient approach in [30], by proposing the CCRP_06 algorithm. CCRP_06 is a heuristic algorithm that uses time series to incorporate capacity constraints and generalizes shortest path search algorithms. This algorithm produces high-quality solutions and is scalable to large evacuation networks.

In this paper, we present a comprehensive overview of the algorithm framework for the evacuation route planning problem and propose new approaches to address the limitation of previous studies. We propose an improved heuristic algorithm (CCRP_06) by exploring available design decisions. We characterize the design space available in the context of the CCRP_06 algorithm and evaluate the performance of the CCRP_06 algorithm for each of the design decisions. A wide range of shortest path algorithms and data structures are explored and experiment results show that Dijkstra's algorithm with double-bucket data structure gives the best performance for CCRP_06. We prove that CCRP_06, which uses Dijkstra's algorithm with double-bucket, has a run-time of $O(p \cdot (m + 2Cn))$, which is faster than LP based methods in real evacuation scenarios. We also prove that CCRP_06 requires less memory than the LP algorithm. Experimental evaluation using various network configurations show that CCRP_06 produces high quality solutions and is much more computationally efficient than LP algorithms. It is also shown that CCRP_06 has a clear advantage over the LP algorithm when increasing the number of destination nodes in the network.

The shortest path algorithm used in our approach assumes that the edge travel times include traffic delays at intersections. It also assumes that the travel times are not time-dependent. We plan to incorporate existing work in this area, such as [41], to address this limitation.

Another interesting possibility for future work is to integrate our CCRP approach with the traffic assignment-simulation approach. The traffic assignment-simulation approach uses traffic simulation tools, such as DYNASMART [32] and DynaMIT [7], to conduct stochastic simulation

of traffic movements based on origin-destination traffic demands and uses queuing methods to account for road capacity constraints. Although it may take a long time to complete the simulation process for a large transportation network, this approach does have the capability to predict locations for traffic congestion, in contrast to CCRP, which assumes that traffic moves at a certain speed on each road segment.

ACKNOWLEDGMENT

We are particularly grateful to members of the Spatial Database Research Group at the University of Minnesota for their helpful comments and valuable discussions. We would like to thank Boris V. Cherkassky, Andrew V. Goldberg and Tomasz Radzik for making their shortest path search C source codes (SPLIB) available and Antonio Frangioni for providing the RelaxIV code. We are very thankful to Daryl Taavola of URS Corporation and Sonia Pitt of Minnesota Department of Transportation for their valuable help and guidance during the implementation of of Minneapolis Metro Evacuation Project. We would also like to express our thanks to Kim Koffolt for improving the readability of this paper.

This work is supported by the Army High Performance Computing Research Center (AHPCRC) under the auspices of the Department of the Army, Army Research Laboratory under contract number DAAD19-01-2-0014 and the Minnesota Department of Transportation under contract number 81655. The content does not necessarily reflect the position or policy of the government and no official endorsement should be inferred. AHPCRC and the Minnesota Supercomputer Institute provided access to computing facilities.

REFERENCES

[1] *Hurricane Evacuation web page*. http://i49south.com/hurricane.htm, 2002.
[2] *Minnesota basemap web site*. http://www.dot.state.mn.us/tda/basemap/, Minnesota Department of Transportation, 2004.
[3] *Monticello evacuation planning web site*. http://www.hsem.state.mn.us/, Minnesota Homeland Security and Emergency Management, 2004.
[4] *"CTS Awards-2006"*. http://www.cts.umn.edu/awards/researchpartnership/, 2006.
[5] *"Walk, Don't Drive to Safety"*. http://www.twincities.com/mld/twincities/news/state/minnesota/14051739.htm, 2006.
[6] R. Ahuja, T. Magnanti, and J. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice Hall, 1993.
[7] M. Ben-Akiva et al. *Development of a Deployable Real-Time Dynamic Traffic Assignment System: DynaMIT and DynaMIT-P User's Guide*. Intelligent Transportation Systems Program, Massachusetts Institute of Technology, 2002.
[8] D. Bertsekas and P. Tseng. RELAX-IV: A Faster Version of the RELAX Code for Solving Minimum Cost Flow Problems. 2004.
[9] D. Bertsimas and J. Tsitsiklis. *Introduction to Linear Optimization* . Athena Scientific, 1997.
[10] S. Brown. Building America's Anti-Terror Machine: How Infotech Can Combat Homeland Insecurity. *Fortune*, pages 99–104, July 2002.
[11] T. V. N. T. S. Center. Improving Regional Transportation Planning for Catastrophic Events(FHWA). *Volpe Center Highlights*, pages 1–3, July/August 2002.
[12] L. Chalmet, R. Francis, and P. Saunders. Network Model for Building Evacuation. *Management Science*, 28:86–105, 1982.
[13] B. Cherkassky, A. Goldberg, and T. Radzik. Shortest Paths Algorithms: Theory and Experimental Evaluation . *Mathematical Programming*, 73:129–174, 1996.
[14] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, 2nd edition, 2001.
[15] T. H. S. Council. Planning Scenarios, Executive Summaries, Created for Use in National, Federal, State, and Local Homeland Security Preparedness Activities. July 2004.
[16] E. Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959.
[17] ESRI. GIS for Homeland Security, An ESRI white paper, November 2001.
[18] L. Ford and D. Fulkerson. *Flows in Network*. Princeton University Press, 1962.
[19] R. Francis and L. Chalmet. A Negative Exponential Solution To An Evacuation Problem. *Research Report No.84-86, National Bureau of Standards, Center for Fire Research*, October 1984.
[20] H. Hamacher and S. Tjandra. Mathematical Modeling of Evacuation Problems: A state of the art. *Pedestrian and Evacuation Dynamics*, pages 227–266, 2002.
[21] B. Harden and S. Moreno. *Thousands Fleeing Rita Jam Roads From Coast*. The Washington Post, September 2005.
[22] P. Hart, N. Nilsson, and R. B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of System Science and Cybernetics*, 4(2), 1986.
[23] B. Hoppe and E. Tardos. Polynomial Time Algorithms For Some Evacuation Problems. *Proceedings of the 5th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 433–441, 1994.
[24] B. Hoppe and E. Tardos. The Quickest Transshipment Problem. *Proceedings of the 6th annual ACM-SIAM Symposium on Discrete Algorithms*, pages 512–521, January 1995.
[25] J. Jarvis and H. Ratliff. Some Equivalent Objectives for Dynamic Network Flow Problems. *Management Science*, 28:106–108, 1982.

[26] J. Kennington and R. Helgason. *Algorithm for Network Programming*. Wiley and Sons, 1980.
[27] T. Kisko and R. Francis. Evacnet+: A Computer Program to Determine Optimal Building Evacuation Plans. *Fire Safety Journal*, 9:211–222, 1985.
[28] T. Kisko, R. Francis, and C. Nobel. *EVACNET4 User's Guide*. University of Florida, 1998.
[29] D. Klingman, A. Napier, and J. Stutz. NETGEN: A Program for Generating Large Scale Capacitated Assignment, Transportation, and Minimum Cost Flow Network Problems. *Management Science*, 20:814–821, 1974.
[30] Q. Lu, B. George, and S. Shekhar. Capacity Constrained Routing Algorithms for Evacuation Planning: A Summary of Results. *Advances in Spatial and Temporal Databases, Proceeding of 9th International Symposium on Spatial and Temporal Databases*, August 2005.
[31] Q. Lu, Y. Huang, and S. Shekhar. Evacuation Planning: A Capacity Constrained Routing Approach. *Proceedings of the First NSF/NIJ Symposium on Intelligence and Security Informatics*, pages 111–125, June 2003.
[32] H. Mahmassani, H. Sbayti, and X. Zhou. *DYNASMART-P Version 1.0 User's Guide*. Maryland Transportation Initiative, University of Maryland, September 2004.
[33] N. Nilsson. *Principles of Artificial Intelligence*. Tioga Publishing Co., 1980.
[34] S. Pallottino. Shortest-Path Methods: Complexity, Interrelations and New Propositions . *Networks*, 14:257–267, 1984.
[35] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison Wesley, 1984.
[36] D. R.B. Algorithm 360:Shortest Path Forest with Topological Ordering. *Comm. ACM*, 12:632–633, 1969.
[37] S. Shekhar. Evacuation Planning: Presentation at UCGIS Congressional Breakfast Program on Homeland Security. *http://www.ucgis.org/winter2004/program.htm*, February 2004.
[38] S. Shekhar, A. Fetterer, and B. Goyal. Materialization Trade-offs in Hierarchical Shortest Path Algorithms. *Proc. Intl. Symp. on Large Spatial Databases, Springer Verlag (Lecture Notes in Computer Science)*, 1997.
[39] S. Shekhar and Q. Lu. Evacuation Planning for Homeland Security. *Minnesota Homeland Security and Emergency Management newsletter*, October 2004.
[40] F. Zhan and C. Noon. Shortest Paths Algorithms: An Evaluation Using Real Road Networks. *Transportation Science*, 32:65–73, 1998.
[41] A. Ziliaskopoulos and H. Mahmassani. A Note on Least Time Path Computation Considering Delays and Prohibitations for Intersection Movements. *Transportation Research B*, 30(5):359–367, 1996.