Shelter Location and Evacuation Route Assignment Under Uncertainty: A Benders Decomposition Approach

Vedat Bayram, Hande Yaman

To cite this article:

Published online in Articles in Advance 11 Aug 2017

https://doi.org/10.1287/trsc.2017.0762

Full terms and conditions of use: http://pubsonline.informs.org/page/terms-and-conditions

This article may be used only for the purposes of research, teaching, and/or private study. Commercial use or systematic downloading (by robots or other automatic processes) is prohibited without explicit Publisher approval, unless otherwise noted. For more information, contact permissions@informs.org.

The Publisher does not warrant or guarantee the article’s accuracy, completeness, merchantability, fitness for a particular purpose, or non-infringement. Descriptions of, or references to, products or publications, or inclusion of an advertisement in this article, neither constitutes nor implies a guarantee, endorsement, or support of claims made of that product, publication, or service.

Copyright © 2017, INFORMS

Please scroll down for article—it is on subsequent pages
Shelter Location and Evacuation Route Assignment Under Uncertainty: A Benders Decomposition Approach

Vedat Bayram, Hande Yaman

Abstract. Shelters are safe facilities that protect a population from possible damaging effects of a disaster. For that reason, shelter location and traffic assignment decisions should be considered simultaneously for an efficient evacuation plan. In addition, as it is very difficult to anticipate the exact place, time, and scale of a disaster, one needs to take into account the uncertainty in evacuation demand, the disruption/degradation of evacuation road network structure, and the disruption in shelters. In this study, we propose an exact algorithm based on Benders decomposition to solve a scenario-based two-stage stochastic evacuation planning model that optimally locates shelters and that assigns evacuees to shelters and routes in an efficient and fair way to minimize the expected total evacuation time. The second stage of the model is a second-order cone programming problem, and we use duality results for second-order cone programming in a Benders decomposition setting. We solve practical-size problems with up to 1,000 scenarios in moderate CPU times. We investigate methods such as employing a multicut strategy, deriving Pareto-optimal cuts, and using a preemptive priority multiobjective program to enhance the proposed algorithm. We also use a cutting plane algorithm to solve the dual subproblem since it contains a constraint for each possible path. Computational results confirm the efficiency of our algorithms.

1. Introduction

Whether it is natural or man-made, disasters such as hurricanes, earthquakes, terrorist attacks, floods, tsunamis, and nuclear accidents continue to have devastating socioeconomic impacts and affect millions of people, claiming the lives of thousands and causing massive damage in infrastructure. Various traffic management problems arise during disasters; evacuation of the disaster region being one of the most important. In a report by the U.S. Federal Emergency Management Agency (FEMA), it is stated that 45–75 disasters require an evacuation annually (Transportation Research Board 2008). Hurricane Floyd in 1999 (CNN 2001) and Hurricanes Katrina and Rita in 2005 (Transportation Research Board 2008) required millions of people to evacuate, creating the largest traffic jams in U.S. history. If the evacuation is not planned and managed effectively, the surge in evacuation traffic demand can cause congestion and may leave the evacuees in harm’s way, possibly resulting in further losses. Successful evacuation management does not just save lives, it contributes to the community’s regaining functionality in a fast and smooth way (Perry 1979).

To protect an endangered population from the harming impact of a disaster, and to provide the evacuees with food, medical care, and accommodation, shelters are generally chosen among the existing facilities with possible modifications to meet safety standards. The evacuation planning needs to decide whether to construct additional permanent shelters that meet FEMA standards, and the locations of those shelters (Li, Jin, and Zhang 2011). The Federal Emergency Management Agency (1988, 2006, 2008) and the American Red Cross (2002) provide the basis for the selection, design, and construction of shelters against different types of disasters. In the report by the American Red Cross (2002), it is stated that existing unacceptable facilities can be enhanced to be used as hurricane evacuation shelters and minor modifications of municipal, community, or school buildings are suggested, such as the addition of hurricane shutters, while buildings are being planned. The Federal Emergency Management Agency (2006) classifies shelters as standalone shelters and internal shelters, and defines a standalone shelter as a separate building that is designed and constructed.
to withstand the range of natural and man-made hazards, in neighborhoods where existing homes lack shelters.

The preparations to construct a new shelter or to retrofit an existing facility for that purpose require time and needs to be done before the disaster takes place. For that reason, the decision of where to locate the shelters is often made before a disaster occurs. Furthermore, the uncertainty about demand and the infrastructure is not resolved even after the disaster hits. Hence, one has to decide on which locations to use as shelters, thinking of different scenarios for the problem parameters. Owing to the fact that in an evacuation, the population at risk will try to reach one of those safe shelters as quickly as possible, the decision of where to locate the shelters should be considered simultaneously with evacuation traffic management decisions, as it affects the time to evacuate a disaster region drastically (Bayram, Tansel, and Yaman 2015). Addressing these two problems separately may lead to suboptimal results.

Generally, evacuation planning is done with inexact or incomplete information, as it is not easy to predict when and where disasters will occur and with how much impact. Because of the unpredictability of human behavior during disasters, specifically whether they will obey the evacuation order or not, it is not easy to estimate the evacuation demand. In 2005, the evacuation of the endangered population during Hurricane Rita caused 100-mile-long congestion on the highway, with further fatalities not caused by the hurricane itself (O’Driscoll, Wolf, and Hampson 2005). The impact of the disaster may cause road segments to be inundated in a flood or blocked by debris in a hurricane or after an earthquake, resulting in the loss of capacity or total disruption in some parts of the evacuation roadway infrastructure. Likewise, the predetermined shelter sites can be affected.

Bayram and Yaman (2015) proposed a scenario-based two-stage stochastic evacuation planning model that optimally locates shelters and assigns evacuees to the nearest shelters and the shortest paths to those shelters within a given degree of tolerance to minimize the expected total evacuation time. The nonlinear mixed-integer programming model they propose considers the uncertainty about future realizations of the evacuation demand, disruption in the road network and degraded road capacities, and disruption of the shelters. They show the significance of including uncertainty in planning for evacuations.

Our aim is to propose an exact algorithm based on Benders decomposition (BD) (Benders 1962) to solve the formulation proposed by Bayram and Yaman (2015) with a large number of scenarios. The second stage of the model is a second-order cone programming (SOCP) problem since the nonlinear objective function is represented with SOCP. We propose an algorithm that uses duality results for SOCP to derive optimality cuts in a BD setting. We solve practical-size problems with up to 1,000 scenarios in moderate CPU times. We investigate methods such as adopting a multicut strategy, deriving Pareto-optimal cuts, using a reduced primal subproblem and preemptive priority multiobjective program to enhance the proposed algorithm. The dual subproblem contains a constraint for each path, and hence its size gets larger as the tolerance level increases. To deal with this issue, we propose a cutting plane framework to solve the dual subproblem. Computational results confirm the efficiency of our algorithm as it is considerably faster and can solve instances with a larger number of scenarios compared to solving the extended formulation (EF) with an off-the-shelf solver. Furthermore, employing a cutting plane framework enables us to solve instances with larger networks and higher tolerance levels.

The rest of the paper is organized as follows. In Section 2, we present a literature review on evacuation planning focusing on stochastic location-allocation problems and their solution methodologies. In Section 3, we describe the problem and present a two-stage stochastic mixed-integer nonlinear programming formulation (MINLP). We propose a BD approach in Section 4 and explore ways to improve it in Section 5. In Section 6, we introduce a cutting plane framework for solving the subproblem. We present the computational results of our study in Section 7 and conclude in Section 8.

2. Literature Review

2.1. Traffic Assignment Models

Evacuation planning/management models are mostly based on existing traffic assignment models. The common models are the stochastic user equilibrium (SUE), user equilibrium (UE), system optimal (SO), and nearest allocation (NA) models. The SUE approach assumes that the evacuees choose the shortest travel time path depending on their perception of the travel time. The UE approach is a special case of the SUE approach when the variance of travel time perception by the evacuees is zero. In the UE approach, it is assumed that the evacuees have perfect information about the travel times on every possible route and that they can find the optimal routes (Sheffi 1985). For both SUE and UE, the goal of the evacuees is to minimize their individual travel times, and for that reason, SUE and UE do not necessarily minimize the total evacuation time in the evacuation network. Furthermore, the perfect information assumption may not be valid in case of a disaster (Galindo and Batta 2013), since the surge in traffic demand is unusual and it is difficult for evacuees to guess the traffic congestion on a route, let alone choose the minimum-latency
routes using their past experiences (Pel, Bliemer, and Hoogendoorn 2012).

The main goal of the evacuation management authority, on the other hand, is to evacuate everyone to safety as soon as possible by minimizing the total evacuation time. This traffic assignment plan is called SO. The SO solution may allocate some evacuees to shelters much further away than they would normally choose to go, and to routes much longer than they would normally take, for the benefit of the other evacuees. Bayram, Tansel, and Yaman (2015) show in their study that such a traffic assignment may assign some evacuees to routes/shelters tens of times longer and farther away than the shortest/nearest one they would take. In a disaster, it may not be reasonable to ask evacuees to accept such distant shelters and such long routes, since they will be trying to flee from the danger zone and reach safety at a shelter as quickly as possible.

In a disaster, the information on path lengths or free-flow travel times is more accessible to evacuees compared to actual travel times. This idea is the motivation for the NA model—i.e., in the NA model, each evacuee uses a shortest path based on length (geographical distance) or free-flow travel time to reach the nearest shelter. Although this approach may be a reasonable one during an evacuation, such a traffic assignment may lead to poor system performance.

The notion of constrained system optimal (CSO) traffic assignment is first defined by Jahn et al. (2005) for a route guidance system. This model takes into consideration the individual needs through additional constraints to make sure that drivers are assigned to “acceptable” paths only. Bayram, Tansel, and Yaman (2015) propose a CSO model that considers the shelter location decisions simultaneously with efficient but fair evacuation traffic assignment decisions by assigning evacuees to the nearest shelters and to the shortest paths to those shelters—shortest and nearest within a given degree of tolerance—to minimize the total evacuation time. They show that the location and the number of shelters opened drastically affect the evacuation plan and that the SO solution may be unacceptably unfair to evacuees, whereas the NA solution may result in substantial deterioration in system performance. Their results show that the CSO model is a good compromise between efficiency and fairness for a suitable tolerance factor.

2.2. Evacuation Planning Models

There is a vast amount of literature that proposes new ideas, models, or solution methodologies to support evacuation planning/management decisions. However, despite the fact that evacuation planning is typically characterized by great uncertainties, the studies in the literature mostly rely on deterministic models that adopt a single hazard scenario such as worst case or most probable scenario. Some of these models do not consider the optimal selection of shelter sites (Yamada 1996; Cova and Johnson 2003; Kalafatas and Peeta 2009; Xie, Lin, and Waller 2010; Tüydeş 2005; Chiu et al. 2007; Stepanov and Smith 2009; Ng, Park, and Waller 2010; So and Daganzo 2010; Hamacher and Tjandra 2002; Bretschneider 2013; Bish, Sherali, and Hobekia 2013). The deterministic evacuation studies that consider the optimal selection of shelter sites and that also take into account the congestion effect are either single level or bilevel models. Single-level models (Sherali, Carter, and Hobeika 1991) decide on the shelter locations and the shelter/traffic assignments in a SO manner. The bilevel models (Kongsomsaksakul, Yang, and Chen 2005) bring together the two conflicting SO and UE ideas. They specify the locations of shelter sites in a SO manner at the upper level and assign evacuees to shelters and routes in a UE manner at the lower level.

The evacuation studies in the literature that take into consideration uncertainty mostly focus on demand uncertainty (Yao, Mandal, and Chung 2009; Huibregtse, Hoogendoorn, and Bliemer 2010; Ng and Waller 2010; Yazici and Ozbay 2010) and/or capacity uncertainty (Shen et al. 2008; Ng and Waller 2010; Yazici and Ozbay 2010). Shen et al. (2008) develop two different scenario-based, stochastic, bilevel models that minimize the maximum UE travel time among all node shelter pairs by locating shelters at the upper level and assigning evacuees to shelters and routes in a UE manner at the lower level. The first model they propose decides on the locations of shelters and considers the distance between the demand nodes and the shelter sites as well as the demand as uncertain parameters. To solve this first model, they present a genetic algorithm-based approach. The second model is proposed to model real-time decision making during evacuations, and a simulation-based approach that uses a successive shortest path algorithm is developed to solve it. Yao, Mandal, and Chung (2009) consider the demand uncertainty in their study. They propose a cell transmission model (CTM)–based, robust linear programming (LP) model that considers the susceptibility of an area to a disaster at a particular time, minimizing the evacuees’ threat exposure during evacuation. They solve their model using an off-the-shelf solver. Huibregtse, Hoogendoorn, and Bliemer (2010) propose a model that considers uncertainty in demand, the behavior of people, and the hazard in a scenario-based setting. They use an off-the-shelf solver to solve their problem. Ng and Waller (2010) consider demand and capacity uncertainty together in a scenario-based evacuation planning model. They provide a framework that...
determines the amount of demand inflation and supply deflation necessary to ensure a user-specified reliability level. They solve their problem using an off-the-shelf solver for a range of nine scenarios. Yazıcı and Özbay (2010) take into consideration the uncertainty in demand and capacity simultaneously. They propose a CTM-based SO dynamic traffic assignment formulation with probabilistic constraints. They use a penalty points method by Prékopa (1995) to write the deterministic equivalent formulation of the problem and solve it with an off-the-shelf solver for three scenarios. Kulshrestha et al. (2011) develop a bilevel model that minimizes the total cost to establish and operate shelters and evacuates everyone to safety in a UE manner. They focus on demand uncertainty and confine the uncertain demand to an uncertainty set. Their model is formulated as a mathematical program with complementarity constraints and is solved by a cutting plane algorithm, for a total of three (nominal, low, and high) demand scenarios. Li et al. (2012) propose a scenario-based model that chooses optimal locations of shelters, which are robust for a range of hurricane events, by considering disruption in shelter sites. At the upper level of their model, the central authority selects the shelter sites for a particular scenario. The objective of the upper-level problem is to minimize the weighted sum of the expected unmet shelter demand and the expected total network travel time. In the lower level, evacuees choose their routes in a dynamic UE manner. They develop heuristic algorithms based on Lagrangian relaxation and present a case study for the state of North Carolina for 33 hurricane scenarios.

To our knowledge, there is one study that simultaneously considers the uncertainty in disruption/degradation in road network structure, evacuation demand, and disruption in shelters. Bayram and Yaman (2015) introduce a novel scenario-based model that decides simultaneously on the locations of shelters and the allocations of evacuees to shelters and routes under uncertainty. Their model incorporates evacuees’ preferences and fairness considerations by routing the evacuees on paths that are not much longer than the shortest paths to the nearest shelters. They solve practical-size problems exactly by using a SOCP approach for a range of up to 50 scenarios by using an off-the-shelf solver.

2.3. Our Contribution
We base our work on the stochastic CSO (SCSO) problem by Bayram and Yaman (2015). To be able to model a stochastic evacuation planning problem more realistically, one needs to consider a large number of scenarios. As the number of scenarios grows, the EF developed by Bayram and Yaman (2015) may not be solved within reasonable CPU times or may not be solved at all. The evacuation models in the literature that take into account congestion by employing a nonlinear objective function are generally solved by heuristic methodologies, especially for large evacuation road networks. Against this backdrop, we propose an exact algorithm based on Benders decomposition. We test various ways of enhancing the algorithm and report the results of our computational experiments. Furthermore, we propose a cutting plane algorithm to solve the dual subproblem for instances with larger networks and tolerance levels. The results show that the algorithm can solve real problems with up to 1,000 scenarios and is faster compared to solving the EF with an off-the-shelf solver.

3. Problem Description and Model Formulation
The SCSO problem proposed by Bayram and Yaman (2015) decides on where to locate $p$ shelters and how to assign evacuees at origins in risk zones to their destinations (shelters), including the decision of how to assign evacuees to routes so as to minimize total evacuation time. The problem is defined on a directed road network $G = (N, A)$, where $N$ is the set of nodes and $A$ is the set of arcs (road segments) in the network. We define $O \subset N$ as the set of origin (demand) nodes from where the population at risk are to be evacuated and $F \subset N$ as the set of destination nodes (potential shelter sites) where evacuees reach to safety, $O$ and $F$ being disjoint.

Since we are proposing a strategic evacuation planning model, and since no-notice evacuations are adequately represented by static models as evacuees are loaded into an evacuation network at once (Özbay et al. 2012), we use a static modeling approach. The travel time spent on a given road segment is a function of traffic flow and increases monotonically, since an increase in traffic volume decreases the travel speed because of congestion and hence increases the travel time along the road segment. To express the relationship between travel time and the volume of traffic on a road segment, we employ the Bureau of Public Roads (BPR) function, which is convex, nondecreasing, and differentiable, as in Sherali, Carter, and Hobetika (1991); Kongsomaksakul, Yang, and Chen (2005); Ng and Waller (2010); and Li et al. (2012). The BPR function is formulated as

$$t(x) = t^0 \left(1 + \alpha \left(\frac{x}{c}\right)^\beta\right),$$

where $t(x)$ is the travel time at which assigned volume $x$ can travel on the road segment, $c$ is the practical capacity (maximum flow rate), and $t^0$ is the free-flow travel time at zero volume. The parameters $\alpha \geq 0$ and $\beta \geq 0$ are the parameters that reflect the road characteristics and they are taken as $0.15$ and $4$, respectively.
by the U.S. Department of Commerce BPR (Bureau of Public Roads 1964).

In a two-stage stochastic setting, the first stage of our evacuation problem is about where to locate the shelters, before a disaster takes place. The binary variable \( y_s \) is 1 if a shelter is located/opened at node \( s \in F \), and 0 otherwise. Given the shelter location decisions from the first stage and the realization of the evacuation demand and the impact of the disaster on the road network and the shelters, the second stage assigns evacuees to shelters and to routes. We define \( \Omega \) as the set of scenarios in a disaster and associate a probability \( p(\omega) \) to each scenario \( \omega \in \Omega \). We define \( F(\omega) \subseteq F \) as the set of potential shelters that are usable (not disrupted) in scenario \( \omega \in \Omega \). \( c_s(\omega) \) as the (possibly degraded) capacity of road segment \( a \) in scenario \( \omega \in \Omega \) with \( 0 \leq c_s(\omega) \leq c_a \), for all \( a \in A \), and \( A(\omega) \subseteq A \) as the set of usable arcs—i.e., \( c_s(\omega) > 0 \), \( a \in A(\omega) \). A shelter \( s \) is reachable from demand point \( r \) in scenario \( \omega \) if this shelter is usable and if there exists a route with usable arcs from \( r \) to \( s \) in this scenario. We define \( F(\omega) \) as the set of reachable shelters for demand point \( r \in O \) in scenario \( \omega \in \Omega \). For feasibility, we assume that there exists at least one reachable shelter for every origin \( r \in O \) in every scenario \( \omega \). The amount of demand at origin \( r \in O \), \( w_r(\omega) \), is the number of passenger vehicles that will be evacuated in scenario \( \omega \in \Omega \).

Let \( P_{rs}^\lambda(\omega) \) be the set of alternative paths from demand point \( r \) to shelter \( s \) in scenario \( \omega \in \Omega \). The tolerance of an evacuee to the length of a path he or she is offered to take is denoted by \( \lambda \). Based on this tolerance level, the evacuation planning process does not assign an evacuee to a path whose length is more than \((1 + \lambda)\) times the length of a shortest path to the closest open and usable shelter in a given scenario. This implies that some evacuees may be assigned to an open shelter within this degree of tolerance, although it might not be the nearest one. We define \( P_{rs}^\lambda(\omega) = \{ \pi \in P_{rs}(\omega) : d_\pi \leq (1 + \lambda)d_{rs}(\omega) \} \) as the set of acceptable and usable paths from origin \( r \) to destination \( s \) of tolerance level \( \lambda \) in a given scenario \( \omega \in \Omega \). In this definition, \( d_\pi \) is the length of path \( \pi \), and \( d_{rs}(\omega) \) is the length of a shortest path from \( r \) to \( s \) in scenario \( \omega \in \Omega \). This set is computed using an algorithm developed by Byers and Waterman (1984). We use geographical distances to compute the set \( P_{rs}^\lambda(\omega) \). Free-flow travel times \( t_{rs}(\omega) \) could also be used.

For a given origin \( r \) and a scenario \( \omega \in \Omega \), the set of acceptable paths is defined based on the length of a shortest path from node \( r \) to the closest open and usable shelter. As we do not know a priori which shelters will be opened, we do not know the actual set of acceptable paths; yet we know that this set is a subset of the union of \( P_{rs}^\lambda(\omega) \) over all potential shelters \( s \).

In our model, we also use the following decision variables: \( \pi_x(\omega) \) is the fraction of origin \( r \)'s demand that uses path \( \pi \in P_{rs}^\lambda(\omega) \) from origin \( r \in O \) to shelter \( s \in F(\omega) \) in scenario \( \omega \in \Omega \) and \( x_s(\omega) \) is the amount of traffic on arc \( a \in A(\omega) \) in scenario \( \omega \in \Omega \).

The EF developed by Bayram and Yaman (2015) for the SCSO problem is as follows:

\[
\min \sum_{\omega \in \Omega} p(\omega) \sum_{a \in A(\omega)} t_{rs}^\lambda \left( 1 + \alpha \left( \frac{x_s(\omega)}{c_a(\omega)} \right)^{\beta} \right) x_s(\omega) \tag{1}
\]
\[
\text{s.t.} \sum_{x \in F(\omega)} y_x = p(\omega) \tag{2}
\]
\[
\sum_{a \in F(\omega) \pi \in P_{r}^\lambda(\omega)} v_{n}(\omega) = 1, \quad \forall r \in O, \omega \in \Omega, \tag{3}
\]
\[
\sum_{a \in F(\omega)} y_x \geq 1, \quad \forall r \in O, \omega \in \Omega, \tag{4}
\]
\[
\sum_{a \in F(\omega) \pi \in P_{r}^\lambda(\omega)} v_{n}(\omega) \leq y_x, \quad \forall r \in O, \omega \in \Omega, s \in F(\omega), \tag{5}
\]
\[
\sum_{a \in F(\omega) \pi \in P_{r}^\lambda(\omega)} t_{rs}(\omega) v_{n}(\omega) + y_x \leq 1, \quad \forall r \in O, \omega \in \Omega, i \in F(\omega), \tag{6}
\]
\[
x_s(\omega) = \sum_{r \in O} \sum_{a \in F(\omega) \pi \in P_{r}^\lambda(\omega): \pi \ni r} w_r(\omega) v_{n}(\omega), \quad \forall \omega \in \Omega, a \in A(\omega), \tag{7}
\]
\[
v_x(\omega) \geq 0, \quad \forall \omega \in \Omega, \pi \in \bigcup_{r \in O, \omega \in \Omega} P_{r}^\lambda(\omega), \tag{8}
\]
\[
y_x \in \{0, 1\}, \quad \forall x \in F. \tag{9}
\]

Objective function (1) minimizes the expected total evacuation time spent by the evacuees in the network. There may be reasons such as the available budget and personnel that restrict the number of shelters that can be opened. Constraint (2) limits the number of shelters open to this prespecified number \( p \). This constraint can be replaced with a budget constraint if the data on the associated costs are available. Constraints (3) ensure that evacuation demand from every origin \( r \) is assigned to a shelter and a route that leads to that shelter, for every scenario. We added constraints (4) to ensure that there is at least one open and reachable shelter for each demand point and each scenario. Constraints (5) prevent assigning demand to a non-open shelter. Constraints (6) ensure that if the shelter at site \( i \) is open and usable in scenario \( \omega \), then the demand at origin node \( r \) cannot be routed on any path whose length is longer than \((1 + \lambda)\) times the length of a closest path to the shelter. Finally, the set of constraints (7) computes the traffic on every arc in each scenario, and constraints (8) and (9) are variable restrictions.

Bayram and Yaman (2015) point out that the SCSO problem is NP-hard even when \( \alpha = 0 \) and \( G \) is bipartite. They also note that SCSO generalizes the classical \( p \)-median facility location problem. SCSO also generalizes the SO and NA traffic assignment approaches. When \( \lambda = 0 \), we have the NA model, and when \( \lambda = \infty \), we obtain a model for the SO traffic assignment.
Finally, SCSO generalizes the congested facility location (Desrochers, Marcotte, and Stan 1995; Fischetti, Ljubić, and Sinnl 2016) where the congestion costs at facilities can be modeled by splitting facility nodes into arcs with convex congestion costs.

4. Benders Decomposition Approach

To be able to model a stochastic evacuation problem more realistically, one needs to consider a large number of scenarios. As the number of scenarios increases, the EF developed by Bayram and Yaman (2015) may not be solved within reasonable CPU times or may not be solved at all. For that reason, we develop BD (Benders 1962) and generalized BD (Geoffrion 1972) based approaches to solve our problem considering a large number of scenarios and explore methodologies to accelerate our BD algorithm. This algorithm can also be considered as an “L”-shaped algorithm (LSA) developed by Van Slyke and Wets (1969) to solve stochastic programs with recourse. Our problem is well fit for the BD approach and LSA as these algorithms have been successfully implemented for location (Laporte, Louveaux, and van Hamme 1994; Contreras, Cordeau, and Laporte 2011; Noyan, Balci, and Atakan 2016; Martins de Sá et al. 2015; Álvarez-Miranda, Fernández, and Ljubić 2015), routing (Cordeau, Soumis, and Desrosiers 2000; Cordeau et al. 2001; Laporte, Louveaux, and van Hamme 2002; Sherali and Zhu 2008; Úster and Kewcharoenwong 2011; Sherali, Bae, and Haouari 2013), location-routing (Franca and Luna 1982), large-scale (Noonan and Giglio 1977; Cordeau et al. 2001; Sherali and Zhu 2008; Contreras, Cordeau, and Laporte 2011; Sherali, Bae, and Haouari 2013; Martins de Sá et al. 2015), nonlinear (Noonan and Giglio 1977), and stochastic optimization (Franca and Luna 1982; Laporte, Louveaux, and van Hamme 1994; 2002; Sherali and Zhu 2008; Noyan, Balci, and Atakan 2016) problems.

In the BD approach, the main idea is to project out the second-stage variables. The resulting problem is called the master problem (MP), and it contains fewer variables but a large number of constraints. These constraints are known as Benders cuts (BC), and most of them are not active at an optimal solution. Because of this fact, the most natural strategy to solve the MP is through relaxation (Geoffrion 1972). An iterative solution methodology is pursued, by solving the relaxed MP at every iteration and passing the optimal solution to subproblems that are basically duals of the original problem with shelter location decisions temporarily fixed to the values obtained from the MP and adding the violated BCs to MP until all of them are satisfied at a relaxed MP solution. Since the MP is a relaxation, its optimal value provides a lower bound on the optimal value, and an upper bound is given by the expectation of the optimal values of the subproblems.

In our model, the shelter location decisions $y$ are taken in the presence of uncertainty. For that reason, they are called first-stage (design) variables. In the second stage, the uncertainty is revealed and recourse decisions—i.e., shelter and route assignment decisions $v$—are taken. However, while making decisions about where to locate the shelters in the first stage, their effect on the second-stage assignment decisions and total evacuation time is taken into account. The future effects of shelter location decisions are measured by taking the expectation of the recourse function on possible scenarios. Our model has a 0–1 first-stage problem and a nonlinear second-stage problem.

The first-stage variables in our problem—i.e., shelter location decisions—are the complicating variables, and their number is less than that of the second-stage variables. We project out the second-stage variables $v$ and $x$ by fixing the first-stage variables $y$ to a value given by the MP. This results in the following primal subproblems, one for each scenario $\omega \in \Omega$.

**Primal Subproblem (PSP($\bar{y}, \omega$))**

\[
\begin{align*}
\min & \quad \sum_{a \in A(\omega)} x_{a}(\omega)^{\frac{1}{\gamma}} \left(1 + a \left(\frac{x_{a}(\omega)}{c_{a}(\omega)}\right)\right) \chi_{a}(\omega) \\
\text{s.t.} & \quad \sum_{\pi \in \Pi(\omega)^{1}(\omega)} \sum_{s \in S(\omega)^{1}(\omega)} \sum_{r \in R(\omega)} v_{\pi}(\omega) = 1, \quad \forall r \in R, \\
& \quad \sum_{\pi \in \Pi(\omega)^{1}(\omega)} \sum_{s \in S(\omega)^{1}(\omega)} \sum_{r \in R(\omega)} v_{\pi}(\omega) \leq \bar{y}_{r}, \quad \forall r \in R, s \in F(\omega), \\
& \quad \sum_{\pi \in \Pi(\omega)^{1}(\omega), \pi \in \Pi(\omega)^{2}(\omega)} \sum_{s \in S(\omega)^{1}(\omega)} \sum_{r \in R(\omega)} v_{\pi}(\omega) \leq 1 - \bar{y}_{r}, \forall r \in R, i \in F(\omega), \\
& \quad x_{a}(\omega) = \sum_{r \in R(\omega)} \sum_{\pi \in \Pi(\omega)^{1}(\omega), \pi \in \Pi(\omega)^{2}(\omega)} w_{\pi}(\omega) v_{\pi}(\omega), \\
& \quad \forall a \in A(\omega), \\
& \quad v_{\pi}(\omega) \geq 0, \quad \forall \pi \in \bigcup_{r \in R(\omega), s \in S(\omega)} \Pi(\omega)^{1}(\omega), \\Pi(\omega)^{2}(\omega), \\
\end{align*}
\]

where $\bar{y} \in V \subseteq Y$ is a fixed vector for the complicating variables, $V \subseteq Y$ is the set of vectors $y$ that renders $PSP(\bar{y}, \omega)$ feasible, and $Y$ is the set of all possible shelter location decisions $y$. Given the locations of the shelters, each subproblem is a nonlinear CSO shelter and traffic assignment problem.

With the theoretical findings in the last two decades and the applicability of efficient interior point algorithms (Nesterov, Nemirovskii, and Ye 1994), SOCP has become a state-of-the-art technique in mathematical programming. We refer the reader to Ben-Tal and Nemirovski (2001) and Alizadeh and Goldfarb (2003) for an introduction to SOCP. Ben-Tal and Nemirovski (2001) state that as a well-structured convex optimization problem, solving an SOCP is no more difficult than solving an LP of a similar size. Many convex optimization problems can be modeled as SOCPs (see, e.g., Lobo et al. 1998; Ben-Tal and Nemirovski 2001).

Motivated by these studies, we reformulate the \( \text{PSP} (\bar{y}, \omega) \) as a SOCP model as in Bayram and Yaman (2015). By employing SOCP, the nonlinearity is transferred to the constraint set in the form of second-order quadratic constraints. This is done as follows. We first define auxiliary variables \( \mu_x (\omega) \) for each \( a \in A(\omega), \omega \in \Omega \) and move the nonlinearity from the objective function to the constraints—i.e., the objective function of the \( \text{PSP} (\bar{y}, \omega) \) becomes

\[
\sum_{a \in A(\omega)} \left( t^0_x \theta_x (\omega) + \frac{t^0_\omega}{c_x (\omega)} \mu_x (\omega) \right),
\]

and we add the constraints \( x_\omega (\omega) \beta + 1 \leq \mu_x (\omega) \) for all \( a \in A, \omega \in \Omega \).

Consider an inequality of the form

\[
r^2 \leq s_1 s_2 \ldots, s_{2^r} \quad \text{for} \quad r, s_1, s_2, \ldots, s_{2^r} \geq 0.
\]

An equivalent representation of the inequality (16) can be achieved by using \( O(2^r) \) variables and \( O(2^r) \) inequalities of the form

\[
u^2 \leq v_1 v_2, u, v_3, v_2 \geq 0,
\]

which are referred to as hyperbolic inequalities since they describe half a hyperboloid (Lobo et al. 1998; Ben-Tal and Nemirovski 2001; Alizadeh and Goldfarb 2003; Günlük and Linderth 2008). Then, each hyperbolic inequality can easily be transformed into a second-order cone inequality

\[
\|2u, v_1 - v_2\| \leq v_1 + v_2.
\]

We take \( \beta = 4 \) and represent \( x_\omega (\omega) \beta \leq \mu_x (\omega) \) with hyperbolic inequalities of the form

\[
\begin{align*}
x_\omega (\omega) & \leq \theta_x (\omega) h_x (\omega), \\
\theta_x (\omega) & \leq u_x (\omega) x_\omega (\omega), \\
u_x (\omega) & \leq \mu_x (\omega) x_\omega (\omega), \\
h_x (\omega) & = 1, \theta_x (\omega), u_x (\omega), x_\omega (\omega), \mu_x (\omega) \geq 0,
\end{align*}
\]

where \( h_x (\omega), \theta_x (\omega), \) and \( u_x (\omega) \) are auxiliary variables that are used to define hyperbolic inequalities.

These hyperbolic inequalities are represented by their respective quadratic cone constraints

\[
\begin{align*}
\|2x_\omega (\omega), \theta_x (\omega) - h_x (\omega)\| & \leq \theta_x (\omega) + h_x (\omega), \\
\|2\theta_x (\omega), u_x (\omega) - x_\omega (\omega)\| & \leq u_x (\omega) + x_\omega (\omega), \\
\|2u_x (\omega), \mu_x (\omega) - x_\omega (\omega)\| & \leq \mu_x (\omega) + x_\omega (\omega), \\
h_x (\omega) & = 1, \theta_x (\omega), u_x (\omega), x_\omega (\omega), \mu_x (\omega) \geq 0.
\end{align*}
\]

Hence, the subproblem is still a nonlinear programming problem, but it can be solved efficiently once represented as SOCP. Note that SOCP problems are solved with the barrier algorithm in CPLEX (IBM ILOG CPLEX 2011).

The resulting conic primal subproblem with SOCP constraints (CPSP) is given below.

**Conic Primal Subproblem (CPSP) \((\bar{y}, \omega)\)**

\[
\begin{align*}
\min \quad & \sum_{a \in A(\omega)} \left( t^0_x x_\omega (\omega) + \frac{t^0_\omega}{c_x (\omega)} \mu_x (\omega) \right) \\
\text{s.t.} \quad & (11)\text{--}(14), \\
& x_\omega (\omega) + \rho_x (\omega) \leq \delta_x (\omega), \quad \forall a \in A(\omega), \\
& \theta_x (\omega) + \sigma_x (\omega) \leq \phi_x (\omega), \quad \forall a \in A(\omega), \\
& u_\omega (\omega) + \gamma_x (\omega) \leq \eta_x (\omega), \quad \forall a \in A(\omega), \\
& -x_\omega (\omega) + 2x_\omega (\omega) = 0, \quad \forall a \in A(\omega), \\
& -\rho_x (\omega) + \theta_x (\omega) = 1, \quad \forall a \in A(\omega), \\
& -\delta_x (\omega) + \theta_x (\omega) = -1, \quad \forall a \in A(\omega), \\
& -\phi_x (\omega) + 2\theta_x (\omega) = 0, \quad \forall a \in A(\omega), \\
& -\sigma_x (\omega) + u_x (\omega) - x_\omega (\omega) = 0, \quad \forall a \in A(\omega), \\
& -\psi_x (\omega) + \mu_x (\omega) + x_\omega (\omega) = 0, \quad \forall a \in A(\omega), \\
& -\delta_x (\omega) + x_\omega (\omega) = 0, \quad \forall a \in A(\omega), \\
& \eta_x (\omega) + \phi_x (\omega), \mu_x (\omega) \geq 0, \quad \forall a \in A(\omega).
\end{align*}
\]

Objective function (23) is modified from the original objective function as defined above, and constraints (11)–(14) are the original constraints from \( \text{PSP} (\bar{y}, \omega) \). Constraints (24)–(26) define the three second-order quadratic cones, and constraints (27)–(35) are generated by replacing each term (the two terms on the left-hand side inside the norms and the term on the right-hand side) of SOCP constraints (19)–(21) by a single auxiliary variable to help derive the dual of the CPSP(\( \bar{y}, \omega \)). Constraints (36) and (37) are variable restrictions.

We associate the dual variables \( z_\omega (\omega), \Gamma_, (\omega), \Lambda_\omega (\omega), \psi_x (\omega), c_{1\omega} (\omega), c_{2\omega} (\omega), c_{3\omega} (\omega), c_{4\omega} (\omega), c_{5\omega} (\omega), c_{6\omega} (\omega), \). \)
c_{g_2}(\omega), c_{g_3}(\omega), and c_{g_4}(\omega) for constraints (11)–(14) and (27)–(35), respectively, and the resulting dual subproblem (DSP) is formulated as follows:

**Dual Subproblem (DSP(\bar{y}, \omega))**

\[
\max \left\{ \sum_{r \in O} z_r(\omega) + \sum_{r \in F(\omega)} \Gamma_r(\omega) \bar{y}_r \right. \\
+ \sum_{r \in F(\omega)} \sum_{s \in F(\omega)} \Lambda_r(\omega)(1 - \bar{y}_r) + \sum_{a \in A(\omega)} c_{g_2}(\omega) \\
- \sum_{a \in A(\omega)} c_{g_3}(\omega) \right\}
\]

\[
\text{s.t. } z_r(\omega) + \Gamma_r(\omega) + \sum_{a \in A(\omega)} \Lambda_r(\omega)
- \sum_{a \in A(\omega)} w_r(\omega) \psi_a(\omega) \leq 0,
\quad \forall r \in O, s \in F(\omega), \pi \in P^l_{\pi r}(\omega),
\psi_a(\omega) + 2c_{g_2}(\omega) - c_{g_5}(\omega) + c_{g_7}(\omega) - c_{g_8}(\omega) \\
+ c_{g_9}(\omega) \leq t^0_a, \quad \forall a \in A(\omega),
\]

\[
c_{g_8}(\omega) + c_{g_9}(\omega) \leq \frac{t^0_a}{c_{g_4}(\omega)} \left(1 + \lambda\right) \psi_a(\omega) + 2c_{g_2}(\omega) - c_{g_5}(\omega) + c_{g_7}(\omega) - c_{g_8}(\omega) \\
+ c_{g_9}(\omega) \leq t^0_a, \quad \forall a \in A(\omega),
\]

\[
c_{g_2}(\omega) + c_{g_3}(\omega) + 2c_{g_4}(\omega) \leq 0, \quad \forall a \in A(\omega),
\]

\[
c_{g_2}(\omega) + c_{g_3}(\omega) + 2c_{g_4}(\omega) \leq 0, \quad \forall a \in A(\omega),
\]

\[
c_{g_2}(\omega) + c_{g_3}(\omega) + 2c_{g_4}(\omega) \leq 0, \quad \forall a \in A(\omega),
\]

\[
c_{g_2}(\omega)^2 + c_{g_3}(\omega)^2 \leq c_{g_4}(\omega)^2, \quad \forall a \in A(\omega),
\]

\[
c_{g_2}(\omega)^2 + c_{g_3}(\omega)^2 \leq c_{g_4}(\omega)^2, \quad \forall a \in A(\omega),
\]

\[
c_{g_2}(\omega)^2 + c_{g_3}(\omega)^2 \leq c_{g_4}(\omega)^2, \quad \forall a \in A(\omega),
\]

\[
\Gamma_r(\omega) \leq 0, \quad \forall r \in O, s \in F(\omega),
\]

\[
\Lambda_r(\omega) \leq 0, \quad \forall r \in O, i \in F(\omega),
\]

\[
c_{g_2}(\omega), c_{g_3}(\omega), c_{g_4}(\omega) \geq 0, \quad \forall a \in A(\omega).
\]

The dual subproblem is also a SOCP problem. Note that when the CPSP(\bar{y}, \omega) is feasible, we can also find a point for which it is strictly feasible. Since it is also bounded, by the strong duality theorem for SOCP problems (Ben-Tal and Nemirovski 2001), the DSP(\bar{y}, \omega) is feasible and bounded, and strong duality holds—i.e., CPSP(\bar{y}, \omega) and DSP(\bar{y}, \omega) attain the same optimal values. Since \( Y \) is a finite discrete set and we need one cut for each element of \( Y \), the generalized BD procedure generates finitely many cuts and terminates in a finite number of steps (Geoffrion 1972).

We ensure that the MP generates shelter location decisions that render every subproblem feasible. Hence, we only add optimality cuts as deemed necessary. The MP is as follows:

**Master Problem (MP)**

\[
\min \theta \quad \text{(50)}
\]

\[
\text{s.t. } \sum_{a \in A(\omega)} y_r = p, \quad \text{(51)}
\]

\[
\sum_{a \in A(\omega)} y_r \geq 1, \quad \forall r \in O, \omega \in \Omega, \quad \text{(52)}
\]

\[
\theta \geq \sum_{\omega \in \Omega} p(\omega) \left( \sum_{a \in A(\omega)} y_r + \sum_{r \in F(\omega)} \sum_{i \in F(\omega)} \Lambda_r(\omega)(1 - y_r) + \sum_{a \in A(\omega)} c_{g_2}(\omega) - \sum_{a \in A(\omega)} c_{g_3}(\omega) \right), \quad \forall \omega \in G, \quad \text{(53)}
\]

\[
\theta \geq l, \quad \text{(54)}
\]

\[
y_r \in \{0, 1\}, \quad \forall s \in F, \quad \text{(55)}
\]

where \( G = \bigcup_{\omega \in \Omega} G(\bar{y}) \) is the set of optimal multiplier vectors and \( \theta \) is the surrogate variable that represents the subproblems in the MP objective function and is a lower bound on the expected total evacuation time (Van Slyke and Wets 1969; Birge and Louveaux 1997).

The objective function (50) of MP minimizes the value of the surrogate variable. Constraint (51) limits the number of shelter sites open to a prespecified number \( p \). By adding induced constraints (52) in the MP, we ensure that SPs are always feasible—i.e., there exists at least an open and reachable shelter for each \( r \) in \( O \), in every scenario \( \omega \in \Omega \). Constraint set (53) is the optimality cuts. Constraints (54) set a lower bound \( l \) on the auxiliary variable \( \theta \). We compute such a lower bound with a very simple heuristic method. In a given scenario, we find the shortest path to the closest shelter for each origin \( r \) and compute the total travel time of the vehicles on this path using the free-flow travel time. The sum of the total travel times on these paths gives us a lower bound for that specific scenario. We take their expected value to compute the lower bound \( l \). Imposing such a lower bound on \( \theta \) in the formulation improves the lower bound during initial iterations and the overall solution time to some extent. Constraints (55) define the types of variables.

### 5. Improving the Performance of the BD Algorithm

Since the BD has been introduced (Benders 1962), many researchers have investigated methods to improve its performance. Geoffrion and Graves (1974) propose a branch-and-bound framework in which they solve the MP in an \( \epsilon \)-optimal fashion instead of solving it to optimality at every iteration. McDaniel and Devine (1977) present a methodology that solves the LP relaxation of the integer subproblem for some initial number of iterations to generate Benders cuts to reduce the computational burden. Magnanti and Wong (1981) propose to accelerate the BD algorithm by generating strong (Pareto-optimal) optimality cuts from the alternate optimal of the Benders subproblem. Papadakos (2008), Fischetti, Salvagnin, and Zanette (2010), Saharidis and Ierapetritou (2010), and Sherali and Lunday (2013) propose alternative methodologies on deriving strong...
nondominated Benders cuts. Van Roy (1986) proposes a cross-decomposition method that unifies Benders decomposition and Lagrangian relaxation into a single framework. Saharidis, Minoux, and Ierapetritou (2010) demonstrate the effectiveness of a new strategy, which they refer to as covering cut bundle generation. The method they propose is based on the idea of generating multiple cuts that involve as many complicating variables as possible or directly generating a high-density Pareto cut by lifting the Pareto-optimal cuts (Tang, Jiang, and Saharidis 2013). Saharidis, Boile, and Theofanis (2011) and Tang, Jiang, and Saharidis (2013) work on deriving valid inequalities to improve the lower bounds obtained by the MP. Naoum-Sawaya and Elhedhli (2013) present an interior-point branch-and-cut algorithm based on BD and the analytic center cutting plane method (ACCPM), and show that the ACCPM-based Benders cuts are both Pareto-optimal and valid for any node of the branch-and-bound tree. Fischetti and Lodi (2003) and Rei et al. (2009) show how local branching can be used to accelerate the classical BD algorithm.

5.1. Multicut Strategy
In our initial experiments, we observed that generating a single cut aggregated from the optimal multiplier vectors of each subproblem results in slow convergence of the BD algorithm. By adding disaggregate cuts, more detailed information is given to the first iteration vectors, i.e., we add an optimality cut for every subproblem related to a scenario in case a violation is identified. Therefore, for any optimal solution of \( DSP(\bar{y}, \omega) \), the Benders optimality cuts are redefined as follows:

\[
\theta(\omega) \geq p(\omega) \left( \sum_{r \in O} z^s_0(r) + \sum_{r \in \Omega} \sum_{a \in A(r)} \Gamma^s_{rL}(a)y_s \right.
\]

\[
+ \sum_{r \in O} \sum_{i \in F(r)} \Lambda^s_{rI}(a)(1 - y_i)
\]

\[
+ \sum_{a \in A} c^s_{2a}(a) - \sum_{a \in A} c^s_{3a}(a) \right),
\]

\[\forall \omega \in \Omega, g \in G.\]

The objective function of the MP is modified as \( \sum_{w \in W} \theta(\omega) \), where \( \theta(\omega) \) is a surrogate variable that represents a subproblem related to a specific scenario \( \omega \in \Omega \), and we set a lower bound for each subproblem—i.e., we modify constraint (54) as \( \theta(\omega) \geq l(\omega) \) for all \( \omega \in \Omega \).

5.2. Implementing a Callback Routine
In the classical implementation of BD, the current relaxed MP is solved to optimality at every iteration of the algorithm, and for that reason a search tree is generated from scratch every time the relaxed MP is solved.

Alternatively, the reformulation can be solved with a branch-and-cut algorithm where the Benders cuts are generated as cutting planes. In this approach, a single integer problem is solved using a single search tree. Each time an integer solution is found, it is either certified as feasible or a Benders cut violated by this candidate solution is added. This can be done using the lazy constraint callback routines available in the commercial solvers.

We refer to the version of the BD algorithm in which we employ the multicut strategy and the callback routine as BD.

5.3. Defining Strong (Pareto-Optimal) Cuts
When the primal subproblem is a network optimization problem such as the facility location on networks, shortest route, and transshipment, it is common to get degenerate solutions, which leads to multiple optimal solutions for the dual subproblem (Magnanti and Wong 1981). Because of this fact, cuts of different strength can be generated. Although any of these are valid optimality cuts, defining strong (Pareto-optimal) ones at every iteration of the algorithm may significantly decrease the number of iterations and hence improve the overall solution time. A cut is said to be Pareto-optimal if it is not dominated by any other cut. Let \( y^0 \) be a point in the relative interior of the convex hull of feasible location vectors, i.e., \( y^0 \in ri(CH(V)) \) and \( v(DSP(\bar{y}, \omega)) \) be the optimal value of \( DSP(\bar{y}, \omega) \). To generate a Pareto-optimal cut, we solve the following auxiliary problem:

\[
\text{Magnanti–Wong (MW) Problem}
\]

\[
\max \left\{ \sum_{r \in O} z^s_r(a) + \sum_{r \in \Omega} \sum_{a \in A(r)} \Gamma^s_{rL}(a)y^0_s + \sum_{r \in O} \sum_{i \in F(r)} \Lambda^s_{rI}(a)(1 - y_i) + \sum_{a \in A} c^s_{2a}(a) - \sum_{a \in A} c^s_{3a}(a) \right\}
\]

\[\text{s.t. (39)–(49),}
\]

\[
\sum_{r \in O} z^s_r(a) + \sum_{r \in \Omega} \sum_{a \in A(r)} \Gamma^s_{rL}(a)y^0_s + \sum_{r \in O} \sum_{i \in F(r)} \Lambda^s_{rI}(a)(1 - y_i) + \sum_{a \in A} c^s_{2a}(a) - \sum_{a \in A} c^s_{3a}(a) = v(DSP(\bar{y}, \omega)).
\]

Constraint (57) in the dual auxiliary problem MW ensures that one chooses an optimal multiplier vector from among alternative ones; and the objective function of the MW problem chooses from among these multiplier vectors the one that generates the strongest cut to be added to the MP.

Papadakos (2008) points out that the dependency of the MW method on the Benders subproblem and on an MP core point may sometimes decrease the performance of the algorithm, and that it may not always be easy to find a readily available core point. Another issue
with the MW problem they discuss is the numerical unboundedness caused by constraint (57), which they show can be eliminated from the MW problem to generate a Pareto-optimal cut. That way, an enhanced MW (EMW) method independent of the subproblem, which enables adding a useful cut before solving the MP, is proposed.

Papadakos (2008) prove that $y^0$ does not have to be a core point or even a point of $V$. Furthermore, since $y^0$ only modifies the objective function and does not alter the feasible region of the MW problem, choosing a $y^0$ which is not in $r(CH(V))$ still generates a valid optimality cut, although it may not be Pareto-optimal. In our implementation, we start this algorithm with $y^0 = 1$ and update this point at every iteration $k$ using the equation

$$y^0_{k} = \frac{1}{2}y^0_{k-1} + \frac{1}{2}y^*_{k}.$$  

Both algorithms with MW and EMW methodologies solve an auxiliary dual subproblem to generate the Pareto-optimal cuts. The main drawback of these two algorithms is that one has to solve the dual subproblem and the MW auxiliary problem at every iteration for every scenario, which may result in long CPU times. Our preliminary experiments show that for both of these algorithms, the number of iterations and the total number of optimality cuts generated generally decrease compared to the BD. However, the CPU times worsen as a result of solving the dual subproblem and the MW auxiliary problem, at every iteration and for every subproblem when there is a violation. We also experimented on using a reduced primal subproblem (RPSP) in combination with the MW or EMW procedure that resulted in long CPU times as well. However, since the EMW problem is independent of the subproblem, we can take advantage of adding initial cuts to the MP before we begin solving it, and continue as in BD. We denote this algorithm as $BD_{JC}$.

Sherali and Lunday (2013) propose a procedure that generates maximal nondominated Benders cuts. Instead of solving an auxiliary MW problem at each iteration, which brings a computational burden and increases the CPU times, the authors solve a preemptive priority multiple objective program. The aim is to solve the original dual subproblem optimally with the first priority, and among alternative optimal solutions to this problem, maximize (56). They point out that there exists a $\zeta > 0$ small enough such that this preemptive priority multiple objective program can be equivalently represented as the following weighted-sum problem:

**Modified Sherali and Lunday Dual Subproblem**

$$\max \left(38\right) + \zeta \sum_{r \in \Omega} z_r(\omega) + \sum_{r \in \Omega} \sum_{s \in F(\omega)} \Gamma_r(\omega)y^0_{s}$$

$$+ \sum_{r \in \Omega} \sum_{s \in F(\omega)} \Lambda_r(\omega)(1 - y^0_{s})$$

$$+ \sum_{s \in A(\omega)} c_2(\omega) - \sum_{s \in A(\omega)} c_3(\omega)$$  

s.t. (39)–(49),

where $y^0$ is a positive-weight vector—i.e., a positive core point solution as we defined previously. We begin with a core point and update it as we described in the EMW method. We take $\zeta = 10^{-11}$. We denote this algorithm as $BD_{SL}$.

Fischetti, Salvagnin, and Zanette (2010) propose a new selection criterion for Benders cuts—in particular, when both violations of feasibility and optimality cuts exist. They represent the primal subproblem as a pure feasibility problem. Preliminary computational studies showed that this methodology could not find a solution in good CPU times for our problem.

We also investigated two alternative algorithms based on the generalized BD (GBD) method introduced by Geoffrion (1972) to solve the MINLP. While the first one is the original GBD algorithm proposed by Geoffrion (1972), the second algorithm uses an acceleration strategy called “variable fixing” proposed by Fischetti, Ljubić, and Sinnl (2016). We observed that BD algorithms that employ SOCP duality perform much better compared to the algorithms that use Lagrangian duality in the GBD setting. We believe that this is partly because solving the dual subproblem is faster than solving the primal subproblem. The GBD algorithms perform better than EF only in a small number of instances. Their advantage over the EF is that they do not encounter memory problems for the instances with a large number of scenarios.

6. Solving the Subproblems with a Cutting Plane Approach

To solve the DSP, we regenerate all possible feasible paths using an algorithm by Byers and Waterman (1984). As the network size and the tolerance level get larger, we may encounter memory problems. Incorporating uncertainty with a large number of scenarios certainly makes the problem more difficult. In this case, it may be advantageous not to work with large models that involve variables for all possible paths and to generate these variables when required within a column-generation framework to solve the subproblem. As we solve the dual of the subproblem, we employ a cutting plane approach to generate the constraints related to these paths when needed.

For a given scenario $\omega \in \Omega$, we begin solving the DSP with a subset of constraints (39). We use the constraints associated with the set of shortest paths between every demand node and functioning shelters in the scenario $\omega$. At every iteration, we determine whether there exists a path for which the respective constraint in the DSP is violated. If such paths are detected, we add the respective constraints to the DSP and repeat the procedure until there is no violated constraint.

For any given scenario $\omega \in \Omega$ and demand point-shelter pairs $r \in O, s \in F(\omega)$, finding a path that violates
constraint (39) requires solving the following separation problem:

**Separation Problem**

\[
\Phi_{rs}(\omega) = \max \left\{ z_r(\omega) + \Gamma_{rs}(\omega) + \sum_{i \in F(\omega)} \Lambda_i(\omega) f_i \right. \\
\left. \quad - \sum_{a \in A(\omega)} w_a(\omega) \psi_a(\omega) x_a \right\}
\]

\[\text{s.t.} \sum_{a \in \delta^-(i)} x_a - \sum_{a \in \delta^+(i)} x_a \begin{cases} 
1 & i = r, \\
0 & i \in N \setminus (r, s), \\
-1 & i = s,
\end{cases} \quad \forall i \in F(\omega), \tag{59}
\]

\[\sum_{a \in \delta^-(i)} l_a x_a \leq (1 + \lambda) d_{ri}^*(\omega) + M f_i, \quad \forall i \in F(\omega), \tag{60}
\]

\[\sum_{a \in \delta^+(i)} l_a x_a \leq (1 + \lambda) d_{ii}^*(\omega), \quad \forall i \in F(\omega), \tag{61}
\]

\[\chi_a \in \{0, 1\}, \quad \forall a \in A(\omega), \quad f_i \in \{0, 1\}, \quad \forall i \in F(\omega), \tag{62}
\]

where \(\delta^-(i)\) and \(\delta^+(i)\) denote the sets of incoming and outgoing arcs of node \(i \in N\), \(x_a\) is 1 if arc \(a\) is used in an optimal solution and 0 otherwise, \(M\) is a very big number, \(f_i\) is 1 if \(\sum_{a \in A(\omega)} l_a x_a > (1 + \lambda) d_{ri}^*(\omega)\) and 0 otherwise, and \(d_{ii}^*(\omega)\) is the shortest path distance to the closest open shelter from origin \(r\) in scenario \(\omega \in \Omega\). If \(\Phi_{rs}(\omega) > 0\), then the constraint related to the path defined by an optimal vector \(\hat{x}\) is added to the DSP, and if \(\Phi_{rs}(\omega) \leq 0\) for all \(r \in O, s \in F(\omega)\), then there is no violated constraint to be added to the DSP in scenario \(\omega \in \Omega\)—i.e., the DSP \((\hat{y}, \omega)\) is solved to optimality.

Considering that the separation problem is solved in every iteration and for every subproblem of the BD algorithm, our early experiments showed that solving the separation problem as it is is not effective in terms of the CPU times. For that reason, we propose the following methodology to solve the separation problem.

For a given \(r \in O\), we compute \(d_{ri}^*(\omega)\) for all \(i \in F(\omega)\) in scenario \(\omega \in \Omega\) and sort them in ascending order. Then, for a given scenario \(\omega \in \Omega\) and demand point-shelter pair \(r \in O, s \in F(\omega)\), we create the scalar vectors \(f = \{0, 0, 0, 0, 0, \ldots, 0\}, \quad f = \{1, 0, 0, 0, 0, \ldots, 0\}, \ldots, \quad f = \{1, 1, 1, 1, 1, 0, \ldots, 0\}\), where the final 1 in the sequence is determined by the index of max \(\{d_{ri}^*(\omega)\}\) \(< d_{ri}^*(\omega)\). The scalar vector \(f\) replaces the binary variable \(f_i\) in the separation problem. We modify constraint (61) as \(\sum_{a \in A(\omega)} l_a x_a \leq (1 + \lambda) d_{ri}^*(\omega)\), where \(i = \arg \min \{d_{ri}^*(\omega) : f_i = 0\}\). Then, the separation problem requires solving a number of single resource-constrained shortest path problems (CSP) over a graph with arc costs equal to \(w_a(\omega) \psi_a(\omega)\) and arc resources as the geographical distances. The CSP is either solved for every vector \(f\) in sequence as defined above or it terminates as soon as a path with the associated constraint is detected. We employ the algorithms by Santos, Coutinho-Rodrigues, and Current (2007) and Yen (1971) to solve the CSP.

Before we start the BD algorithm, we find five shortest paths for every scenario \(\omega \in \Omega, r \in O\), \(s \in F(\omega)\) using the approach proposed by Yen (1971). In the separation procedure, we first check if there exists a violated constraint for any of these paths. If such a path is detected, then the corresponding constraint is added to the DSP. If no such path is detected, then the separation problem is solved exactly.

### 7. Computational Study

In our early experiments, we observed that the solution times of the classical BD algorithm where the MP is solved to optimality as an integer problem at each iteration is much worse than our implementation in which we employ a lazy constraint callback. Likewise, the versions of our BD algorithm that employ the aggregate cut rather than a multicut strategy and/or solving the primal subproblem instead of the dual subproblem to generate the optimality cuts are not promising in terms of CPU times either. Hence, we do not report our computational studies for these algorithms.

Among all of the versions on which we experimented, there are four algorithms that solve our problem quickly. In all versions of our BD algorithm, we solve the dual subproblem to generate the optimality cuts implementing the lazy constraint callback feature of ILOG CPLEX, and we adopt a multicut strategy. In the first three algorithms, the subproblem is solved by pregenerating all possible paths, and the last algorithm uses the cutting plane framework to solve the dual subproblem. Below, we summarize these algorithms.

**Algorithm BD.** This is the basic version of our BD algorithm, in which we employ the dual subproblem, lazy constraint callback, and a multicut strategy.

**Algorithm BD\_IC.** In this algorithm, we take the advantage of the EMW problem that can be solved independently from the dual subproblem and solve the EMW problem only once before we begin solving the MP to generate an initial set of valid cuts. We set the core point \(y^0 = 1\) to generate the initial set of cuts.

**Algorithm BD\_SL.** In this version, the idea is to solve the original dual subproblem optimally with the first priority and, among alternative optimal solutions to this problem, to maximize (56). To achieve this, we use a weighted sum of (38) and (56) with a weight vector of \((1, \zeta)\) as our modified objective function of the dual subproblem. We begin with a core point \(y^0 = 1\) and update it at every iteration \(k\) using the equation, \(y^{k+1}_i = \frac{1}{2} y^{k+1}_i + \frac{1}{2} y^{k+1}_i\). We take \(\zeta = 10^{-11}\).
Algorithm BD_CP. This is the basic version of our BD algorithm, in which we employ the dual subproblem, lazy constraint callback, and a multicut strategy, but where we solve the dual subproblem by implementing a cutting plane approach instead of pregenerating all constraints for all possible paths.

7.1. Scenario Generation

We generated two of our instances—i.e., Istanbul Anatolian and Istanbul European—using real data from a disaster prevention and mitigation study conducted by the Istanbul Metropolitan Municipality (IMM) and the Japan International Cooperation Agency (JICA) (IMM-JICA 2002) for earthquake preparedness and response planning for an impending major earthquake in Istanbul, Turkey, as in Bayram and Yaman (2015). Turkey is among the countries where tsunamis, landslides, and fires have been observed as secondary disasters following a major earthquake (Marano, Wald, and Allen 2010). One of the findings in that report is that there does not exist an emergency evacuation system in Istanbul and that it is imperative that an evacuation system be established to mitigate human casualties due to second or third aftershocks and secondary disasters following the earthquake. For that reason, we assume an emergency evacuation in the sense that we are evacuating people to protect them from the impact of aftershocks and the secondary disasters.

The IMM–JICA report gives details of four scenarios for the earthquake, scenarios A–D, where scenario A is referred to as the most probable scenario and scenario C is the worst case scenario. We take scenario A as the base (nominal) scenario since the evacuation demand specific to each origin node is only given for that scenario in the report. The demand for other scenarios is generated by comparing the length of the fault lines expected to be broken and the expected magnitude of the earthquake in these scenarios to those of the base scenario A.

In the report, Istanbul City is divided into smaller disaster regions using a spatial risk analysis. For each scenario, five different risk zones are determined. We assign disruption/degradation probabilities for each zone in the base scenario and find these probabilities for risk zones of the other scenarios by a similar comparison stated above. The highest probability of disruption is assigned to risk zone 1, as it is closest to the fault line, and the lowest one to risk zone 5. The road segments and shelters in the network are classified into different sets based on the risk zone in which they are located. Given a particular scenario, we randomly determine whether an arc (road segment) is damaged by considering the risk zone in which it is located and the probability of disruption/degradation assigned to this zone. If the arc is damaged, we again randomly specify the amount of capacity it lost by the number of lanes. If there is partial damage, then this arc is degraded; and if it has lost all of its lanes, then this road segment is disrupted and unpassable. We follow a similar approach for the disruption of the shelters. We generate the scenarios in a random fashion using the characteristics of the four main scenarios. Please see Bayram and Yaman (2015) for more detail on how we generate the scenarios.

We downloaded the P-median1 instance from Beasley (1990). We created the demand for each origin node randomly between 1,000 and 2,000 (vehicles). We also generated potential shelter sites randomly on the network for that instance. We assign the arcs and shelters into risk zones assuming there are four basic scenarios as in the Istanbul instances.

We generate instances in such a way that for every origin, there exists at least one reachable shelter—i.e., everyone can be evacuated.

7.2. Computational Testing

The specifics of the instances used in the computational study are shown in Table 1. Here, $|O − F|$ is the number of origin–destination pairs that are connected with a directed path in the original undisrupted graph. We perform our computational tests on a workstation with 2 Xeon E5-2609 4C 2.4 GHz CPUs and 96 GB RAM by using Java ILOG CPLEX version 12.5.1.

In Tables 2–6, we compare the computational efficiencies of the three BD algorithms and the EF for different values of $p$, $\lambda$, and $|\Omega|$ (number of scenarios). For each instance, we report the number of iterations the BD algorithm performed, the number of optimality cuts added, and the solution times. Here, the number of iterations refers to the number of times an incumbent solution is passed to subproblems—i.e., the number of times subproblems are solved. We set a time limit of five hours for our experiments. If the problem is not solved to optimality within the time limit, then we report the remaining gap in parentheses in the column of solution times. If a solution cannot be obtained within the time limit, we report such a situation as “No Solution (NS).” We are able to solve the Istanbul Anatolian instances with up to 1,000 scenarios without any memory problems using any of the BD algorithms. On the other hand, using the EF with CPLEX does not generate a solution within five hours for the instances with 1,000 scenarios. All of the BD algorithms perform much better compared to the EF in terms of the CPU times. The BD algorithm performs at least 1.85 times better than the EF; this rate increases up to 19.16 and marks 5.53 on

**Table 1.** Specifics of the Instances Used in the Computational Study

| Instance       | $|N|$ | $|A|$ | $|O|$ | $|F|$ | Total demand | $|O − F|$ |
|----------------|-----|-----|-----|-----|-------------|-------|
| Istanbul Anatolian | 50  | 146 | 13  | 17  | 83,133      | 221   |
| Istanbul European  | 80  | 238 | 25  | 32  | 272,900     | 800   |
| P-median1         | 100 | 396 | 85  | 15  | 123,388     | 1,275 |

Electronics, Communications and Systems Engineering (EDUCOM)
average, not including the actual solution times of the EF for the instances with 1,000 scenarios since these instances hit the time limit. The BD_SL algorithm performs even better, with 2.03, 19.49, and 5.77 values as the minimum, maximum, and average rates, respectively. These numbers for the BD_IC algorithm are 2.23, 25.80, and 5.66, respectively. The BD_SL algorithm generally performs better than the BD algorithm for the Istanbul Anatolian case. In 53 of the 75 total instances for the Istanbul Anatolian network, BD_SL has smaller CPU
times and generally ends in a smaller number of iterations adding a smaller number of optimality cuts. This rate is 41 of 75 total instances for the BD algorithm. The BD algorithm performs better than the BD_SL algorithm in 34 of 75 instances in terms of CPU times.

For the Istanbul European network, EF runs into memory problems for the instances with $|\Omega| = 50$, $\lambda = 0.2$, $|\Omega| = 100$, $\lambda \geq 0.15$, and $|\Omega| = 1,000$, $\lambda \geq 0.1$. When the number of scenarios is 100, for $\lambda = 0.05$ and $p = 10$, the EF can obtain a solution with a gap of 99.5% within the five-hour time limit, and seven out of 25 instances with $|\Omega| = 100$ are of this sort. When $\lambda = 0.1$ and $p = 10$, EF cannot obtain a solution at the root node within the time limit, and three out of 25 instances with $|\Omega| = 100$ encounter this problem. For all of the instances that the EF does not encounter the memory issue when the number of scenarios is 1,000, there is no solution obtained at the root node within the time limit. BD algorithms run into memory problems for the instances with $|\Omega| = 100$, $\lambda = 0.2$ and $|\Omega| = 1,000$, $\lambda \geq 0.15$. Excluding the instances with memory problems, the BD, BD_SL, and BD_IC algorithms solve all of the instances to optimality except for six, five, and six instances, respectively, with $|\Omega| = 1,000$ where they run into the time limit, the majority of the gaps being small. Each of the BD algorithms outperform the EF in terms of the memory problems and the CPU times. Except for the three instances with $|\Omega| = 50$ where EF performs better when $p$ is relatively large and excluding the instances with memory problems and the ones that hit the time limit, the BD algorithm performs at least 1.09 times better than the EF, this rate increases up to 9.22 and marks 3.97 on average. For the BD_SL algorithm, these numbers are 1.14, 8.61, and 3.99 as the minimum, maximum, and average rates, respectively, and this algorithm outperforms EF in 57 of 60 instances. These numbers are 1.07, 8.07, and 3.54, respectively, for BD_IC, taking into account 58 of 60 instances where BD_IC has better CPU times. In 27 of the 60 instances, the BD_SL algorithm performs relatively better compared to BD in terms of CPU times. This rate is 21–60 for the BD_IC algorithm; and BD_IC performs better than BD_SL in 21 of 60 instances in terms of CPU times. The algorithms outperform each other for different instances in terms of the number of iterations. The BD algorithm adds generally a smaller number of optimality cuts with a small number of scenarios, but starting with larger $\lambda$ values when the number of scenarios is 100, and for a larger number of scenarios, BD_SL adds generally a lower number of optimality cuts.

We observe similar results for the P-median network instances and report them in Tables 5 and 6. In more than two-thirds of the instances with 50 scenarios, EF hits the time limit with no solution at the
Bayram and Yaman: Shelter Location and Evacuation Route Assignment Under Uncertainty
Transportation Science, Articles in Advance, pp. 1–31, © 2017 INFORMS

<table>
<thead>
<tr>
<th>$\omega$</th>
<th>$p$</th>
<th>$\lambda$</th>
<th>No. of iter.</th>
<th>No. of cuts</th>
<th>Sol. time</th>
<th>BD</th>
<th>BD SL</th>
<th>BD IC</th>
<th>EF (SCSO)</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>10</td>
<td>0</td>
<td>114</td>
<td>4,930</td>
<td>603</td>
<td>114</td>
<td>4,930</td>
<td>601</td>
<td>136</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>0</td>
<td>131</td>
<td>5,553</td>
<td>559</td>
<td>133</td>
<td>5,646</td>
<td>583</td>
<td>211</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>0</td>
<td>184</td>
<td>7,630</td>
<td>798</td>
<td>174</td>
<td>7,297</td>
<td>748</td>
<td>207</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
<td>0</td>
<td>210</td>
<td>8,823</td>
<td>1,076</td>
<td>224</td>
<td>9,006</td>
<td>1,119</td>
<td>189</td>
</tr>
<tr>
<td>50</td>
<td>30</td>
<td>0</td>
<td>50</td>
<td>1,843</td>
<td>249</td>
<td>50</td>
<td>1,843</td>
<td>241</td>
<td>46</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>0.05</td>
<td>108</td>
<td>4,916</td>
<td>612</td>
<td>119</td>
<td>5,324</td>
<td>655</td>
<td>121</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>0.05</td>
<td>119</td>
<td>5,084</td>
<td>539</td>
<td>127</td>
<td>5,237</td>
<td>570</td>
<td>136</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>0.05</td>
<td>167</td>
<td>6,692</td>
<td>747</td>
<td>164</td>
<td>6,451</td>
<td>734</td>
<td>189</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
<td>0.05</td>
<td>158</td>
<td>5,895</td>
<td>808</td>
<td>153</td>
<td>5,874</td>
<td>784</td>
<td>158</td>
</tr>
<tr>
<td>50</td>
<td>30</td>
<td>0.05</td>
<td>20</td>
<td>716</td>
<td>106</td>
<td>20</td>
<td>716</td>
<td>99</td>
<td>21</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>0.1</td>
<td>145</td>
<td>6,661</td>
<td>1,107</td>
<td>151</td>
<td>6,904</td>
<td>1,168</td>
<td>156</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>0.1</td>
<td>92</td>
<td>4,140</td>
<td>586</td>
<td>94</td>
<td>4,233</td>
<td>586</td>
<td>158</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>0.1</td>
<td>168</td>
<td>6,782</td>
<td>1,071</td>
<td>148</td>
<td>6,228</td>
<td>904</td>
<td>133</td>
</tr>
<tr>
<td>50</td>
<td>25</td>
<td>0.1</td>
<td>144</td>
<td>6,064</td>
<td>1,039</td>
<td>144</td>
<td>6,084</td>
<td>1,025</td>
<td>126</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>0.05</td>
<td>24</td>
<td>781</td>
<td>179</td>
<td>24</td>
<td>781</td>
<td>171</td>
<td>16</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>0.05</td>
<td>139</td>
<td>6,367</td>
<td>2,569</td>
<td>141</td>
<td>6,406</td>
<td>2,625</td>
<td>129</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>0.05</td>
<td>100</td>
<td>4,433</td>
<td>1,439</td>
<td>102</td>
<td>4,531</td>
<td>1,472</td>
<td>164</td>
</tr>
<tr>
<td>30</td>
<td>25</td>
<td>0.05</td>
<td>153</td>
<td>6,208</td>
<td>2,206</td>
<td>135</td>
<td>5,567</td>
<td>1,926</td>
<td>186</td>
</tr>
<tr>
<td>30</td>
<td>30</td>
<td>0.05</td>
<td>14</td>
<td>546</td>
<td>279</td>
<td>14</td>
<td>546</td>
<td>252</td>
<td>9</td>
</tr>
<tr>
<td>30</td>
<td>35</td>
<td>0.05</td>
<td>144</td>
<td>6,781</td>
<td>10,906</td>
<td>157</td>
<td>7,464</td>
<td>12,025</td>
<td>193</td>
</tr>
<tr>
<td>30</td>
<td>40</td>
<td>0.05</td>
<td>99</td>
<td>4,380</td>
<td>6,439</td>
<td>102</td>
<td>4,534</td>
<td>6,626</td>
<td>179</td>
</tr>
<tr>
<td>30</td>
<td>45</td>
<td>0.05</td>
<td>156</td>
<td>6,464</td>
<td>10,433</td>
<td>152</td>
<td>6,659</td>
<td>9,834</td>
<td>183</td>
</tr>
<tr>
<td>25</td>
<td>25</td>
<td>0.2</td>
<td>87</td>
<td>5,320</td>
<td>6,375</td>
<td>106</td>
<td>4,074</td>
<td>7,782</td>
<td>134</td>
</tr>
<tr>
<td>25</td>
<td>30</td>
<td>0.2</td>
<td>6</td>
<td>199</td>
<td>443</td>
<td>6</td>
<td>199</td>
<td>451</td>
<td>7</td>
</tr>
</tbody>
</table>

Average: 115, 4,813, 2,140, 115, 4,805, 2,197, 132, 5,469, 2,805 | Out of memory |

Average: 156, 3,333, 2,800, 167, 14,391, 2,942, 176, 14,893, 3,319

Table 4. Comparison of Different Algorithms With Respect to Computational Effectiveness (Istanbul European Instances)
Table 5. Comparison of Different Algorithms With Respect to Computational Effectiveness (P-median1 Instances)

<table>
<thead>
<tr>
<th>[Ω]</th>
<th>p</th>
<th>λ</th>
<th>No. of iter.</th>
<th>No. of cuts</th>
<th>Sol. time</th>
<th>No. of iter.</th>
<th>No. of cuts</th>
<th>Sol. time</th>
<th>No. of iter.</th>
<th>No. of cuts</th>
<th>Sol. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>50</td>
<td>5</td>
<td>0</td>
<td>187</td>
<td>8,841</td>
<td>1,468</td>
<td>187</td>
<td>8,841</td>
<td>1,462</td>
<td>208</td>
<td>9,736</td>
<td>1,694</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
<td>152</td>
<td>7,244</td>
<td>982</td>
<td>152</td>
<td>7,244</td>
<td>985</td>
<td>177</td>
<td>8,261</td>
<td>1,170</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
<td>126</td>
<td>5,960</td>
<td>937</td>
<td>127</td>
<td>6,010</td>
<td>929</td>
<td>152</td>
<td>7,086</td>
<td>1,138</td>
</tr>
<tr>
<td>50</td>
<td>11</td>
<td>0</td>
<td>68</td>
<td>2,691</td>
<td>403</td>
<td>68</td>
<td>2,691</td>
<td>398</td>
<td>68</td>
<td>2,729</td>
<td>419</td>
</tr>
<tr>
<td>50</td>
<td>12</td>
<td>0</td>
<td>59</td>
<td>2,180</td>
<td>408</td>
<td>59</td>
<td>2,180</td>
<td>400</td>
<td>44</td>
<td>1,745</td>
<td>310</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
<td>238</td>
<td>11,224</td>
<td>1,930</td>
<td>236</td>
<td>11,224</td>
<td>1,930</td>
<td>191</td>
<td>8,891</td>
<td>1,615</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>0</td>
<td>260</td>
<td>12,339</td>
<td>1,790</td>
<td>250</td>
<td>11,676</td>
<td>1,710</td>
<td>247</td>
<td>11,656</td>
<td>1,730</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>0</td>
<td>181</td>
<td>8,823</td>
<td>1,437</td>
<td>180</td>
<td>8,772</td>
<td>1,444</td>
<td>171</td>
<td>8,259</td>
<td>1,390</td>
</tr>
<tr>
<td>50</td>
<td>11</td>
<td>0</td>
<td>63</td>
<td>2,636</td>
<td>430</td>
<td>63</td>
<td>2,636</td>
<td>420</td>
<td>54</td>
<td>2,309</td>
<td>379</td>
</tr>
<tr>
<td>50</td>
<td>12</td>
<td>0</td>
<td>38</td>
<td>1,557</td>
<td>308</td>
<td>38</td>
<td>1,557</td>
<td>299</td>
<td>41</td>
<td>1,595</td>
<td>332</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
<td>283</td>
<td>13,477</td>
<td>2,518</td>
<td>282</td>
<td>13,426</td>
<td>2,488</td>
<td>289</td>
<td>13,775</td>
<td>2,586</td>
</tr>
<tr>
<td>50</td>
<td>1</td>
<td>0</td>
<td>300</td>
<td>14,284</td>
<td>2,202</td>
<td>316</td>
<td>15,021</td>
<td>2,345</td>
<td>327</td>
<td>15,425</td>
<td>2,444</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>0</td>
<td>227</td>
<td>10,966</td>
<td>1,950</td>
<td>231</td>
<td>11,173</td>
<td>1,956</td>
<td>228</td>
<td>10,920</td>
<td>1,978</td>
</tr>
<tr>
<td>50</td>
<td>11</td>
<td>0</td>
<td>94</td>
<td>4,060</td>
<td>690</td>
<td>92</td>
<td>3,967</td>
<td>677</td>
<td>96</td>
<td>4,123</td>
<td>720</td>
</tr>
<tr>
<td>50</td>
<td>12</td>
<td>0</td>
<td>43</td>
<td>1,912</td>
<td>371</td>
<td>43</td>
<td>1,912</td>
<td>363</td>
<td>47</td>
<td>2,044</td>
<td>415</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
<td>0</td>
<td>449</td>
<td>21,889</td>
<td>4,208</td>
<td>448</td>
<td>21,819</td>
<td>4,187</td>
<td>470</td>
<td>22,641</td>
<td>4,484</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>0</td>
<td>431</td>
<td>25,493</td>
<td>4,175</td>
<td>494</td>
<td>23,786</td>
<td>3,834</td>
<td>506</td>
<td>24,419</td>
<td>3,910</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>0</td>
<td>385</td>
<td>18,855</td>
<td>3,424</td>
<td>329</td>
<td>16,176</td>
<td>2,881</td>
<td>339</td>
<td>16,243</td>
<td>3,056</td>
</tr>
<tr>
<td>50</td>
<td>11</td>
<td>0</td>
<td>121</td>
<td>5,497</td>
<td>897</td>
<td>120</td>
<td>5,458</td>
<td>888</td>
<td>102</td>
<td>4,650</td>
<td>762</td>
</tr>
<tr>
<td>50</td>
<td>12</td>
<td>0</td>
<td>57</td>
<td>2,593</td>
<td>493</td>
<td>57</td>
<td>2,593</td>
<td>490</td>
<td>56</td>
<td>2,580</td>
<td>500</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>0</td>
<td>456</td>
<td>26,286</td>
<td>5,431</td>
<td>555</td>
<td>26,716</td>
<td>5,515</td>
<td>531</td>
<td>25,462</td>
<td>5,365</td>
</tr>
<tr>
<td>50</td>
<td>7</td>
<td>0</td>
<td>621</td>
<td>29,989</td>
<td>5,036</td>
<td>647</td>
<td>31,234</td>
<td>5,234</td>
<td>628</td>
<td>30,194</td>
<td>5,091</td>
</tr>
<tr>
<td>50</td>
<td>8</td>
<td>0</td>
<td>480</td>
<td>23,369</td>
<td>4,523</td>
<td>476</td>
<td>23,199</td>
<td>4,472</td>
<td>470</td>
<td>22,716</td>
<td>4,455</td>
</tr>
<tr>
<td>50</td>
<td>11</td>
<td>0</td>
<td>191</td>
<td>8,740</td>
<td>1,454</td>
<td>183</td>
<td>8,395</td>
<td>1,393</td>
<td>194</td>
<td>8,887</td>
<td>1,505</td>
</tr>
<tr>
<td>50</td>
<td>12</td>
<td>0</td>
<td>106</td>
<td>4,825</td>
<td>938</td>
<td>119</td>
<td>5,386</td>
<td>1,055</td>
<td>132</td>
<td>5,869</td>
<td>1,198</td>
</tr>
</tbody>
</table>

Average: 232, 11,033, 1,938, 230, 10,924, 1,910, 231, 10,889, 1,946

EF (SCSO)
Table 6. Comparison of Different Algorithms With Respect to Computational Effectiveness (P-median1 Instances)

<table>
<thead>
<tr>
<th>[Ω]</th>
<th>p</th>
<th>λ</th>
<th>No. of iter.</th>
<th>BD</th>
<th>No. of cuts</th>
<th>BD</th>
<th>Sol. time</th>
<th>BD_SL</th>
<th>No. of cuts</th>
<th>BD_SL</th>
<th>Sol. time</th>
<th>BD_IC</th>
<th>No. of iter.</th>
<th>BD_IC</th>
<th>No. of cuts</th>
<th>BD_IC</th>
<th>Sol. time</th>
<th>Sol. time</th>
</tr>
</thead>
<tbody>
<tr>
<td>1,000</td>
<td>5</td>
<td>0</td>
<td>106</td>
<td>95,538</td>
<td>(0.94)</td>
<td>107</td>
<td>99,711</td>
<td>(0.93)</td>
<td>102</td>
<td>97,485</td>
<td>(0.93)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>7</td>
<td>0</td>
<td>131</td>
<td>126,418</td>
<td>(0.844)</td>
<td>131</td>
<td>126,438</td>
<td>(0.842)</td>
<td>126</td>
<td>115,339</td>
<td>(0.833)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>8</td>
<td>0</td>
<td>114</td>
<td>105,536</td>
<td>(0.74)</td>
<td>114</td>
<td>106,677</td>
<td>(0.76)</td>
<td>110</td>
<td>95,456</td>
<td>(0.76)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>11</td>
<td>0</td>
<td>69</td>
<td>57,008</td>
<td>8,977</td>
<td>85</td>
<td>56,050</td>
<td>11,030</td>
<td>63</td>
<td>54,130</td>
<td>8,354</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>12</td>
<td>0</td>
<td>43</td>
<td>35,423</td>
<td>6,745</td>
<td>43</td>
<td>35,423</td>
<td>6,622</td>
<td>51</td>
<td>38,104</td>
<td>7,877</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>5</td>
<td>0.05</td>
<td>103</td>
<td>94,581</td>
<td>(0.90)</td>
<td>102</td>
<td>91,572</td>
<td>(0.89)</td>
<td>98</td>
<td>90,580</td>
<td>(0.86)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>7</td>
<td>0.05</td>
<td>125</td>
<td>115,522</td>
<td>(0.725)</td>
<td>123</td>
<td>114,530</td>
<td>(0.728)</td>
<td>118</td>
<td>110,492</td>
<td>(0.696)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>8</td>
<td>0.05</td>
<td>105</td>
<td>101,459</td>
<td>(0.63)</td>
<td>105</td>
<td>100,524</td>
<td>(0.65)</td>
<td>101</td>
<td>89,618</td>
<td>(0.64)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>11</td>
<td>0.05</td>
<td>88</td>
<td>70,131</td>
<td>12,912</td>
<td>91</td>
<td>73,820</td>
<td>13,169</td>
<td>53</td>
<td>48,062</td>
<td>7,909</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>12</td>
<td>0.05</td>
<td>29</td>
<td>25,558</td>
<td>5,254</td>
<td>29</td>
<td>25,500</td>
<td>4,941</td>
<td>33</td>
<td>25,845</td>
<td>5,794</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>5</td>
<td>0.1</td>
<td>96</td>
<td>82,540</td>
<td>(0.90)</td>
<td>96</td>
<td>59,999</td>
<td>(0.88)</td>
<td>93</td>
<td>84,506</td>
<td>(0.88)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>7</td>
<td>0.1</td>
<td>115</td>
<td>105,405</td>
<td>(0.721)</td>
<td>114</td>
<td>104,415</td>
<td>(0.721)</td>
<td>113</td>
<td>103,449</td>
<td>(0.647)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>8</td>
<td>0.1</td>
<td>98</td>
<td>94,691</td>
<td>(0.58)</td>
<td>100</td>
<td>96,658</td>
<td>(0.58)</td>
<td>95</td>
<td>88,532</td>
<td>(0.63)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>11</td>
<td>0.1</td>
<td>108</td>
<td>91,185</td>
<td>16,644</td>
<td>107</td>
<td>91,667</td>
<td>16,739</td>
<td>93</td>
<td>83,599</td>
<td>14,777</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>12</td>
<td>0.1</td>
<td>45</td>
<td>37,479</td>
<td>8,432</td>
<td>45</td>
<td>37,479</td>
<td>8,132</td>
<td>46</td>
<td>38,462</td>
<td>8,631</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>5</td>
<td>0.15</td>
<td>88</td>
<td>76,436</td>
<td>(0.76)</td>
<td>89</td>
<td>80,668</td>
<td>(0.84)</td>
<td>88</td>
<td>85,528</td>
<td>(0.84)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>7</td>
<td>0.15</td>
<td>112</td>
<td>106,416</td>
<td>(0.619)</td>
<td>111</td>
<td>105,363</td>
<td>(0.619)</td>
<td>108</td>
<td>102,596</td>
<td>(0.604)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>8</td>
<td>0.15</td>
<td>94</td>
<td>86,544</td>
<td>(0.46)</td>
<td>95</td>
<td>85,581</td>
<td>(0.46)</td>
<td>93</td>
<td>87,536</td>
<td>(0.64)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>11</td>
<td>0.15</td>
<td>116</td>
<td>106,023</td>
<td>(0.308)</td>
<td>116</td>
<td>104,133</td>
<td>(0.279)</td>
<td>114</td>
<td>104,296</td>
<td>(0.241)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>12</td>
<td>0.15</td>
<td>61</td>
<td>51,500</td>
<td>11,345</td>
<td>62</td>
<td>52,465</td>
<td>11,125</td>
<td>55</td>
<td>45,690</td>
<td>10,282</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>5</td>
<td>0.2</td>
<td>80</td>
<td>68,461</td>
<td>(0.76)</td>
<td>86</td>
<td>72,391</td>
<td>(0.72)</td>
<td>81</td>
<td>78,411</td>
<td>(0.82)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>7</td>
<td>0.2</td>
<td>106</td>
<td>100,540</td>
<td>(0.519)</td>
<td>105</td>
<td>100,463</td>
<td>(0.52)</td>
<td>101</td>
<td>94,492</td>
<td>(0.613)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>8</td>
<td>0.2</td>
<td>82</td>
<td>79,596</td>
<td>(0.45)</td>
<td>89</td>
<td>86,679</td>
<td>(0.45)</td>
<td>87</td>
<td>81,536</td>
<td>(0.54)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>11</td>
<td>0.2</td>
<td>111</td>
<td>104,501</td>
<td>(0.285)</td>
<td>112</td>
<td>103,804</td>
<td>(0.291)</td>
<td>107</td>
<td>98,237</td>
<td>(0.319)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1,000</td>
<td>12</td>
<td>0.2</td>
<td>87</td>
<td>79,860</td>
<td>(0.21)</td>
<td>95</td>
<td>87,321</td>
<td>(0.21)</td>
<td>91</td>
<td>78,484</td>
<td>(0.21)</td>
<td>(NS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Average 92 82,565 15,762 93 81,364 14,544 86 77,577 14,571 (NS)

In Figure 1, we illustrate how the upper (UB) and lower (LB) bounds are updated across iterations for the BD, BD_SL, and BD_IC algorithms when the number of scenarios is 1,000, λ = 0.2, p = 10 and λ = 0.05, p = 30 for the Istanbul Anatolian and Istanbul European networks, respectively.

By solving the dual subproblem with a cutting plane approach, we aim to overcome the memory problems we encounter when the size of the network or the tolerance level gets larger. For small tolerance levels—i.e., when the number of feasible paths is small—pregenerating the feasible paths and constructing the constraints related to them generally brings an advantage in terms of the CPU times. As the tolerance level increases, the CPU times worsen and, at some point, memory issues are encountered. For that reason, in

Figure 1. (Color online) Iteration vs. UB and LB for Istanbul Anatolian and European Networks
Table 7. The Results for the BD_CP Algorithm (Istanbul European)

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
quickly as possible. Our method also determines the assignment of evacuees to shelters and routes. Clearly, once the disaster hits, these assignments can be modified by the evacuation management authority by using real-time information and dynamic evacuation management models.

In our work, we assume that all of the evacuees comply with the guidance the central authority gives; yet this may not always be the case, and there may not be full compliance. The evacuees that do not comply with the evacuation orders may try to minimize their individual travel time while the evacuation authority aims to minimize the total evacuation time. An interesting extension would be a game-theoretic approach where some percentage of the evacuees in the network do not comply with the evacuation orders, and the central manager tries to find the best strategy to evacuate the disaster region as soon as possible.

Acknowledgment

The research of the second author is supported by the Turkish Academy of Sciences.

References


