Transportation Research Part B 45 (2011) 1190–1211

Contents lists available at ScienceDirect

Transportation Research Part B

journal homepage: www.elsevier.com/locate/trb

Reliable logistics networks design with facility disruptions

Peng Peng[a], Lawrence V. Snyder[b,*], Andrew Lim[a,d], Zuli Liu[c]

[a] *Department of Management Sciences, City University of Hong Kong, Tat Chee Avenue, Kowloon, Hong Kong*
[b] *Department of Industrial and Systems Engineering, Lehigh University, 200 West Packer Avenue, Bethlehem, PA 18015, USA*
[c] *Department of Computer Science, Sun Yat-sen University, Guangzhou 510275, PR China*
[d] *School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, PR China*

A R T I C L E I N F O

Keywords:
Reliable network design
Facility disruptions
p-Robustness
Metaheuristics

A B S T R A C T

This paper studies a strategic supply chain management problem to design reliable networks that perform as well as possible under normal conditions, while also performing relatively well when disruptions strike. We present a mixed-integer programming model whose objective is to minimize the nominal cost (the cost when no disruptions occur) while reducing the disruption risk using the *p*-robustness criterion (which bounds the cost in disruption scenarios). We propose a hybrid metaheuristic algorithm that is based on genetic algorithms, local improvement, and the shortest augmenting path method. Numerical tests show that the heuristic greatly outperforms CPLEX in terms of solution speed, while still delivering excellent solution quality. We demonstrate the tradeoff between the nominal cost and system reliability, showing that substantial improvements in reliability are often possible with minimal increases in cost. We also show that our model produces solutions that are less conservative than those generated by common robustness measures.

1. Introduction

1.1. Motivation

Key players in the supply chain, including manufacturers, retailers and distributors, have realized the value of comprehensive, long-term network planning in which they make detailed plans for constructing new facilities, expanding distribution networks, partnering with new suppliers, and other important supply chain activities. Many mathematical models have been proposed to solve a variety of supply chain network design problems. See Magnanti and Wong (1984), Owen and Daskin (1998), Daskin et al. (2005), Meixell and Gargeya (2005) for extensive reviews of models, algorithms and applications.

However, most of the studies assume that facilities are always available. Carefully constructed plans from these models can be severely ruined if they fail to consider disruptions in the design phase and therefore lack countermeasures when disruptions do strike. The popularization of the "lean" concept, which allows minimum redundancy, and the development of global supply chains has further exacerbated the problem.

From the terrorist attacks of 9/11 to the catastrophic devastation caused by Hurricane Katrina, recent events (Barrionuevo and Deutsch, 2005; Latour, 2001; Mouawad, 2005) have shown higher risks arising from disruptions and have changed the facets of the modern business world. While many believe that our international supply chains are strong and reliable, in reality many are fragile and easily disrupted when the unexpected occurs. For example, in 2008, the Boeing company was forced to pay huge amounts in compensation for postponing the delivery of the Dreamliner 787 due to delays in supply

* Corresponding author.
E-mail addresses: adxypeng@cityu.edu.hk (P. Peng), larry.snyder@lehigh.edu (L.V. Snyder), lim.andrew@cityu.edu.hk (A. Lim), liuzuli@gmail.com (Z. Liu).

0191-2615/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.trb.2011.05.022

for some critical components (Bathgate and Hayashi, 2008). Most recently, a disastrous earthquake and the following tsunami have struck Japan, not only causing heavy casualties and property losses, but also halting the production in a broad spectrum of the country's industries, because of plant damage, transportation blockage or power outages. The global electronics industry will be expecting a large near-term supply shortage, since Japan is the major supplier for electronic components such as semiconductors, LCD panels, flash memory chips and so on (Clark and Takahashi, 2011).

Many potential threats can lead to facility disruptions, e.g., operational contingencies such as equipment failures or supplier discontinuities; natural disasters; industrial accidents; power outages; labor strikes; and terrorism. Although these disruption events may only lead to short-term facility contingencies, they can also cause not only serious operational consequences, such as higher transportation costs, order delays, inventory shortages, loss of market shares, and so on, but also extended negative financial effects. An empirical study by Hendricks and Singhal (2005) has shown that over the time period of 1989–2000, the abnormal stock returns of firms that have been affected by disruptions were nearly 40%. Evidence has also shown that these firms had a hard time recovering from the negative effects of disruptions and that their equity risk increased significantly around the announcement date. Similar findings are described by Hicks (2002).

In this paper, we study the problem of designing a supply chain network, which consists of supply, transshipment, and demand nodes. Once such a supply chain infrastructure is built, it will be very difficult and costly to modify the design. Therefore, it is important to design supply chain systems that attain continuity and efficiency in the presence of all sorts of disruptions from the start. Consider a simple supply chain depicted in Fig. 1, in which customers located in El Paso and San Antonio are to be served by potential supply nodes in San Francisco, Los Angeles, San Diego, Atlanta and Pittsburgh via several transshipment nodes. The annual fixed costs, amount of supplies and demands are shown in Table 1. The distances are calculated by highway travel distance. As shown in Fig. 2, the optimal design, without considering disruptions, has a cost of \$1,636,000 per year when there is no disruption. However, suppose the transshipment node in Phoenix, AZ is disrupted for some unexpected reasons, e.g, an industrial accident. In this case, the company is forced to fulfill the demand in El Paso from Pittsburgh, which results in a much higher cost of \$2,087,800, with an increase of \$451,800, or 27.6%.

Now suppose, instead, that we have carefully planned for disruption scenarios that may occur, and we have developed the new design shown in Fig. 3. The major difference is that we open a backup transshipment facility in Chino, CA. This salvages the west-coast supply route and reduces the cost after a disruption in Phoenix to \$1,758,300 (15.8% saving from the first solution). On the other hand, the new solution is more expensive under normal circumstances—\$1,742,900—but this is only 6.5% more expensive than the first solution. Indeed, in our numerical studies on larger instances (Section 4), we find that substantial improvements in reliability are often obtainable with minor increases in nominal cost.

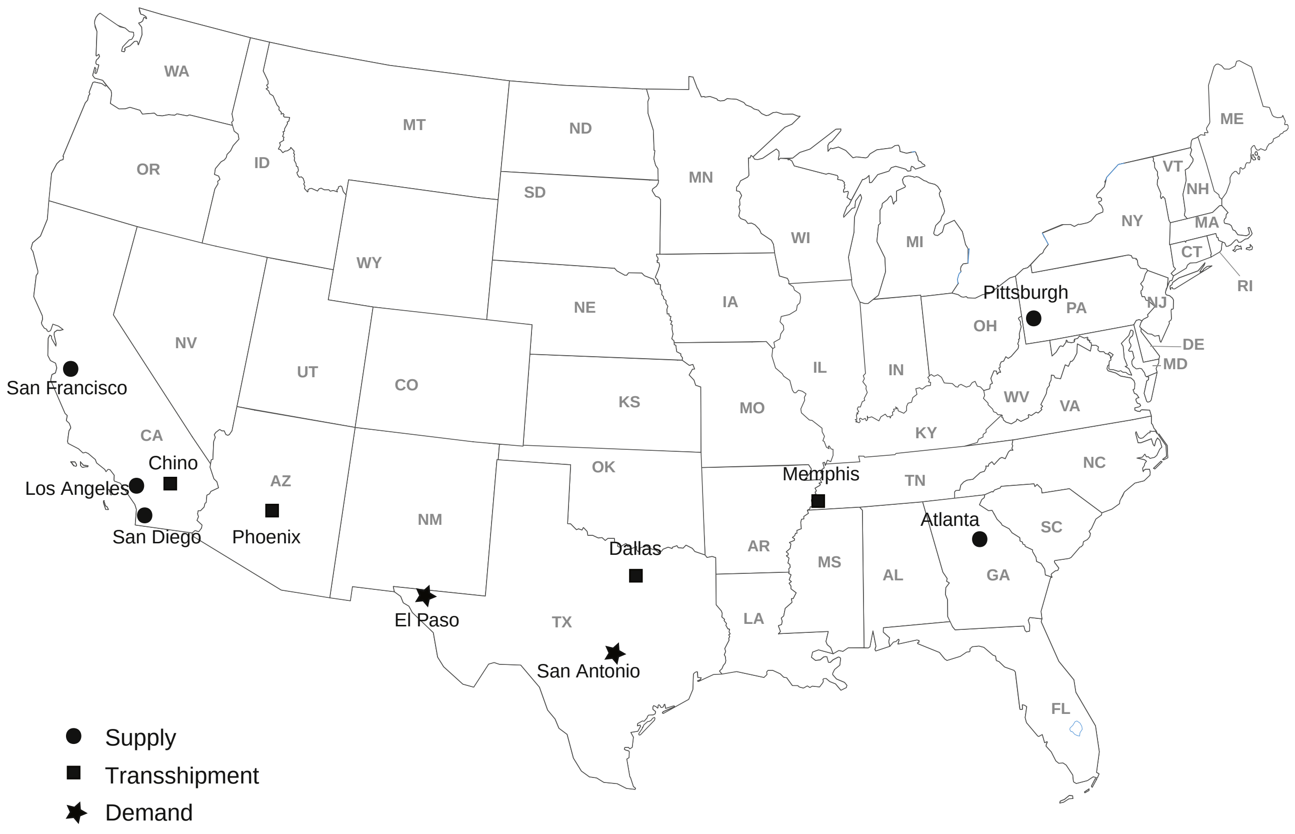

Fig. 1. The supply chain design problem.

Table 1
Data for numerical example.

Location	Fixed cost	Supply	Capacity
Pittsburgh, PA	15,000	1000	1000
Los Angeles,CA	11,900	100	100
San Diego, CA	18,200	100	100
San Francisco, CA	18,700	100	100
Atlanta, GA	18,600	200	200
Phoenix, AZ	13,200	0	300
Dallas, TX	11,700	0	1000
Memphis, TN	18,000	0	1000
Chino, CA	16,700	0	200
San Antonio, TX	N/A	−500	N/A
El Paso, TX	N/A	−300	N/A

1.2. Reliability vs. robustness

When studying supply chain risks arise from supply uncertainties, or more specifically, failure of suppliers, the two terms *reliability* and *robustness* are commonly used without rigorous distinction. A few papers, including Snyder (2003) and Bunschuh et al. (2006), have discussed the differences between these two terms. According to Bunschuh et al. (2006), robustness is defined as the system's ability to perform its intended function relatively well in the presence of failures of components or subsystems; while reliability is defined as the probability that a system or component performs its intended function within a given time horizon and environment. Their definitions are somehow restricted to the situations in which components of the supply chain system may fail during operation. In this paper, we adopt the definitions from Snyder (2003), in which the author considers a broader scope of uncertainties in supply chain management. A similar definition of reliability is adopted in Snyder and Daskin (2005) and Shen et al. (2007) to solve facility location problems.

A supply chain is *robust* if it performs well with respect to uncertain future conditions, such as demands, lead times, supplies and so on; while a supply chain is *reliable* if it performs well when parts of the system fail, for example, when a distribution center becomes unavailable due to poor weather. In other words, the term "robustness" is more broadly referred to solutions that perform well across a range of scenarios, in expected performance, worst-case performance, or any of a number of other measures that have appeared in the literature over the past. In contrast, "reliability" refers to a different approach to uncertainty in which we are hedging against failures in the system described by a given solution. In that sense, one can view robustness as concerned with uncertainty in the data, while reliability refers to uncertainty in the solution itself. Another way to view the distinction in the context of supply chain network design is that robustness is concerned with demand-side uncertainty (demands, costs, lead times, or any other aspect of the distribution of goods to customers), while reliability is concerned with supply-side uncertainty (availability of plants, distribution centers, supply capacity, or any other aspect related to production and distribution of the product).

On the other hand, *robust optimization* is a methodology which concerns with finding solutions that perform well with respect to uncertain future conditions. It has been widely applied in supply chain design problems with uncertain parameters. The goal of this work is to create a reliable, and yet economical logistics network for a manufacturer of a given final product. Uncertainty is modeled by a set of scenarios that specify a subset of facilities which becomes non-operational, or disrupted. In order to achieve this goal, robust optimization with p-robustness measure is applied.

1.3. Model overview

In this paper, we study the logistics network design problems (LNDP) with facility disruptions. The LNDP generalizes the classical capacitated facility location problem by considering multiple echelons, and makes decisions regarding the selection of suppliers, the locations of factories and warehouses, the assignment of suppliers to customers, and the flows of products through the network. Cordeau et al. (2006) introduce a general formulation for the LNDP. While most previous work does not take disruptions into consideration, the aim of our model is to design a supply chain network that performs well in the long run and is able to deal with short-term contingencies efficiently.

Most network design models with disruptions (see Section 1.4) assume that probabilistic information about the disruptions is known. However, since disruptions tend to be rare events arising from varied sources, historical data can be difficult to come by, and probabilities can be difficult to estimate. Instead, we opt to use robust optimization, which provides an alternative way of coping with uncertainty and does not require probabilistic information. Kouvelis and Yu (1997) motivate the use of a robustness approach for making decisions in uncertain environments. The aim of this approach is to find decisions that will have a reasonable objective value under any scenario specified by the modeler. Thus, there is no need to estimate the probability distribution of the random parameters.

Many robustness criteria have been proposed in the literature, and most have been applied to facility location and/or network design problems under demand uncertainty; see Snyder (2006) for a review. The most commonly applied robustness criteria are *minimax cost* and *minimax regret*, which minimize the maximum cost or regret among all scenarios. Mo and Harrison (2003) propose a conceptual framework of robust supply chain network design under uncertain demand that is

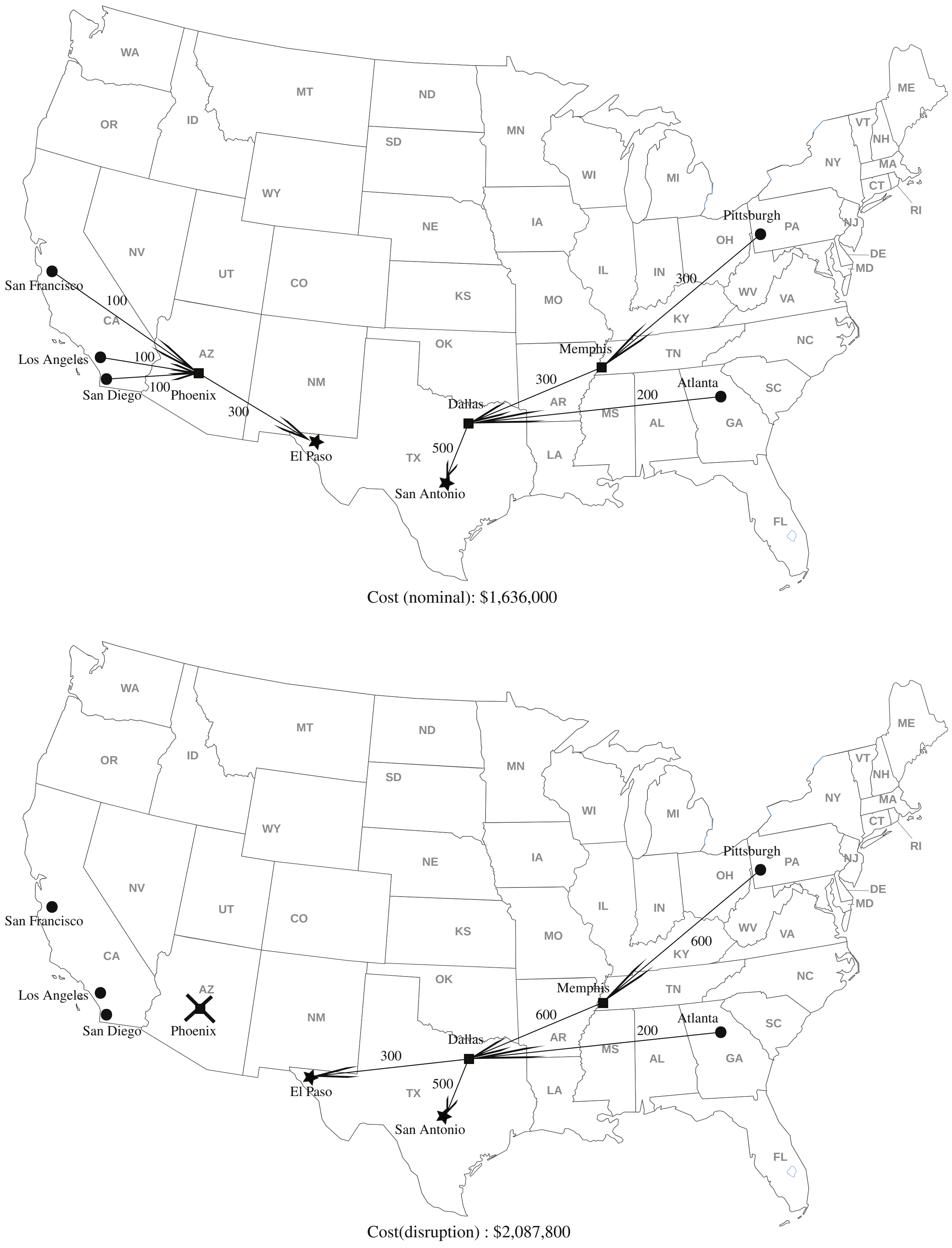

Fig. 2. Supply chain design without considering disruption.

modeled using a discrete probability distribution. They discuss various robustness measures and solution methods for solving the robust problems. The *p*-robustness measure, in which the relative regret in each scenario is required to be no more than a constant *p*, was first applied in a facility layout problem (Kouvelis et al., 1992) and later in an uncapacitated network

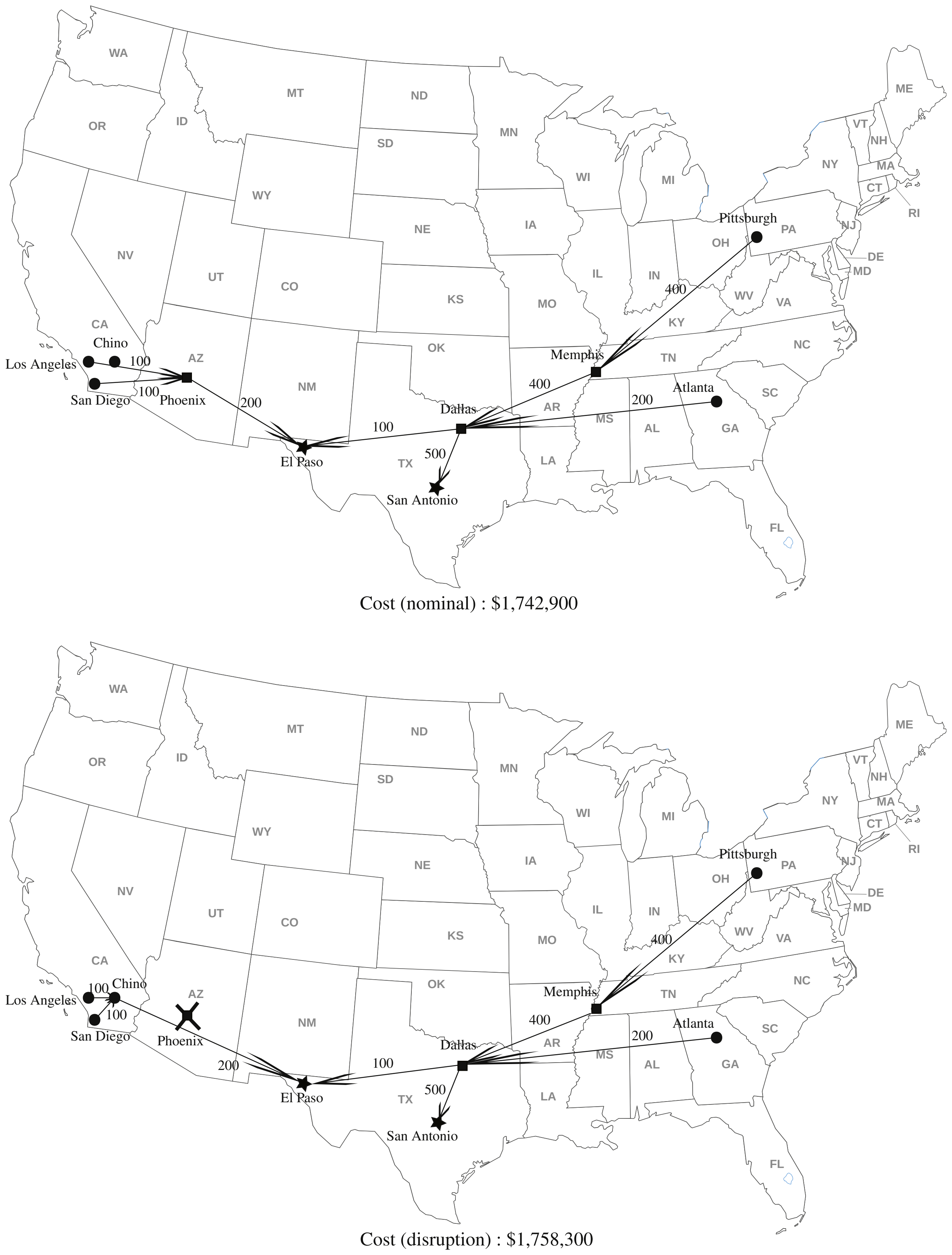

Fig. 3. Supply chain design considering disruption.

design problem (Gutiérrez et al., 1996). Snyder and Daskin (2006a) adopt the term "*p*-robust" to describe this measure; they formulate the stochastic *p*-robust *k*-median problem (*p*-S*k*MP) and the *p*-robust uncapacitated fixed charge location problem (*p*-UFLP), where the expected total cost is minimized.

We formulate a mixed-integer programming model which incorporates the p-robustness measure in the constraints. The objective is to minimize the total nominal cost, i.e, the total cost when there is no disruption, while restricting the relative regret in each scenario to be no more than p, for a constant $p > 0$. (See Section 2.2 for a more formal definition of p-robustness.) We use a scenario-based modeling approach, in which each scenario specifies a set of facilities that fails simultaneously. We wish to note that our modeling framework is flexible enough to accommodate other objective functions, such as expected cost. We will refer to our problem as the *p-robust logistics network design problem* (p-LNDP). To the best of our knowledge, we are the first to use the p-robustness measure to design reliable multi-echelon supply chain networks to ensure high-level performance during disruptions.

Our modeling approach was motivated by the needs of a Fortune 500 manufacturing company, for whom we applied our model. The company wished to limit the regret in each scenario in order to bound its exposure to risk, but was more concerned with minimizing the nominal cost rather than the expected cost or some other stochastic measures, since the nominal scenario is the most likely to occur and since disruption probabilities are difficult to estimate. Our research shows that, with little additional investment in infrastructure, the supply chain can be made significantly more resilient to disruptions, which makes planning for disruptions more attractive from a managerial prospective.

1.4. Literature review

An emerging body of research considers facility disruptions in the supply chain design and logistics literature. Tang (2006) highlights the need for designing supply chains that are resilient to disruptions using a variety of examples. He discusses robust strategies from a management perspective for mitigating supply chain disruptions, which will enable a supply chain to function smoothly and to continually serve customers during disruptions. Kleindorfer and Saad (2005) introduce a conceptual framework for disruption risk management in supply chains, which is based on the risk management literature and models of supply chain coordination.

One of the first mathematical models for facility location with unreliable suppliers is presented by Drezner (1987), who studies the unreliable p-median and (p,q)-center location problems, in which a facility has a given probability of becoming inactive. Snyder and Daskin (2005) formulate reliable versions of the uncapacitated fixed-charge location problem (UFLP) and the P-median problem, both of which aim to minimize a weighted sum of the nominal cost (the cost when no disruptions occur) and the expected cost accounting for random disruptions. For tractability, they make the strong assumption that all facilities have the same probability of failure. Berman et al. (2007), Cui et al. (2010), Li and Ouyang (2010), Lim et al. (2009), Lim et al. (2010), Shen et al. (2007), Snyder et al. (2006), and Zhan et al. (2008) all consider models similar to Snyder and Daskin's but relax the uniform-disruption-probability assumption using a variety of modeling approaches. Our model relaxes this assumption using a scenario-based stochastic programming approach. It also differs from the works cited above by considering general, multi-echelon network design problems (of which facility location problems are special cases) and by considering a robustness constraint rather than using an expected-cost objective.

Like the present paper, Bunschuh et al. (2006) study disruptions in a multi-echelon supply chain network. They create redundancy to improve the robustness of the network, by adding supplier sourcing constraints that give upper bounds on the total amount a customer can source from a single supplier. As a result, customers are forced to be assigned to multiple suppliers. However, this approach does not explicitly consider the possibility of disruption for each supplier. Snyder et al. (2006) introduce a wide range of strategic planning models for facility location and supply chain network design problems under the threat of disruptions, including a network design model that is similar to ours except that it uses an expected-cost objective rather than a robustness constraint. (They do not suggest solution methods for this model.) Scapparra and Cappanera (2010) consider a network design problem with disruptions, but with a shortest-path objective rather than a min-cost-flow-type objective.

Note that topics on network robustness and reliability have been discussed extensively in the literature on the design of survivable telecommunication networks (see, e.g., Soni and Pirkul (2000)). Survivable networks are defined as networks that are still functional after failure of certain network components. The network and cost structures of these networks, however, differ significantly from those of supply chain networks. The focus of survivable communication networks is to ensure the connectivity of the network in case of failure, which is generally not considered as a major objective in supply chain problems.

The rest of the paper is organized as follows. In Section 2, we formulate the p-LNDP. A GA-based hybrid metaheuristic is proposed in Section 3. We report our numerical results in Section 4, as well as managerial insights drawn from these results. Section 5 concludes our study.

2. Formulation of the *p*-LNDP

2.1. Notations

We first introduce the notations that will be used throughout the paper. Suppose that we have a general network $(\mathcal{V}, \mathcal{A})$. Let $\mathcal{V}_S, \mathcal{V}_T$, and $\mathcal{V}_D$ denote the sets of supply, transshipment, and demand nodes, respectively. For notational convenience, define $\mathcal{V}_0 = \mathcal{V}_S \cup \mathcal{V}_T$ as the set of all supply and transshipment nodes, which are the nodes for which open/close decisions are required; we sometimes refer to these nodes as "facilities." Let S be the set of scenarios, each of which specifies a set of facilities that are disrupted simultaneously. Denote $s = 0$ as the nominal scenario, i.e., the scenario with no disruptions.

Assume that if a customer's demand cannot feasibly be met, we incur a penalty that is proportional to the customer's demand. It is also possible to choose not to serve a customer if this penalty is smaller than the cost of serving the customer. The penalty may be interpreted as a lost-sales cost or as the cost of serving the customer from an outside supplier on an emergency basis. We model this contingency by assuming that $\mathcal{V}_S$ contains an "emergency facility" that has no fixed cost (and is therefore always open in the optimal solution), is never disrupted, and has infinite capacity. The unit transportation cost from this facility to each customer is equal to the unmet-demand penalty.

2.1.1. Parameters

Let:

- f_j = fixed cost to open facility $j \in \mathcal{V}_0$
- q_{ij} = unit transportation cost on arc $(i,j) \in \mathcal{A}$
- k_j = capacity of facility $j \in \mathcal{V}_0$
- b_j = supply of node $j \in \mathcal{V} : b_j \geqslant 0$ if $j \in \mathcal{V}_S$, representing the supply; $b_j = 0$ if $j \in \mathcal{V}_T$; and $b_j \leqslant 0$ if $j \in \mathcal{V}_D$, representing the demand
- a_{js} = 1 if facility $j \in \mathcal{V}_0$ is disrupted in scenario $s \in S$, 0 otherwise
- p = desired robustness level, $p \geqslant 0$
- c_s^* = optimal cost of scenario $s \in S$

Although we define a_{js} as a binary parameter, our model and heuristic apply equally well if these parameters may also be fractional, representing partial disruptions. The robustness coefficient p is the maximum allowable relative regret. For the sake of simplicity, we assume that p is the same for every scenario, although our model and heuristic still apply if this parameter is set differently for each scenario and denoted p_s, to account for the fact that different scenarios may hold different importance levels for the decision maker. The regret is computed using c_s^*, which is an input to the model. The c_s^* values can be computed by solving $|S|$ deterministic capacitated network design problems, one for each scenario, using any available method.

2.1.2. Decision variables

The decisions are made in two stages. In the first stage, we decide which supply nodes and transshipment nodes to open and assign flows through the network to satisfy customers' demands. Once some facilities are disrupted, i.e. a specific scenario has occurred, we may re-assign the flows but may not open new facilities. We define the following sets of decision variables:

- X_j = 1 if node $j \in \mathcal{V}_0$ is opened, 0 otherwise
- Y_{ijs} = amount of flow on arc $(i,j) \in \mathcal{A}$ in scenario $s \in S$

Note that the assignment variables ($\mathbf{Y}$) are dependent on s whereas the location variables ($\mathbf{X}$) are not, reflecting the two-stage nature of the problem.

2.2. Definition of p-robustness

We follow the definition of "p-robust" given by Snyder and Daskin (2006a):

Definition. For a given set S of scenarios, let P_s be the deterministic minimization problem for scenario s (a deterministic fixed-charge capacitated network design problem) and let c_s^* be the optimal objective value for P_s. Let $\mathbf{X}$ be a feasible vector of the location variables, $\mathbf{Y}$ be a feasible vector of the flow variables, and $c_s(\mathbf{X},\mathbf{Y})$ be the objective value of $(\mathbf{X},\mathbf{Y})$ in scenario s. Then $(\mathbf{X},\mathbf{Y})$ is called *p-robust* if for all $s \in S$,

$$\frac{c_s(\mathbf{X},\mathbf{Y}) - c_s^*}{c_s^*} \leqslant p \tag{1}$$

or equivalently,

$$c_s(\mathbf{X},\mathbf{Y}) \leqslant (1+p)c_s^* \tag{2}$$

where $p \geqslant 0$ is a given constant, indicating the desired robustness level. (As noted above, the robustness level may alternately be scenario-dependent, given by p_s, but we assume $p_s = p$ for all s for simplicity.) The left-hand side of (1) is the relative regret for scenario s. The p-robust measure sets upper bounds on the maximum allowable relative regret for each scenario.

2.3. Formulation

We propose the following mixed-integer programming model for the p-LNDP.

$$(p-\mathrm{LNDP})\quad \text{minimize} \sum_{j\in\mathcal{V}_0} f_jX_j + \sum_{(i,j)\in\mathcal{A}} q_{ij}Y_{ij0} \tag{3}$$

$$\text{subject to} \sum_{j\in\mathcal{V}_0} f_jX_j + \sum_{(i,j)\in\mathcal{A}} q_{ij}Y_{ijs} \leqslant (1+p)c_s^* \quad \forall s\in S\setminus\{0\} \tag{4}$$

$$\sum_{(j,i)\in\mathcal{A}} Y_{jis} \leqslant b_j \quad \forall j\in\mathcal{V}_S,\ s\in S \tag{5}$$

$$\sum_{(j,i)\in\mathcal{A}} Y_{jis} - \sum_{(i,j)\in\mathcal{A}} Y_{ijs} = 0 \quad \forall j\in\mathcal{V}_T,\ s\in S \tag{6}$$

$$-\sum_{(i,j)\in\mathcal{A}} Y_{ijs} = b_j \quad \forall j\in\mathcal{V}_D,\ s\in S \tag{7}$$

$$\sum_{(j,i)\in\mathcal{A}} Y_{jis} \leqslant (1-a_{js})k_jX_j \quad \forall j\in\mathcal{V}_0,\ s\in S \tag{8}$$

$$X_j\in\{0,1\} \quad \forall j\in\mathcal{V}_0 \tag{9}$$

$$Y_{ijs} \geqslant 0 \quad \forall (i,j)\in\mathcal{A},\ s\in S \tag{10}$$

The objective function (3) minimizes the nominal cost, including fixed location costs and transportation costs. Constraints (4) enforce the p-robust criterion, requiring that the scenario cost may not be more than $100(1+p)\%$ of the optimal scenario costs c_s^*. If $p=\infty$, the p-robustness constraints become inactive and the formulation is equivalent to a deterministic logistics network design problem. Constraints (5)–(7) are the flow conservation constraints: for supply nodes, we require the flow out to be less than or equal to the supply; for demand nodes, we require the flow in to equal the demand; and for transshipment nodes, we require the flow in to equal the flow out. Constraints (8) ensure that the total flow through a node does not exceed its capacity when it is opened and fully functional in that scenario, and prevent any flow when it is closed or disrupted. Constraints (9) and (10) are standard integrality and nonnegativity constraints.

The optimal scenario costs c_s^* are calculated by solving the LNDP below for each of the scenarios s.

$$(\mathrm{P}_s)\quad c_s^* = \text{minimize} \sum_{j\in\mathcal{V}_0} f_jX_j + \sum_{(i,j)\in\mathcal{A}} q_{ij}Y_{ijs} \tag{11}$$

$$\text{subject to} \sum_{(j,i)\in\mathcal{A}} Y_{jis} \leqslant b_j \quad \forall j\in\mathcal{V}_S \tag{12}$$

$$\sum_{(j,i)\in\mathcal{A}} Y_{jis} - \sum_{(i,j)\in\mathcal{A}} Y_{ijs} = 0 \quad \forall j\in\mathcal{V}_T \tag{13}$$

$$-\sum_{(i,j)\in\mathcal{A}} Y_{ijs} = b_j \quad \forall j\in\mathcal{V}_D \tag{14}$$

$$\sum_{(j,i)\in\mathcal{A}} Y_{jis} \leqslant (1-a_{js})k_jX_j \quad \forall j \tag{15}$$

$$X_j\in\{0,1\} \quad \forall j \tag{16}$$

$$Y_{ijs} \geqslant 0 \quad \forall i,j \tag{17}$$

The objective function in (p-LNDP) indicates that decision makers are more interested in obtaining a solution that performs well under normal conditions, while the p-robustness constraints suggest that they are willing to make additional investments in infrastructure to protect against future disruptions to the supply chain facilities. As we will show later, the additional price they pay depends on their desired robustness level, i.e., to what level they want to be protected.

An alternate objective function, which accounts for more than the nominal scenario, is given by

$$\text{minimize} \quad \sum_{s\in SI} w_s \left[\sum_{j\in\mathcal{V}_0} f_jX_j + \sum_{(i,j)\in\mathcal{A}} q_{ij}Y_{ijs}\right] \tag{18}$$

where SI is the set of scenarios of interest and w_s is the weight of scenario s and represents its importance in decision making. If $SI=S$ and w_s is the probability that scenario s occurs, then (18) is the expected cost. For simplicity, we assume that only the nominal scenario is considered, i.e. $w_0=1$, $w_s=0\,\forall s\in S\backslash\{0\}$, in the following discussion, though our model and heuristic apply equally well to alternate objectives such as (18).

2.4. Complexity

Proposition 1 below says that the p-LNDP is NP-hard, since it can be reformulated as an extension of the capacitated fixed-charge network design problem(CFNDP), which is itself NP-hard.

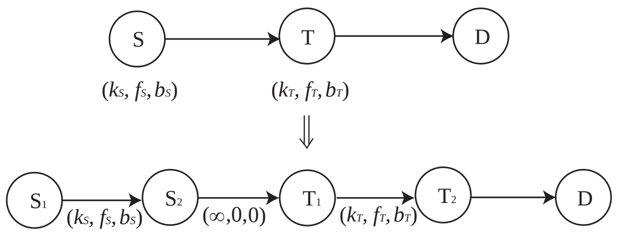

Fig. 4. Auxiliary graph.

Proposition 1. *The p-LNDP is NP-Hard.*

Proof. Let $|S| = 1$ and $p = \infty$, the p-LNDP reduces to the deterministic LNDP. We construct an auxiliary graph by as follows. For any one of the supply and transshipment nodes, we replace it with an arc connecting two dummy nodes, where the arc has the same fixed cost and capacity, but the flow cost is 0. Then we make open/close decisions on the arcs. Fig. 4 gives an example consisting of the original graph and its auxiliary graph; the symbols in the parentheses are capacity, fixed-cost and supply, correspondingly. Then the problem is equivalent to the classical capacitated fixed-charge network design problem (CFNDP). It contains as a special case the uncapacitated fixed charge network design problem, which is NP-hard. □

Moreover, not only is the p-LNDP NP-hard, but the problem of determining whether a given instance of the p-LNDP is feasible is itself NP-complete. Therefore, unlike most other robust optimization problems, it can be challenging just to find a feasible solution for the p-LNDP (and other p-robust problems). As p decreases, it becomes increasingly difficult to find feasible solutions. Our heuristic appears to be quite successful at finding feasible solutions when they exist. As we will see in Section 4, in our numerical study, our heuristic occasionally found feasible solutions for instances for which CPLEX could not, and the reverse was never true.

Proposition 2. *For $|S| \geqslant 2$ and $p \geqslant 0$, the feasibility problem for the p-LNDP is NP-complete.*

Proof. Consider an arbitrary instance of the stochastic p-robust UFLP (p-SUFLP) (Snyder and Daskin, 2006a). We will generate an instance of the p-LNDP that is feasible if and only if the p-SUFLP instance is feasible. The proposition then follows from the fact that the feasibility problem for the p-SUFLP is NP-complete (Snyder and Daskin, 2006a, p. 977). Please refer to Appendix A for detail proof of Proposition 2. □

Proposition 2 is proven for the case in which p is the same for all scenarios. Since this is a special case of the more general problem in which p_s is scenario specific, the feasibility problem is also NP-complete for the more general problem.

2.5. Identifying disruption scenarios

One of the remaining questions is how to identify the disruption scenarios. Assume that $|\mathcal{V}_0| = N$, implying a total of 2^N possible disruption scenarios, which leads to exponential growth in problem size and increasing computational burden to solve the problem. Moreover, when the number of scenarios is large, the value of p has to be chosen more carefully in order to maintain feasibility.

However, the number of scenarios to be considered in practice is much smaller. Decision makers may prefer to identify scenarios of concern using expert judgment. One option is to start by identifying scenarios in which exactly 1 facility is disrupted. The facilities in each of the scenarios are those more likely to be disrupted according to historical data or the decision maker's *a priori* beliefs on the importance of these facilities. The probability of more than two unrelated facilities to be disrupted at the same time is very small in reality. For the failure of multiple facilities simultaneously, additional scenarios can be identified by examining the potential threats to the supply chain, and how the facilities will be affected. For example, facilities that are located in the same area have a higher probability of becoming inactive together during disruption events such as earthquakes, electricity shortages, and so on. Similarly, facilities whose employees belong to the same labor union are more likely to be disrupted simultaneously by strikes. Since every supply chain is unique, every company will have a specific set of "what-if" contingencies that needs to be considered. This approach is demonstrated when generating test data set "US-10-15-30-65".

If the number of scenarios is very large, the scenario space may be approximated using sample average approximation (Kleywegt et al., 2001), in which the optimization problem is solved using a subset of the scenarios sampled using Monte Carlo simulation. The details of this approach are outside the scope of this paper.

3. Hybrid meta-heuristic algorithm

Algorithms based on Lagrangian relaxation and Benders decomposition have been proposed to solve p-robust network design problems (Kouvelis et al., 1992; Gutiérrez et al., 1996; Snyder and Daskin, 2006a). However, these methods can be computationally demanding, especially when the number of scenarios is large. (None of the three papers cited above report computational results on any problems with more than 14 scenarios.) Practitioners are often more interested in obtaining near-optimal solutions in relatively short time. Genetic algorithms (GA) are powerful global search heuristics inspired by evolution theory (Holland, 1975). GAs have gained popularity for their ease of implementation and successful application in a wide variety of optimization and search problems (Gen and Cheng, 2000).

We propose a hybrid metaheuristic algorithm which combines a genetic algorithm, local improvement search, and a shortest augmenting path method. A major improvement to the basic GA scheme is that for each generation, we apply a local search procedure called *learning* to improve the average population fitness. Numerical experiments (Section 4) show that the algorithm outperforms CPLEX in terms of CPU time and solution quality for most instances tested. Moreover, the algorithm can be easily adapted to solve extensions of our problem, such as additional constraints or alternate objective functions.

Our GA encodes values for the locations of the supply and transshipment nodes (the $\mathbf{X}$ variables). Once the $\mathbf{X}$ variables are chosen, the $\mathbf{Y}$ variables can be set optimally by solving $|S|$ capacitated min-cost network flow problems, one for each scenario, keeping the $\mathbf{X}$ variables fixed. This provides the objective function value for a given choice of $\mathbf{X}$ variables. The flow problems can be solved using the classical min-cost augmenting path algorithm (Ahuja et al., 1993).

3.1. Genetic algorithm-based heuristic

Unlike other search techniques that work on improving one single solution, GAs maintain a *population* of solutions and expend relatively little effort on each one. Each individual in the population is called a *chromosome* and represents a solution to the problem. A chromosome is often coded as a binary string of 0s and 1s. The population is updated through a process of successive iterations called *evolution*. A new population generated by one iteration is a new *generation*. In each generation, the fitness of every individual is evaluated, and multiple individuals are stochastically selected from the current population based on their fitness. The next generation is formed from these individuals by either merging two chromosomes using a *crossover* operation or modifying a chromosome using a *mutation* operation. The new population is then used in the next iteration of the algorithm. Typically, the algorithm terminates when either a maximum number of generations has been produced, a satisfactory fitness level has been reached for the population or the best solution converges, i.e. no obvious improvement can be observed for a certain number of generations. A more comprehensive overview on GAs and their applications can be found in Gen and Cheng (2000). GAs have been applied to various facility location and network design problems and have proven to be a very effective heuristic method to solve this type of problem, especially problems of large scale (see, e.g. Alp et al. (2003), Drezner and Wesolowsky (2003), Snyder and Daskin (2006)).

In this paper, we introduce a GA-based heuristic to solve the *p*-LNDP model. Different from the basic GA scheme, we apply a local improvement procedure called *learning*, which tends to find a better solution from the current chromosome before the crossover and mutation operation for each individual in each generation. Numerical tests show that this step greatly reduces the solution time. The details of our algorithm are described next.

3.1.1. Representation scheme

We use an n-digit binary string chromosome structure to represent a solution $\mathbf{X}$, where $n \equiv |\mathcal{V}_0|$. The ith digit on the string indicates whether the ith facility is open ("1") or not ("0"). For example, in Fig. 5, facilities 1, 2 and 4 out of a total of eight potential sites are chosen in the current solution. In order the speed up the local search procedure (introduced later), we apply a greedy method to sort the sequence of the binary digits in a chromosome based on two criteria:

(1) Cost savings of each facility. The idea here is to estimate a facility's value based on the cost savings that would result from adding one unit of capacity to the facility. We first assume that all candidate sites $j \in \mathcal{V}_0$ are opened and have capacity k_j, then solve the min-cost flow problem (MCFP) and obtain the optimal flow cost c_f^*. For each facility j, we then increase its capacity by *1* unit, keeping all other facilities at their original capacities, and solve the resulting MCFP to obtain the optimal flow cost $c_f^*(j)$. The cost savings for facility j is defined as $c_f^* - c_f^*(j)$, which represents how much we can save by adding one unit of capacity to the facility. We solve the MCFP $|\mathcal{V}_0| + 1$ times to calculate the cost savings for all candidate facilities. The higher a facility's cost savings, the higher the probability it will be opened in the optimal solution.

(2) The total number of times each facility is opened in the optimal scenario solutions. We solve the single-scenario problem for each $s \in S$. For each candidate facility, we count the number of scenarios in which it is opened in the optimal scenario solution. The more solutions in which a facility is opened, the more it is likely to be opened in the optimal full problem solution.

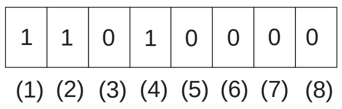

Fig. 5. Chromosome structure.

As a preprocessing step of our algorithm, we sort the facility indices in the binary string in non-increasing order first by (1) and then (to break ties) by (2). Therefore, in the local search procedure to be introduced below, we will first examine solutions in which we open the facilities that are more likely to be opened in the optimal solution.

3.1.2. Fitness function

For a specific solution **X**, we solve the MCFP for each scenario using the shortest augmenting path method to get the optimal flow assignments **Y**. From the solution (**X**,**Y**) we obtain the objective value $c_0(\mathbf{X},\mathbf{Y})$, as well as the scenario costs $c_s(\mathbf{X},\mathbf{Y})$. The objective function is a commonly used fitness function to justify the quality of a solution X. However, unlike most network design-type problems, a given location vector **X** is not guaranteed to produce a feasible solution, even after the scenario flows are found optimally, due to the p-robustness constraints. Therefore, we introduce a modified fitness function as follows:

$$f(X) = c_0(\mathbf{X},\mathbf{Y}) + \omega \cdot \sqrt{\sum_{s\in S}\left[\mathbf{1}_A\left(c_s(\mathbf{X},\mathbf{Y}) - (1+p)c_s^*\right)\right]^2} \tag{19}$$

where $\mathbf{1}_A(x)$ is an indicator function defined as

$$\mathbf{1}_A(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases} \tag{20}$$

and ω is a factor that the decision-maker can adjust to weight the penalty of violating the p-robust constraints. Based on the results of some preliminary experiments, we set $\omega = 50$ in our numerical tests. A smaller fitness value $f(X)$ indicates better fitness. If a solution **X** is feasible, the second part of the equation is 0 and the fitness value is exactly the objective value; if the solution violates some of the p-robust constraints, its fitness will be penalized by an amount based on the constraint violations.

3.2. Initial population generation

The experimental work by Alander (1992) suggests that a value between n and $2n$ is optimal for the population size (*pop_size*), where n is the length of a chromosome ($|\mathcal{V}_0|$, in our case). In our computational study, we fix the population size at 100, which is within $[n,2n]$ for most of the test instances.

The initial population is generated by first selecting the optimal solutions of each scenario by solving the scenario subproblems (P_s). If there are more than *pop_size* scenarios, we choose the *pop_size* best solutions. If the number of scenarios is smaller than *pop_size*, the rest of the individuals are generated by crossover between randomly drawn pairs from the optimal scenario solutions.

3.3. GA operators

In each generation, we perform the *crossover* and *mutation* operations after the learning process. Crossover starts by choosing two individuals randomly from the current population. For each pair, two new individuals are generated by switching some randomly chosen digits at the same position on the chromosomes (See Fig. 6). For each pair, a probability is drawn uniformly from the set $\{0.0, 0.1, 0.2, \ldots, 1.0\}$, and each digit is swapped with that probability.

Mutation serves as a mechanism to prevent the algorithm from becoming trapped in local optima and to ensure a diverse population. In a given iteration, each individual has a probability of 20% of being chosen for mutation. If selected, one digit will be randomly selected and changed from 1 to 0 or 0 to 1, as shown in Fig. 7.

3.4. The learning process

Before the crossover and mutation operation, we perform a local improvement procedure, which we call *learning*, for all individuals during each iteration. The philosophy behind this process is that we want to "train" the individuals to be "stronger" and "smarter", thus improving the average population fitness and increasing the probability of producing higher-quality

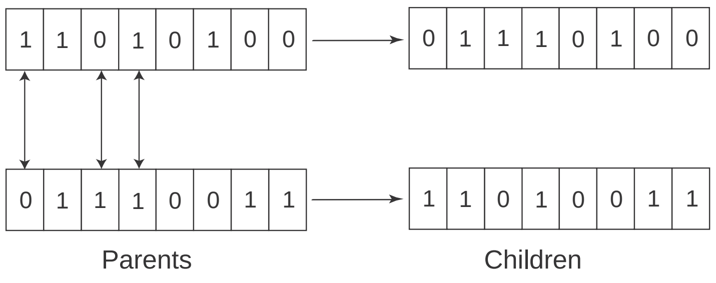

Fig. 6. Crossover operation.

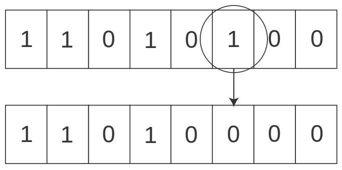

Fig. 7. Mutation operation.

offspring. Local improvement heuristics are well known to add a great deal of power to GAs by greatly shortening the time needed for convergence, and therefore improving the solution speed. Various strategies of local improvement have been incorporated into the basic GA scheme. Liaw (2000) proposes a hybrid genetic algorithm with local improvement procedure based on tabu search (TS) for the open shop scheduling problem. The local improvement is applied to each newly generated offspring before inserting it into the population. Snyder and Daskin (2006) present a random-key genetic algorithm for the Generalized Traveling Salesman Problem. The heuristic combines a genetic algorithm (GA) with a local tour improvement heuristic for each new individual in the population. See Jog et al. (1989), Miller et al. (1993), Torabia et al. (2006) for more examples. Our learning process, which consists of a tree search (described next), is performed for each individual in every generation.

3.4.1. Constructing the search tree

To undertake the learning process for a given individual, we construct a *search tree* whose root node consists of that individual. The child nodes of a given node in the tree are obtained by setting each 1 in the parent node's chromosome to 0, one by one. Therefore, the number of children of a given node equals the number of 1s in the chromosome. For example, the children of a node 1110101 are {1110100, 1110001, 1100101, 1010101, 0110101}. The branching process continues until all leaves of the tree have only a single 1 in their chromosomes. An example of a complete search tree with 11100 as the root node is illustrated in Fig. 8.

After the search tree has been constructed, we traverse the tree in the hope of finding a better solution than the root node. Note that there may be duplicate branches if a search tree is constructed in this way; to avoid wasted effort, we check for duplicates and skip the repeated branches during the search.

3.4.2. Local improvement

We apply a *depth-first search* (DFS) algorithm to traverse the search tree. The search starts from the root node and explores as far as possible along each branch before backtracking. To accelerate the search procedure, we prune the tree before reaching the leaves using the following heuristic. Let the current node be X with fitness $f(X)$. For each child X' of X, we compare its fitness $f(X')$ with $f(X)$. We explore the branch X' if $f(X') < \delta f(X)$, where $\delta \leqslant 1$, and we prune the branch X' otherwise. For every node X except the end nodes, we initially set $\delta = 1$ and dynamically update it throughout the algorithm, setting it to $f(X')/f(X)$ each time we find a new node satisfying $f(X') < \delta f(X)$.

We terminate the learning process when the tree has been exhaustively searched (including pruning), or when a pre-specified time limit has been reached. Upon termination, we replace the root-node solution with the best solution found during the search. (We used 5 s as the time limit for each individual. We tried longer time limits, e.g., 20 s, but there was no obvious improvement in the algorithm's performance.)

Of course, it is possible that our pruning process will prune good descendants of X, which may even contain the optimal solution. However, we use the crossover and mutation operators to create new individuals as a compromise. Our numerical tests suggest that this strategy is effective, i.e. in most of the test instances, we are able to find the optimal or near-optimal (gap <5%) solutions.

3.5. Termination criteria

Generally speaking, GAs stop when the best fitness of the population converges, or when the algorithm reaches a pre-specified time limit or iteration limit. In our numerical tests, we observed that the objective function value converges

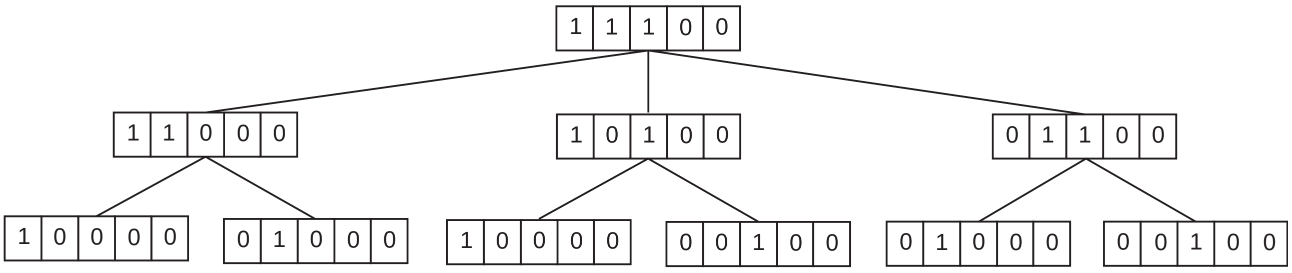

Fig. 8. Search tree.

within 20 generations most of the time. Therefore, we terminate the algorithm after 30 iterations, or when the improvement in the best solution is less than 10^{-5} for 10 consecutive iterations. Furthermore, we compare the best solution with the lower bound obtained from the LP relaxation of (p-LNDP), calculated using CPLEX in a pre-processing step, and stop when the gap is less than 5%.

Note that we do not require the best solution to be kept in the population, to avoid being trapped in local optima, but we always record the best solution found during the algorithm's execution. At termination, we compare the best solution in current population with the best solution kept in record, and return the better one.

3.6. The algorithm

The heuristic may be summarized algorithmically as follows:

```
Step 0 Set $t \leftarrow 0, k \leftarrow 0$
Step 1 Generate the initial population; let $X_t \leftarrow$ the best solution of generation $t$; let $X^* \leftarrow$ the best solution in record (i.e., $X^* \leftarrow X_0$)
Step 2 If $f(X_t) < f(X^*)$, let $X^* \leftarrow X_t$; in either case, apply the learning process for each individual
Step 3 Apply the crossover and mutation operators; compute $\Delta = (f(X_t) - f(X_{t-1}))/f(X_{t-1})$
    • If $\Delta \leqslant 10^{-5}$, let $k \leftarrow k + 1$
    • else, let $k \leftarrow 0$
Step 4 If $t \geqslant 30$ or $k \geqslant 10$, go to Step 5; else, let $t \leftarrow t + 1$, go to Step 2
Step 5 If $f(X_t) < f(X^*)$, let $X^* \leftarrow X_t$; return the best solution found $X^*$.
```

4. Computational results

We performed a series of numerical experiments to evaluate the performance of our algorithm. We coded the algorithm in C++ and executed it on a computer with an Intel Xeon E5430 2.66GHz processor and 2 GB of RAM, operating under Microsoft Windows XP Professional SP2. We benchmark our results using the branch-and-bound algorithm in CPLEX 11.0, which we ran on the same hardware. Computation times are reported in seconds.

4.1. Experimental design

We generated 49 random data sets of different problem sizes with the number of facilities ranging from 60 to 150, the numbers of scenarios ranging from 10 to 40, and the edge density choosing from 20%, 30% and 50%. We first chose the number of supply, transshipment and demand nodes, then constructed arcs between nodes based on the probability specified by the edge density. The arcs between the "emergency" facility and the demand nodes are constructed with probability 1. The resulting instances are labeled "$d - |\mathcal{V}_S| - |\mathcal{V}_T| - |\mathcal{V}_D| - |S|$," where d is the edge density. (For example, instance "20%-10-20-30-10" has 10 supply nodes, 20 transshipment nodes, 30 demand nodes, 10 scenarios, and an edge density of 20%.)

The fixed costs f_j and unit transportation costs q_{ij} are drawn uniformly from [5000, 15,000] and [1, 500], respectively. The unmet-demand penalty (i.e., the per-unit transportation cost from the emergency facility to each demand node) is 1500. At each demand node, the parameter b_j (representing the negative of the demand) is drawn uniformly from $[-110, -50]$. At each supply node, the supply b_j is determined as follows. To ensure the feasibility of the model under most of the data sets we generated, we first calculate the average required supply $\bar{s}$ by

$$\bar{s} = \frac{|\mathcal{V}_D|}{|\mathcal{V}_S|} \cdot \bar{d} \tag{21}$$

where $\bar{d}$ is the average demand, which is 80 in our case. Each b_j is then drawn uniformly from $[1.5\bar{s}, 2.5\bar{s}]$.

The capacity k_j of a supply node is the same as its supply b_j, while the capacity of each transshipment node is determined similarly by first calculating

$$\bar{c} = \frac{\sum_{j \in \mathcal{V}_D} d_j}{|\mathcal{V}_T|}, \tag{22}$$

then drawing k_j uniformly from $[1.5\bar{c}, 2.5\bar{c}]$. The capacity of the emergency facility is always set to infinity.

We generate disruption scenarios randomly, where each facility may be disrupted with probability 10% (This probability is intentionally chosen to be high in order to demonstrate the impact of disruptions and the performance of the model when disruptions are a significant factor.) If this process generates duplicate scenarios, then the duplicates are removed and the procedure is repeated until we obtain $|S|$ unique scenarios. The existence of a feasible solution depends heavily on the value of p. A large p, for example, $p = 1$, will usually guarantee feasibility of the problems. However, it will be less realistic since it will allow large increase in cost in disrupted scenarios. Small values of p will lead to infeasibility. For further discussion of this issue, including a heuristic to detect infeasibility, see Snyder and Daskin (2006a). A method to find the smallest feasible p

for a set of data is introduced in Section 4.4. We chose to run all of the tests with the same p for consistency. Therefore, we set $p = 0.15$ for all instances.

One of the data sets consists of geographic data of the United States, in which we choose 10 major industrial cities as the supply nodes, 15 major distribution hub cities as the transshipment nodes and the 30 largest cities, measured by population, as the demand nodes. The demands are proportional to the cities' population and scaled down by dividing by 10,000. The transportation costs are the travel distances between cities. The scenarios are defined by first including all 25 scenarios in which only one node is disrupted; then including the 20 scenarios in which all cities in the same state are disrupted, for all 20 states in the data set; and then generating 20 random scenarios. Other parameters are determined as described above.

We set a time limit of 1200 s and an optimality gap of 5% to terminate the algorithm.

4.2. Algorithm performance

We compare the performance of our algorithm with that of CPLEX in Table 2. The "Test Problem" column gives the instance name. For each algorithm, three columns report the run time ("Time"), objective value ("Cost") and optimality

Table 2
Algorithm performance – vs. CPLEX.

Test problem	CPLEX			Heuristic			% DIFF	
	Time	Cost	Gap (%)	Time	Cost	Gap (%)	Time (%)	Cost (%)
20%-10-20-30-10	9.7	1,089,309	2.13	3.6	1,083,225	1.56	37.15	99.44
20%-10-20-30-20	13.3	1,083,397	1.58	8.4	1,100,347	3.17	63.30	101.56
20%-10-20-30-30	23.4	1,083,225	1.20	4.9	1,098,120	3.30	20.98	101.38
20%-10-20-40-10	7.0	1,301,360	0.82	6.0	1,297,906	0.55	86.08	99.73%
20%-10-20-40-20	11.5	1,308,196	1.12	8.0	1,297,906	0.33	69.57	99.21
20%-10-20-40-30	18.3	1,307,370	0.72	10.9	1,307,370	1.48	59.53	100.00
20%-20-30-50-10	15.4	1,445,399	7.36	14.8	1,356,548	0.76	95.98	93.85
20%-20-30-50-20	39.8	1,352,886	0.48	11.5	1,356,836	7.54	28.89	100.29
20%-20-30-50-30	251.1	1,357,762	0.84	7.9	1,356,836	0.77	3.15	99.93
20%-20-30-50-40	90.6	1,363,529	0.42	11.5	1,367,487	0.72	12.70	100.29
20%-20-40-60-10	31.6	1,265,404	0.82	15.8	1,265,404	0.87	50.06	100.00
20%-20-40-60-20	57.4	1,286,958	2.48	13.8	1,270,751	5.35	24.03	98.74
20%-20-40-60-30	125.8	1,290,052	1.88	29.4	1,277,615	5.00	23.37	99.04
20%-20-40-60-40	396.5	1,306,235	3.08	20.8	1,302,072	2.75	5.25	99.68
20%-40-50-60-10	487.5	1,061,264	1.54	397.3	1,069,501	2.33	81.50	100.78
20%-40-50-60-20	1200	n/a	n/a	301.3	1,106,233	5.21	25.11	n/a
20%-40-50-60-30	1200	n/a	n/a	1200	n/a	n/a	n/a	n/a
20%-40-50-60-40	1200	n/a	n/a	451.1	1,109,213	4.49	37.59	n/a
30%-10-20-30-10	6.1	775,589	2.23	5.5	772,216	1.78	90.76	99.57
30%-10-20-30-20	13.2	786,124	3.62	6.3	772,216	1.79	47.62	98.23
30%-10-20-30-30	25.7	777,349	2.32	7.8	774,351	1.93	30.35	99.61
30%-10-20-40-10	10.3	837,035	1.85	7.5	836,723	1.81	72.53	99.96
30%-10-20-40-20	16.2	837,585	1.36	7.8	836,723	1.26	48.27	99.90
30%-10-20-40-30	35.5	848,359	2.63	6.8	836,723	5.60	19.17	98.63
30%-20-30-50-10	66.0	937,162	1.97	8.8	929,912	8.82	13.33	99.23
30%-20-30-50-20	176.7	947,762	2.13	85.1	942,491	1.56	48.16	99.44
30%-20-30-50-30	145.9	952,897	3.11	17.7	951,886	3.00	12.13	99.89
30%-20-30-50-40	1200	n/a	n/a	26.8	973,515	3.32	2.23	n/a
30%-20-40-60-10	85.7	946,954	0.47	13.6	944,588	8.43	15.87	99.75
30%-20-40-60-20	124.2	980,612	2.26	20.8	985,606	2.78	16.74	100.51
30%-20-40-60-30	478.5	983,037	2.05	33.4	996,175	3.41	6.98	101.34
30%-20-40-60-40	1200	n/a	n/a	37.6	1,035,180	7.28	3.13	n/a
30%-40-50-60-10	227.7	908,885	6.79	108.1	904,201	6.23	47.48	99.48
30%-40-50-60-20	1200	n/a	n/a	1200	n/a	n/a	n/a	n/a
30%-40-50-60-30	1200	n/a	n/a	1200	n/a	n/a	n/a	n/a
50%-10-20-30-10	18.8	540,019	2.78	2.3	528,375	0.57	12.23	97.84
50%-10-20-30-20	22.4	548,073	2.47	6.4	541,350	2.17	28.52	98.77
50%-10-20-30-30	68.6	547,223	1.73	7.5	545,787	5.43	10.94	99.74
50%-10-20-40-10	19.1	666,770	2.44	8.9	665,422	2.25	46.65	99.80
50%-10-20-40-20	31.1	696,778	6.98	20.9	667,780	2.53	67.29	95.84
50%-10-20-40-30	66.6	672,728	3.09	7.5	670,613	2.77	11.26	99.69
50%-20-30-50-10	106.6	700,737	2.51	43.1	697,477	2.03	40.43	99.53
50%-20-30-50-20	467.7	697,477	2.25	99.3	697,477	2.25	21.23	100.00
50%-20-30-50-30	652.4	706,586	3.15	123.3	711,734	3.90	18.90	100.73
50%-20-30-50-40	612.6	706,654	3.34	502.5	708,560	3.62	82.03	100.27
50%-20-40-60-10	205.4	694,156	1.73	188.0	696,787	2.12	91.55	100.38
50%-20-40-60-20	1200	n/a	n/a	148.9	706,080	3.14	12.41	n/a
50%-20-40-60-30	1200	n/a	n/a	n/a	n/a	n/a	n/a	n/a
US-10-15-30-65	660.0	2,601,379	0.37	29.2	2,601,379	0.37	4.42	100.00

gap vs. the CPLEX lower bound ("Gap"). The CPU times for both algorithms include the time required to compute the optimal scenario costs, c_s^*. The CPU times for the GA also include the time required to calculate the LP relaxation bound, which is used as part of the stopping criteria.

The two algorithms are compared in the ("% DIFF") column, where the ("Time") column gives the percentages of CPLEX's CPU time required by our algorithm and the ("Cost") gives the percent difference between the objective function values. A value less than 100% in the ("Cost") column indicates that our algorithm found a better solution, while a value less than 100% in the ("Time") column indicates that our algorithm was faster, which occurs in all instances.

As shown in Table 2, when compared with CPLEX, our algorithm was able to find the same or better solutions for 34 (or 69.4%) of the 49 data sets, while taking only a small fraction of CPLEX's time (39.5% on average). There are three instances for which CPLEX could not return a 5% solution in 1200 s, while our algorithm found solutions within that gap for every instance. For four instances, neither CPLEX nor our algorithm was able to find a 5% solution in 1200 s. Note that the computational time generally increases with the number of scenarios for the same network setting.

We also compare our heuristic to the basic GA scheme, i.e., without the local improvement process, for the randomly generated data sets. A selection of the results is given in Table 3. (Other instances are omitted due to space considerations, but the results are similar.) Our heuristic is able to find solutions at least as good as those from the basic GA while taking only a small fraction of the time.

We tested our heuristic on instances with smaller disruption probabilities (3% and 5%) but found little difference in performance. Therefore, detail results are omitted.

To test whether the algorithm's performance is significantly affected by the value of p, we also performed numerical experiments with different p values for several data sets. As shown in Fig. 9, the performance of our algorithm is not affected much by the value of p for most of the test data sets, except for 20%-40-50-60-20, for which a longer CPU time is required for smaller p. However, this problem is much harder with smaller p values, and, as we can see in Table 2, CPLEX cannot even find a feasible solution for $p = 0.15$ within 1200 s.

Table 3
Algorithm performance – vs. genetic algorithm.

Test problem	Basic GA			Heuristic			Time
	Time	Cost	Gap (%)	Time	Cost	Gap (%)	% DIFF
20%-10-20-30-10	8.0	1,083,225	0	1.2	1,083,225	0	15.00
20%-10-20-30-20	21.0	1,083,225	0	7.2	1,356,548	0	34.29
20%-10-20-40-30	26.13	1,307,370	0	3.6	1,307,370	0	13.78
20%-20-30-50-30	481.78	1,352,590	0.01	63.4	1,352,590	0.01	13.16
20%-20-30-50-40	637.34	1,083,721	3.57	64.2	1,082,597	3.45	29.93
30%-10-20-30-20	35.83	772,216	0	3.5	772,216	0	9.77
30%-10-20-40-20	66.75	836,723	0	1.5	836,723	0	2.25
30%-20-30-50-20	247.13	946,704	1.00	68.0	942,491	0.60	27.63
30%-20-40-60-10	57.33	944,588	0	6.8	944,588	0	11.86
30%-40-50-60-20	2237.59	931,241	5.00	413.9	927,263	4.70	18.50
50%-20-30-50-30	1671.16	660,894	1.00	18.0	660,894	1.00	1.08

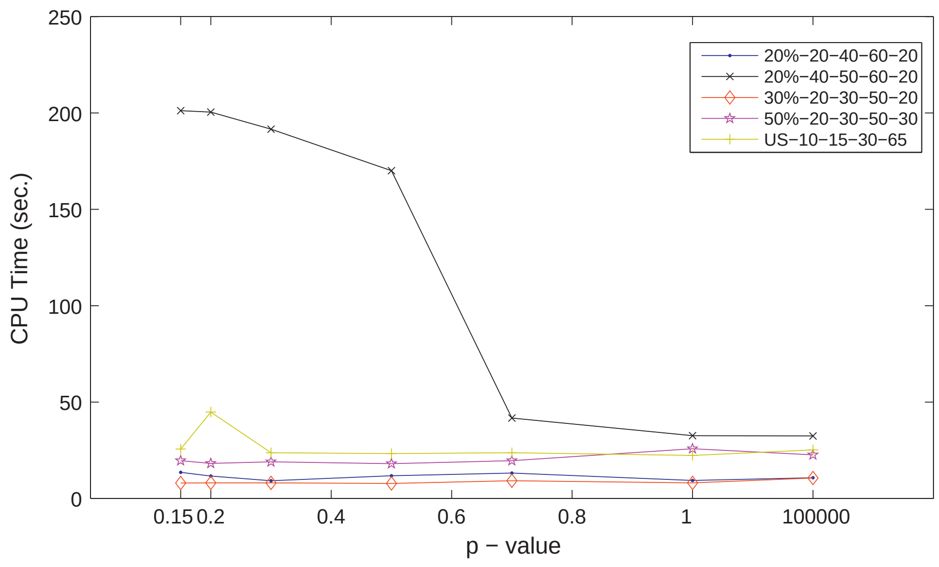

Fig. 9. Algorithm performance with different p values.

However, CPLEX's solution times are much more affected by different p values, as shown in Fig. 10.

4.3. Comparison with other robustness criteria

The two most widely studied robustness measures, minimax cost and minimax regret, have been questioned for being too conservative from a managerial stand-point, since they protect against only the worst-case scenario, which may only occur with a small probability. As a result, the relative regrets may be very large in other scenarios. In business, the philosophy of "prepare for the worst and wish for the best" may not be favored by managers whose objectives are to boost profit, or minimize cost. Our modeling approach attempts to take the perspective of a manager, for whom minimizing the company's nominal cost is a more suitable goal for the long term, while limiting the scenario costs greatly reduces the short-term risk when exposed to disruptions. This approach is less conservative compared with other robustness measures, as demonstrated in Table 4. The minimax cost and regret are formulated with the objective functions given below, with all constraints (4)–(10). These models are solved to optimality by CPLEX.

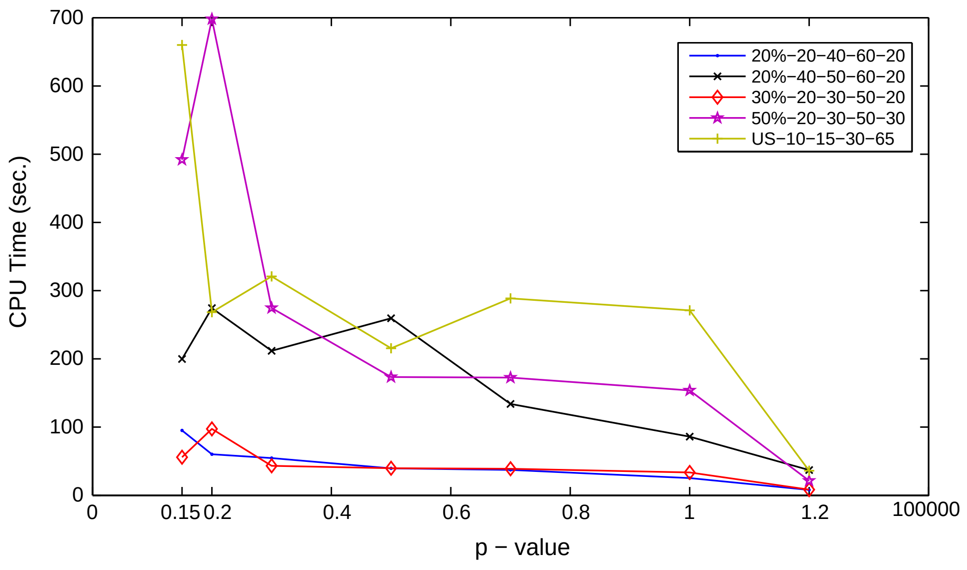

Fig. 10. CPLEX performance with different p values.

Table 4
Comparison with other robust criteria.

S	Opt	p-Robust		Minimax cost			Minimax regret		
		Cost	Regret	Cost	% DIFF	Regret	Cost	% DIFF	Regret
0	1,293,872	1,304,742	1.008	1,380,486	5.81	1.067	1,340,636	2.75	1.036
1	1,380,412	1,417,387	1.027	1,536,837	8.43	1.113	1,435,828	1.30	1.040
2	1,491,544	1,536,831	1.030	1,718,892	11.85	1.152	1,539,185	0.15	1.032
3	1,377,004	1,424,161	1.034	1,616,208	13.48	1.174	1,420,871	−0.23	1.032
4	1,355,321	1,395,719	1.030	1,471,926	5.46	1.086	1,408,501	0.92	1.039
5	1,468,679	1,520,669	1.035	1,681,322	10.56	1.145	1,558,724	2.50	1.061
6	1,493,352	1,574,351	1.054	1,530,685	−2.77	1.025	1,562,348	−0.76	1.046
7	1,490,743	1,526,260	1.024	1,575,124	3.20	1.057	1,555,332	1.90	1.043
8	1,360,442	1,368,789	1.006	1,477,890	7.97	1.086	1,401,844	2.41	1.030
9	1,326,277	1,338,619	1.009	1,424,519	6.42	1.074	1,374,447	2.68	1.036
10	1,326,277	1,338,619	1.009	1,424,519	6.42	1.074	1,374,447	2.68	1.036
11	1,293,872	1,304,742	1.008	1,381,811	5.91	1.068	1,340,636	2.75	1.036
12	1,314,535	1,342,785	1.021	1,471,419	9.58	1.119	1,381,522	2.88	1.051
13	1,714,311	1,843,482	1.075	1,763,248	−4.35	1.029	1,816,693	−1.45	1.060
14	1,412,676	1,488,826	1.054	1,511,722	1.54	1.070	1,502,212	0.90	1.063
15	1,375,834	1,394,089	1.013	1,502,552	7.78	1.092	1,430,055	2.58	1.039
16	1,357,671	1,414,395	1.042	1,477,127	4.44	1.088	1,441,102	1.89	1.061
17	1,314,910	1,336,351	1.016	1,492,141	11.66	1.135	1,373,204	2.76	1.044
18	1,293,872	1,304,742	1.008	1,380,486	5.81	1.067	1,340,636	2.75	1.036
19	1,421,817	1,513,518	1.064	1,713,979	13.24	1.205	1,476,344	−2.46	1.038
20	1,378,631	1,417,273	1.028	1,502,552	6.02	1.090	1,430,055	0.90	1.037

$$\text{(Minimax Cost)} \quad \text{minimize} \max_{s \in S} c_s(X,Y) \tag{23}$$

$$\text{(Minimax Regret)} \quad \text{minimize} \max_{s \in S} \frac{c_s(X,Y) - c_s^*}{c_s^*} \tag{24}$$

We have chosen a set of randomly generated data of size 20%-10-20-40-20. We have set $p = 0.15$. For each scenario, the table lists the optimal scenario costs c_s^* ("Opt" column). Then, for each model (p-LNDP, minimax cost, and minimax regret), the table lists the scenario cost ("Cost") and relative regret ("Regret") for each scenario. It also gives the percent differences ("% Diff") between the scenario cost of p-LNDP versus the minimax cost and minimax regret models. These results are depicted graphically in Figs. 11 and 12, which show the scenario costs and regrets, respectively. The table and figures demonstrate that the minimax cost and regret models protect against the worst-case scenario (scenario 13 in this case). Our model, in comparison, has a slightly higher cost in scenario 13 but has lower costs in all but four of the other scenarios. We can also see that the relative regrets can be very large in the minimax cost model ($\geqslant$15%), while they have been controlled in the p-robust and minimax regret models. Moreover, the p-robust model has smaller relative regrets than the minimax regret model in most of the test instances. Our approach focuses more on the nominal scenario and is less driven by the disruption scenarios than other risk measures. Numerical results do show that the p-robust model is less conservative than the other two models.

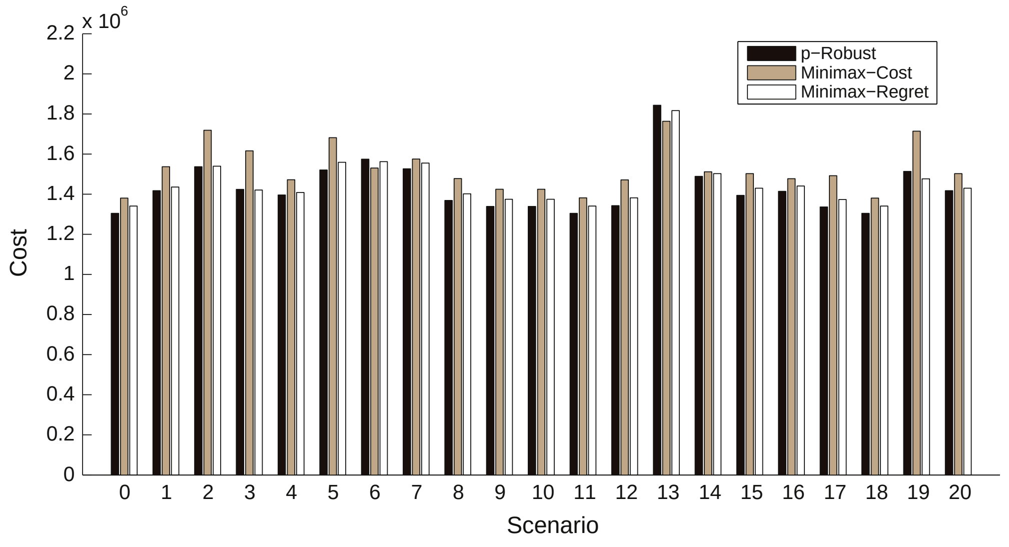

Fig. 11. Comparison with other robust criteria: cost.

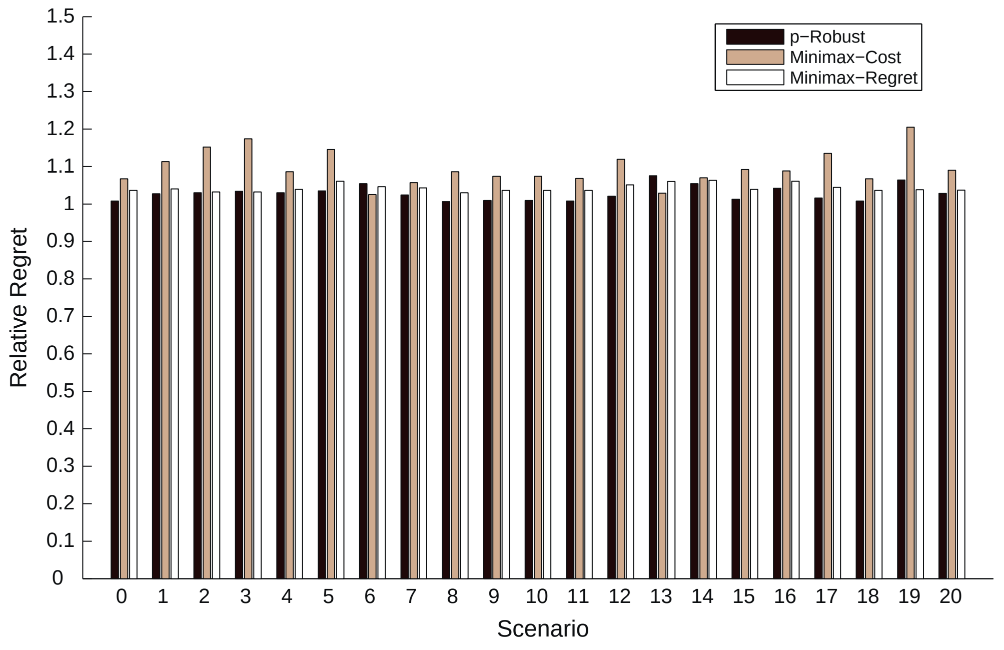

Fig. 12. Comparison with other robust criteria: relative regret.

Another advantage of the p-robust measure is that it provides decision makers more flexibility by allowing them to adjust p, while the minimax approaches only give one solution. We will next discuss this in detail.

4.4. Interpreting p-robustness

One of the main purposes of this research is to show the "price of robustness", in other words, how much it costs to design a more robust and reliable system that has much lower scenario costs when disruptions occur. As shown in Snyder and Daskin (2006a), "robustness" can often be bought with little increase in cost. We are able to observe similar phenomena in our problem. We generate a tradeoff curve between the maximal relative regret and the nominal cost as follows. We first set $p = \infty$ and solve the problem, and calculate the maximum relative regret over all scenarios. Note that when $p = \infty$, the p-robust constraints become inactive and the problem becomes a fixed-charge capacitated network design problem. Then we set p to the maximum relative regret minus 0.000001 and re-solve the problem, find the maximum relative regret again and reduce p again. At each iteration, the objective value increases and the maximum regret decreases because we have tightened the p-robust constraints. We continue this process until no feasible solution can be found. (Note that the smallest p value with a feasible solution also provides an upper bound for the optimal objective value of the minimax regret problem.)

Table 5
Nominal cost vs. maximum regret.

Problem	p	Cost	% Inc.	Max Reg.	% Dec.
*P*1	∞	1,081,873	0.00	0.6926	0.00
20-40-60-20-20%	0.6925	1,084,742	0.27	0.5972	13.75
	0.5972	1,087,107	0.22	0.5100	14.60
	0.5099	1,089,461	0.22	0.4325	15.17
	0.4325	1,089,760	0.03	0.4213	2.57
	0.4213	1,089,967	0.02	0.4197	0.35
	0.4197	1,091,741	0.16	0.4183	0.31
	0.4183	1,093,949	0.20	0.4145	0.88
	0.4145	1,095,008	0.10	0.4139	0.13
	0.4139	1,094,801	−0.02	0.4106	0.76
*P*2	∞	1,240,547	0.00	0.6856	0.00
30-40-50-20-20%	0.5497	1,241,184	0.04	0.5463	0.60
	0.5463	1,244,359	0.26	0.5344	2.16
	0.5344	1,244,441	0.01	0.5212	2.46
	0.5212	1,245,636	0.10	0.5122	1.70
	0.5122	1,245,737	0.01	0.3942	23.01
	0.3942	1,247,396	0.13	0.3908	0.84
	0.3908	1,249,465	0.17	0.3241	17.04
	0.3241	1,250,898	0.11	0.2961	8.61

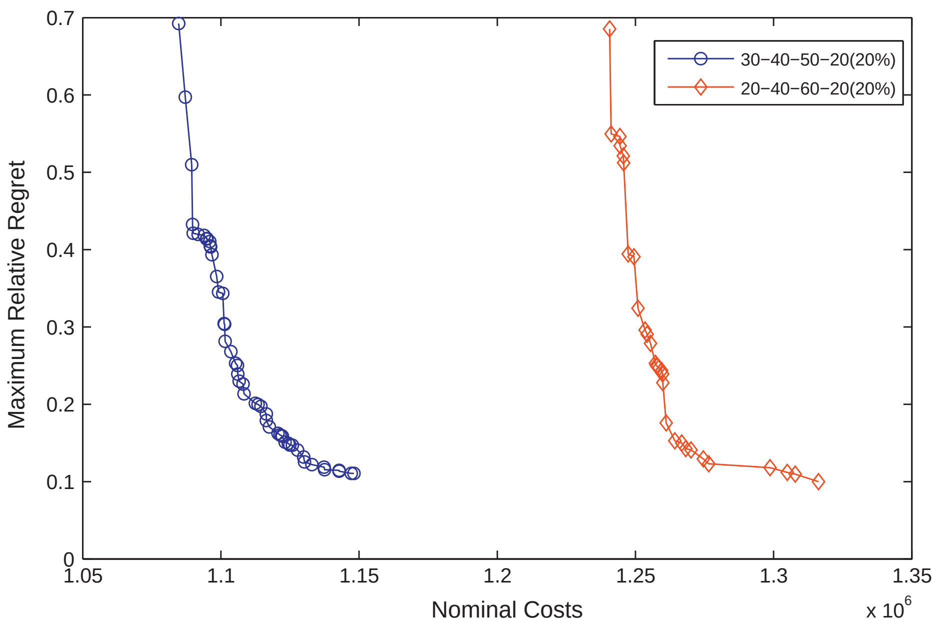

Fig. 13. Tradeoff between maximal relative regret and nominal cost.

This experiment was performed on two data sets, *P*1 (20-40-60-20-20%) and *P*2 (30-40-50-20-20%). The results are summarized in Table 5. The column "*p*" gives the value of *p*. The column "Cost" gives the objective value returned by our heuristic. "% Inc." is the percentage by which the objective value increased compared to the value obtained using the previous *p* value. The "Max Reg." column is the maximum relative regret over all scenarios, while "% Dec." is the percentage by which the maximum relative regret decreased compared to the previous value. We only include the first 10 solutions found because these will be of greatest interest to managers (since they have the lowest nominal costs). The tradeoff curves between objective value and maximum relative regret of both *P*1 and *P*2 are plotted in Fig. 13.

Both the table and the figure suggest that the left-most portions of the curves are quite steep, which means that the maximum relative regret can be greatly reduced with only a small increase in the nominal cost. For example, in *P*1, we are able to reduce the maximum relative regret from 0.6925 by 26.3% to 0.5099, with a slight increase in cost from 1,084,742 to 1,089,461, or 0.4%. In other words, with little increase in total operational cost in the nominal scenario, one can obtain robust supply chain networks that are more resilient to disruptions.

5. Conclusion

In this paper, we present a *p*-robust supply chain network design model that minimizes the nominal cost, while subject to the constraint that the solution must have a relative regret of no more than *p* in each scenario. We propose a metaheuristic to solve the model, and computational experiments show that we can obtain quality solutions that are very close to optimal given a fraction of the time required by CPLEX. This makes our heuristic attractive in situations in which numerous experiments must be carried out and obtaining high-quality, near-optimal solutions in a short period of time is desired. We also demonstrate that our approach can produce less conservative solutions than those obtained by the traditional robustness criteria minimax cost and minimax regret. From a managerial point of view, we have shown that robustness can be improved greatly without significant increases in investment, i.e., creating financial constraints to companies with careful up-front planning at the supply chain design phase. These results demonstrate the importance of increasing supply flexibility as a strategy to increase the reliability of a supply chain network. A tradeoff curve generated by choosing different *p* values can help management choose the desired robustness level based on budget constraints.

One possible extension is to consider multicommodity problems, in which several types of goods compete for limited capacity of the facilities. The scenario approach can also be adopted to model other sources of uncertainties, such as customer demands, transportation costs, and so on. It is also possible to consider network design together with other supply chain problems, such as inventory management, capacity expansion, vehicle routing, and so on, where traditional results are often in conflict with the objective of designing for disruptions.

Acknowledgments

This project was financed (in part) by NSF Grant #DMI-0522725 and by grants from the Commonwealth of Pennsylvania, Department of Community and Economic Development, through the Pennsylvania Infrastructure Technology Alliance (PITA). This support is gratefully acknowledged.

Appendix A. Proof of Proposition 2

For $|S| \geqslant 2$ *and* $p \geqslant 0$, *the feasibility problem for the p-LNDP is NP-complete.*

Proof. Consider an arbitrary instance of the stochastic *p*-robust UFLP (*p*-SUFLP) (Snyder and Daskin, 2006a). We will generate an instance of the *p*-LNDP that is feasible if and only if the *p*-SUFLP instance is feasible. The proposition then follows from the fact that the feasibility problem for the *p*-SUFLP is NP-complete (Snyder and Daskin, 2006a, p. 977).

An instance of the *p*-SUFLP is specified by a set I of customers, a set J of potential facility sites, a set S of scenarios (each of which specifies demands h_{is} $\forall i \in I$) and per-unit transportation costs d_{ijs} $\forall i \in I,\ j \in J$, fixed costs f_j for the facilities, a robustness parameter p, and optimal objective values z_s^* for the deterministic facility location problem induced by each scenario s. A feasible instance satisfies the following constraints:

$$\sum_{j\in J} Y_{ijs} = 1 \quad \forall i \in I,\ s \in S \tag{A.1}$$

$$Y_{ijs} \leqslant X_j \quad \forall i \in I,\ j \in J,\ s \in S \tag{A.2}$$

$$\sum_{j\in J} f_j X_j + \sum_{i\in I}\sum_{j\in J} h_{is} d_{ijs} Y_{ijs} \leqslant (1+p) z_s^* \quad \forall s \in S \tag{A.3}$$

$$X_j \in \{0,1\} \quad \forall j \in J \tag{A.4}$$

$$Y_{ijs} \in \{0,1\} \quad \forall i \in I,\ j \in J,\ s \in S \tag{A.5}$$

Define a new instance of the *p*-LNDP as follows. Let $\mathcal{V}_S = J$ and $\mathcal{V}_D = I$. Let S' be the scenario set, with $|S'| = |S|$. Let $\mathcal{V}_T$ contain $|I||J||S'|$ dummy transshipment nodes. Between each supply node $j \in \mathcal{V}_S$ and each demand node $i \in \mathcal{V}_D$, there are $|S'|$

transshipment nodes, denoted ijs. The network contains arcs from supply node $j \in \mathcal{V}_S$ to transshipment node ijs for all $i \in \mathcal{V}_D$ and $s \in S'$. It also contains arcs to demand node $i \in \mathcal{V}_D$ from transshipment node ijs for all $j \in \mathcal{V}_S$ and $s \in S'$. There are no other arcs in the network. (See Fig. A1.)

Let $b_i = 1$ for all $i \in \mathcal{V}_S$, $b_{ijs} = 0$ for all $ijs \in \mathcal{V}_T$, and $b_i = -1$ for all $i \in \mathcal{V}_D$. For each $j \in \mathcal{V}_S$, let f_j be the same as the f_j for the corresponding facility in the p-SUFLP instance. Let $f_{ijs} = 0$ for all $ijs \in \mathcal{V}_T$. Let p be equal to p in the p-SUFLP instance. Let $k_j = \infty$ for all $j \in \mathcal{V}_0$.

For each $ijs \in \mathcal{V}_T$ (lying between $j \in \mathcal{V}_S$ and $i \in \mathcal{V}_D$ and representing scenario s), let $q_{j,ijs} = 0$ and $q_{ijs,i} = h_{is}d_{ijs}$. For each $j \in \mathcal{V}_S$ and each scenario $s \in S'$, let $a_{ijs} = 0$ (supply nodes are never disrupted). Finally, for each $ijs \in \mathcal{V}_T$ and each scenario $t \in S'$, let $a_{ijs,t} = 0$ if $s = t$ and 1 otherwise. That is, the only non-disrupted transshipment nodes in scenario s are those whose outbound transportation costs correspond to scenario s in the p-SUFLP. The main idea behind this construction of our instance is that it "translates" a problem with stochastic demand and transportation costs (the p-SUFLP) into one with stochastic disruptions (the p-LNDP).

We first establish that the optimal scenario costs z_s^* in the p-SUFLP instance are equal to the optimal scenario costs c_s^* in the p-LNDP instance. Note that in scenario s of the p-LNDP, only one set of transshipment nodes is available for each (i,j) pair, and the cost of serving demand node i from source node j through that transshipment node is equal to the cost of serving customer i from facility j in scenario s of the p-SUFLP; in particular, both are equal to $h_{is}d_{ijs}$. Moreover, in both problems, facility j can only serve customer i if it is opened, at a fixed cost of f_j. Therefore, problem (P$_s$)—of opening source nodes and allocating them to demand nodes in the p-LNDP—is equivalent to the single-scenario problem of opening facilities and allocating them to customers in the p-SUFLP. Therefore $z_s^* = c_s^*$ for all $s \in S' = S$.

It remains to show that the original p-SUFLP instance is feasible if and only if the new p-LNDP instance is feasible.

($\Rightarrow$) First, suppose that the original p-SUFLP instance is feasible. Let $(\mathbf{X},\mathbf{Y})$ be a feasible solution. Define a solution $(\mathbf{X}',\mathbf{Y}')$ to the p-LNDP as follows. For each $j \in \mathcal{V}_S$, let $X'_j = X_j$. Let $X'_{ijs} = 1$ for every $ijs \in \mathcal{V}_T$. Let $Y'_{j,ijs,s} = Y'_{ijs,i,s} = Y_{ijs}$. (Note that the Y variables are binary while the Y' variables are non-negative, but since the demands in our p-LNDP instance are all equal to 1, they are equivalent.)

By constraints (A.3) in the p-SUFLP, for all $s \in S = S'$,

$$\sum_{j \in J} f_j X_j + \sum_{i \in I}\sum_{j \in J} h_{is}d_{ijs}Y_{ijs} \leqslant (1+p)z_s^* \iff \sum_{j \in \mathcal{V}_0} f_j X'_j + \sum_{(i,j)\in\mathcal{A}} q_{ij}Y'_{ijs} \leqslant (1+p)c_s^*,$$

where the equivalence follows from our construction of the p-LNDP instance and the fact that $z_s^* = c_s^*$. Therefore, constraints (4) of the p-LNDP hold. Constraints (5) and (7) hold at the supply and demand nodes (respectively) because the flow out of each supply node and into each demand node is 1. Constraints (6) hold at the transshipment nodes because $Y'_{j,ijs,s} = Y'_{ijs,i,s}$ by construction. Constraints (8) are satisfied at the supply nodes since supply nodes are never disrupted, they have infinite capacity, and outbound flows only exist if the facility was opened. Constraints (8) are satisfied at the transshipment nodes because flows are only routed through them when non-disrupted, in which case their capacity is infinite. Constraints (9) and (10) are implied by the equivalent constraints in the p-SUFLP. Therefore, our new solution $(\mathbf{X}',\mathbf{Y}')$ is feasible, and so is our instance of the p-LNDP.

($\Leftarrow$) Suppose now that the new p-LNDP instance is feasible. Note that, by the construction of our p-LNDP instance, if the instance is feasible, then there exists a feasible solution in which the flows $\mathbf{Y}'$ are binary. This is because the demands equal 1; there is exactly one feasible path from each supply node j to each demand node i in each scenario s; and if i's demand is split among multiple supply nodes in scenario s, then we can re-assign all of its demand to the supply node j with the smallest d_{ijs} (breaking ties arbitrarily) while maintaining the feasibility of all of the constraints. Therefore, let $(\mathbf{X}',\mathbf{Y}')$ be a feasible binary solution.

To define a solution $(\mathbf{X},\mathbf{Y})$ to the original p-SUFLP instance, for each $j \in J$, let $X_j = X'_j$ and let $Y_{ijs} = Y_{j,ijs,s}$.

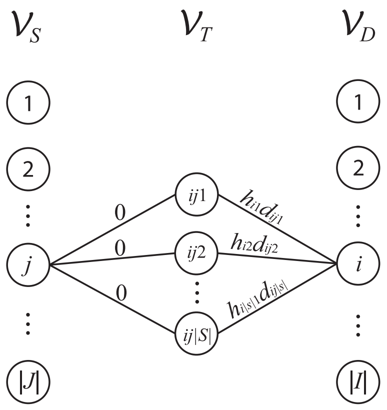

Fig. A1. Artificial instance of p-LNDP.

Constraints (A.1) in the p-SUFLP follow from the construction of our p-LNDP instance and constraints (7) in the p-LNDP. Constraints (A.2) follow from the construction of our instance and constraints (8) in the p-LNDP. By constraints (4) in the p-LNDP, for all $s \in S' = S$,

$$\sum_{j\in\mathcal{V}_0} f_j X'_j + \sum_{(i,j)\in\mathcal{A}} q_{ij} Y'_{ijs} \leqslant (1+p)c^*_s \iff \sum_{j\in J} f_j X_j + \sum_{i\in I}\sum_{j\in J} h_{is} d_{ijs} Y_{ijs} \leqslant (1+p)z^*_s,$$

where the equivalence again follows from our construction of the p-LNDP instance and the fact that $c^*_s = z^*_s$. Therefore, constraints (A.3) in the p-SUFLP hold. Constraints (A.4) and (A.5) hold trivially. Therefore, our new solution $(\mathbf{X}, \mathbf{Y})$ is feasible, and so is the original instance of the p-SUFLP. □

References

Ahuja, R., Magnanti, T., Orlin, J., 1993. Network Flows: Theory, Algorithms, and Applications. Prentice Hall, New York.
Alander, J., 1992. On optimal population size of genetic algorithms. In: CompEuro 1992 Proceedings, Computer Systems and Software Engineering, 6th Annual European Computer Conference. IEEE Computer Society Press, The Hague, Silverspring, MD, pp. 65–70.
Alp, O., Erkut, E., Drezner, Z., 2003. An efficient genetic algorithm for the p-median problem. Annals of Operations Research 122 (1–4), 21–42.
Barrionuevo, A., Deutsch, C., 2005. A distribution system brought to its knees. New York Times, September 1.
Bathgate, A., Hayashi, A., 2008. Airlines lining up for boeing 787 compensation. Reuters, April 10.
Berman, O., Krass, D., Menezes, M.B.C., 2007. Facility reliability issues in network p-median problems: strategic centralization and co-location effects. Operations Research 55 (2), 332–350.
Bunschuh, M., Klabjan, D., Thurston, D., 2006. Modeling robust and reliable supply chains. Working Paper, University of Illinois at Urbana-Champaign, Urbana, IL 61801, USA.
Clark, D., Takahashi, Y., 2011. Quake disrupts key supply chains. The Wall Street Journal Asia, March 12.
Cordeau, J., Pasin, F., Solomon, M., 2006. An integrated model for logistics network design. Annals of Operations Research 144 (1), 59–82.
Cui, T., Ouyang, Y., Shen, Z.-J.M., 2010. Reliable facility location design under the risk of disruptions. Operations Research 58 (4-Part-1), 998–1011.
Daskin, M., Snyder, L., Berger, R., 2005. Facility location in supply chain design. In: Langevin, A., Riopel, D. (Eds.), Logistics Systems: Design and Operation. Springer, New York, pp. 39–66, Chapter 2.
Drezner, Z., 1987. Heuristic solution methods for two location problems with unreliable facilities. The Journal of the Operational Research Society 38 (6), 509–514.
Drezner, Z., Wesolowsky, G., 2003. Network design: selection and design of links and facility location. Transportation Research Part A 37 (3), 241–256.
Gen, M., Cheng, R., 2000. Genetic Algorithms and Engineering Optimizations. Wiley, New York.
Gutiérrez, G., Kouvelis, P., Kurawarwala, A., 1996. A robustness approach to uncapacitated network design problems. European Journal of Operational Research 94 (2), 362–376.
Hendricks, K., Singhal, V., 2005. An empirical analysis of the effect of supply chain disruptions on long-run stock price performance and equity risk of the firm. Production and Operations Management 14 (1), 35–52.
Hicks, M., 2002. When the chain snaps. eWeek – Enterprise News & Reviews. <http://www.eweek.com/c/a/Channel/When-the-Chain-Snaps/>.
Holland, J., 1975. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. University of Michigan Press, Michigan:.
Jog, P., Suh, J., Gucht, D., 1989. The effects of population size, heuristic crossover and local improvement on a genetic algorithm for the travelling salesman problem. In: Proceedings of the Third International Conference on Genetic Algorithms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, pp. 110–115.
Kleindorfer, P., Saad, G., 2005. Managing disruption risks in supply chains. Production and Operations Management 14 (1), 53–68.
Kleywegt, A., Shapiro, A., Homem-De-Mello, T., 2001. The sample average approximation method for stochastic discrete optimization. SIAM Journal on Optimization 12 (2), 479–502.
Kouvelis, P., Kurawarwala, A., Gutiérrez, G., 1992. Algorithms for robust single and multiple period layout planning for manufacturing systems. European Journal of Operational Research 63 (2), 287–303.
Kouvelis, P., Yu, G., 1997. Robust Discrete Optimization and Its Applications. Kluwer Academic Publishers, Nowell, MA.
Latour, A., 2001. Trail by fire: A blaze in albuquerque sets off major crisis for cell-phone giants – nokia handles supply chain shock with aplomb as ericsson of sweden gets burned – was sisu the difference? Wall Street Journal, A1, January 29.
Li, X., Ouyang, Y., 2010. A continuum approximation approach to reliable facility location design under correlated probabilistic disruptions. Transportation Research Part B 44 (4), 535–548.
Liaw, C., 2000. A hybrid genetic algorithm for the open shop scheduling problem. European Journal of Operational Research 124 (1), 28–42.
Lim, M., Daskin, M.S., Bassamboo, A., Chopra, S., 2009. Facility location decisions in supply chain networks with random disruption and imperfect information. Working Paper, Department of Industrial Engineering and Management Sciences, Northwestern University, Evanston, IL.
Lim, M., Daskin, M.S., Bassamboo, A., Chopra, S., 2010. A facility reliability problem: formulation, properties and algorithm. Naval Research Logistics 57, 58–70.
Magnanti, T., Wong, R., 1984. Network design and transportation planning: models and algorithms. Transportation Science 18 (1), 1–55.
Meixell, M., Gargeya, V., 2005. Global supply chain design: a literature review and critique. Transportation Research Part E 41 (6), 531–550.
Miller, J., Potter, W., Gandham, R., Lapena, C., 1993. An evaluation of local improvement operators for genetic algorithms. IEEE Transactions on Systems, Man, and Cybernetics 23 (5), 1340–1351.
Mo, Y., Harrison, T., 2003. A conceptual framework for robust supply chain design under demand uncertainty. In: Geunes, J., Pardalos, P. (Eds.), Supply Chain Optimization. Springer, pp. 43–263.
Mouawad, J., 2005. Katrina's shock to the system. The New York Times, September 4.
Owen, S., Daskin, M., 1998. Strategic facility location: a review. European Journal of Operational Research 111 (3), 423–447.
Scapparra, M., Cappanera, P., 2010. Optimal allocation of protective resources in shortest-path networks. Transportation Science 45 (1), 64–80.
Shen, Z.-J.M., Zhan, R.L., Zhang, J., 2007. The reliable facility location problem: Formulations, heuristics, and approximation algorithms. Working Paper, Department of Industrial Engineering and Operations Research, University of California at Berkeley.
Snyder, L., 2003. Supply chain robustness and reliability: Models and algorithms. Ph.D. dissertation, Northwestern University, Department of Industrial Engineering and Management Sciences.
Snyder, L., 2006. Facility location under uncertainty: a review. IIE Transactions 38 (7), 537–554.
Snyder, L., Daskin, M., 2005. Reliability models for facility location: the expected failure cost case. Transportation Science 39 (3), 400–416.
Snyder, L., Daskin, M., 2006. A random-key genetic algorithm for the generalized traveling salesman problem. European Journal of Operational Research 174 (1), 38–53.
Snyder, L., Daskin, M., 2006a. Stochastic p-robust location problems. IIE Transactions 38 (11), 971–985.

Snyder, L., Scaparra, M., Daskin, M., Church, R., 2006. Planning for disruptions in supply chain networks. In: Greenberg, H. (Ed.), TutORials in operations research. INFORMS, Baltimore, pp. 234–257.

Soni, S., Pirkul, H., 2000. Design of survivable networks with connectivity requirements. Telecommunication Systems 20 (1–2), 133–149.

Tang, C., 2006. Robust strategies for mitigating supply chain disruptions. International Journal of Logistics Research and Applications 9 (1), 33–45.

Torabia, S., Ghomib, S.F., Karimib, B., 2006. A hybrid genetic algorithm for the finite horizon economic lot and delivery scheduling in supply chains. European Journal of Operational Research 173 (1), 173–189.

Zhan, R.L., Daskin, M.S., Shen, Z.-J.M., 2008. Facility reliability with site-specific failure probabilities. Working Paper, Department of Industrial Engineering and Operations Research, University of California-Berkeley.