# A Network Recourse Decomposition Method for Dynamic Networks with Random Arc Capacities

**Warren B. Powell**

Department of Civil Engineering and Operations Research, Princeton University, Princeton, New Jersey 08544

**Raymond K.-M. Cheung**

Department of Industrial and Manufacturing Systems Engineering, Iowa State University, Ames, Iowa 50011

We study a class of two-stage dynamic networks with random arc capacities where the decisions in the first stage must be made before realizing the random quantities in the second stage. The expected total cost of the second stage is a function of the first-stage decisions, known as the expected recourse function, which is generally intractable. This paper proposes a new method to approximate the expected recourse function as a convex, separable function of the supplies to the second stage. First, the second-stage network is decomposed using Lagrangian relaxation into tree subproblems whose expected recourse functions are numerically tractable. Subgradient optimization is then used to update the Lagrange multipliers to improve the approximations. Numerical experiments show that this structural decomposition approach can produce high-quality approximations of the expected recourse functions for large stochastic networks. 

In a variety of problems, such as production planning, logistics, and distribution planning and dynamic fleet management, decisions must be made over time prior to the realization of random demands. These problems can be modeled as dynamic networks with random arc capacities (DNRAC) where the arc capacities are used to represent the random demands. If the goal is to minimize both the total cost of current decisions and the expected total cost in the future time periods, then this class of problems can be formulated as a special class of stochastic programs with network recourse. Wallace [15] gave a specialized procedure to solve these problems when the number of random variables is small. However, we can rely only on approximation methods for large problems. This paper presents a new approximation method for this class of problems using a structural decomposition strategy.

An acyclic, dynamic network with random arc capacities with $t = 1, \ldots, N$ time stages can be formulated as an $N$-stage stochastic program with network recourse. In this paper, we are concerned only with two-stage problems. However, the methodology developed can be extended to multistage problems. To state the problem mathematically, we let $\xi(2)$ denote the random vector defined over the probability space $(\Omega, \mathcal{F}, P)$ with elementary outcomes $\omega \in \Omega$. Let

$\mathcal{N}(t)$ = the set of all points in space in stage $t$,
$\mathcal{N}_s(t)$ = the set of supply points in space in stage $t$,
$\mathcal{N}_t(t)$ = the set of pure transshipment points in space in stage $t$,
$\mathcal{N}_d(t)$ = the set of demand points in space in stage $t$,
$\tilde{\xi}(1)$ = deterministic upper bound from point $i$ to point $j$ in stage 1,

NETWORKS, Vol. 24 (1994) 369–384

CCC 0028-3045/94/070369-16

$x_{ij}(1)$ = flow from point $i$ to point $j$ in stage 1,
$\xi_{ij}(2)$ = random upper bound from point $i$ to point $j$ in stage 2,
$\xi_{ij}(2, \omega)$ = a realization of $\xi_{ij}(2)$,
$x_{ij}(2, \omega)$ = flow from point $i$ to point $j$ in stage 2 for the realization $\xi_{ij}(2, \omega)$,
$x_{ij}(t)$ = cost per unit flow from point $i$ to point $j$ in stage $t$, and
$R_i(t)$ = external supply to point $i$ in stage $t$.

We also define a *state* variable that can significantly simplify our presentation:

$S_i(1)$ = internal supply to point $i$ in stage 2 resulting from decisions in stage 1.

The vector $S(1) = [\cdots, S_i(1), \cdots]^T$ summarizes the state of the system after stage 1. It is used to communicate the decisions in stage 1 with the decisions in stage 2. Without loss of generality, we assume that all points in space are homogeneous over time, meaning that

$$\mathcal{N} = \mathcal{N}(1) = \mathcal{N}(2),$$

and similarly for the sets $\mathcal{N}_s$, $\mathcal{N}_t$, and $\mathcal{N}_d$. The two-stage stochastic programming formulation of DNRAC is given by

$$\min_{x(1)} c(1)^T x(1) + E_\omega Q(S(1), \xi(2, \omega)), \tag{1}$$

subject to

$$\sum_{j \in \mathcal{N}_t \cup \mathcal{N}_d} x_{ij}(1) = R_i(1) \quad \forall i \in \mathcal{N}_s \tag{2}$$

$$\sum_{i \in \mathcal{N}_s \cup \mathcal{N}_t} x_{ij}(1) - \sum_{k \in \mathcal{N}_t \cup \mathcal{N}_d} x_{jk}(1) = 0 \quad \forall j \in \mathcal{N}_t \tag{3}$$

$$\sum_{i \in \mathcal{N}_s \cup \mathcal{N}_t} x_{ij}(1) - S_j(1) = 0 \quad \forall j \in \mathcal{N}_d \tag{4}$$

$$0 \le x_{ij}(1) \le \tilde{\xi}_{ij}(1) \quad \forall i, j \in \mathcal{N}, \tag{5}$$

where for a given $S(1)$ and a realization $\xi(t, \omega)$, $Q(S(1), \xi(2, \omega))$ is a minimization problem:

$$Q(S(1), \xi(2, \omega)) = \min_{x(2,\omega)} c(2)^T x(2, \omega), \tag{6}$$

subject to

$$\sum_{j \in \mathcal{N}_t \cup \mathcal{N}_d} x_{ij}(2, \omega) = R_i(2) + S_i(1) \quad \forall i \in \mathcal{N}_s \tag{7}$$

$$\sum_{i \in \mathcal{N}_s \cup \mathcal{N}_t} x_{ij}(2, \omega) - \sum_{k \in \mathcal{N}_t \cup \mathcal{N}_d} x_{jk}(2, \omega) = 0 \quad \forall j \in \mathcal{N}_t \tag{8}$$

$$\sum_{i \in \mathcal{N}_s \cup \mathcal{N}_t} x_{ij}(2, \omega) \ge 0 \quad \forall j \in \mathcal{N}_d \tag{9}$$

$$0 \le x_{ij}(2, \omega) \le \tilde{\xi}_{ij}(2, \omega) \quad \forall i, j \in \mathcal{N}. \tag{10}$$

Problem (1)–(5) is called the *first-stage problem* and problem (6)–(10) is called the *network recourse problem.* Note that there is no specific amount that must be received at the demand points, which is reflected by constraints (9). In this study, we make the following assumptions:

1. $\xi_{ij}(2)$ are finite, independent random variables with some known discrete distributions.
2. $Q(\cdot, \cdot)$ are finite for all $x(1) \in \Re^n$ and $\omega \in \Omega$.
3. There is no arc connecting nodes in $\mathcal{N}_s$ and no arc connecting nodes in $\mathcal{N}_d$.

The second assumption is also known as *complete recourse* [19].

The study of the expectation functional [such as $E_\omega Q(S(1), \xi(2, \omega))$ in (1)] is the heart of stochastic programming. However, obtaining the exact expected recourse function is feasible only for some very special cases, such as the simple recourse model (see Ziemba [20] and Wets [18]). Therefore, the challenge in stochastic programming becomes developing methodologies to approximate this function, either implicitly or explicitly. In the following, we first state some general techniques, then describe our approach.

For a general stochastic program, we can use the information of scenarios drawn from the distribution. For example, the stochastic quasi-gradient method (see Ermoliev [3]) generates a sequence of linear approximations of the expected recourse function by using stochastic gradients. The L-shaped decomposition method (Van Slyke and Wets [14]) and its stochastic version, stochastic decomposition (Higle and Sen [6]), use cutting planes to establish lower bounds of the expected recourse function. Furthermore, these bounds can be refined when more samples are involved (see Kall et al. [7]). These methods involve solving large deterministic problems that are usually handled by decomposing the original problem into scenario subproblems. On the other hand, the progressive hedging method (see Rockafellar and Wets [12]) solves the deterministic equivalent problem via

augmented Lagrangian functions for a fixed set of scenarios. Its application to a financial model can be found in Mulvey and Vladimirou [8].

In another class of methods, the recourse problem is approximated by a simpler one such that a direct use of the distribution is possible. Hence, the resulting expected recourse can be found explicitly, allowing the use of classical nonlinear techniques in the first-stage problem. For example, if the recourse problem is approximated by a simple recourse model, $E_\omega Q(S(1), \xi(2, \omega))$ becomes a separable function that can be obtained by one-dimensional integrations (see, e.g., Qi [11]). Powell [9] and Frantzeskakis and Powell [5] introduced the concept of nodal recourse, which leads to a successive linearization approximation technique for dynamic networks with random arc capacities.

Dynamic networks in real applications often have random vectors with high dimensionality. It is not clear how accurately a small number of scenarios can approximate the true problem (the number of all realizations can be in the range of $10^{100}$). However, when the acyclic network structure is exploited, we may develop efficient approximation techniques. One attempt has been made in Wallace [16], where a piecewise linear upper bound is developed for the expected network recourse function. In this research, we wish to replace $E_\omega Q(S(1), \xi(2, \omega))$ with an approximation resulting in the following general form:

$$\min_{x(1), S(1)} \; c(1)^T x(1) + \hat{Q}(S(1))$$

$$\text{subject to} \quad (2)\text{–}(5), \quad (11)$$

where $\hat{Q}(S(1)) = \sum_i \hat{Q}_i(S_i(1))$ is a convex, piecewise linear and separable approximation of $E_\omega Q(S(1), \xi(2, \omega))$. The separability assumption greatly simplifies our calculation of $\hat{Q}(S(1))$ and allows the problem (11) to be solved as a pure network (separability has also been used to establish bounds on the expected recourse function, see Birge and Wets [1]). To see this, consider the dynamic network shown in Figure 1(a). Assume that we have found $\hat{Q}_i(S_i(1))$ for each supply node $i$ in stage 2 [see Fig. 1(b)]. As long as these functions are piecewise linear and convex, we can represent them by a set of deterministic links. We then augment the first-stage problem by adding these links to the terminal nodes, as shown in Figure 1(c). It is clear that the resulting problem is a pure network that can be solved efficiently. In this paper, we look at how we obtain the functions $\hat{Q}_i(S_i(1))$.

Recently, Powell and Cheung [10] studied a special class of tree-structured stochastic network problems where the expected recourse function is shown to be numerically tractable. In this paper, we leverage this result for a more general network problem. We present a new method to provide approximations of the expected recourse function based on a structural decomposition of the network recourse problem. In this method, we replace the network recourse function with stochastic programs involving trees that can be solved using the techniques in [10]. We also conduct some numerical experiments to show that the quality of the resulting approximation can be quite high. This approach provides a new perspective on using decomposition in solving stochastic programs, in contrast to the conventional decomposition scheme that produces scenario subproblems. In distinguishing the scenario-based decomposition methods, we refer to the class of methods that are based on the decomposition of the network structure as *network recourse decomposition* (NRD) methods. The method introduced in this paper is a network recourse decomposition method.

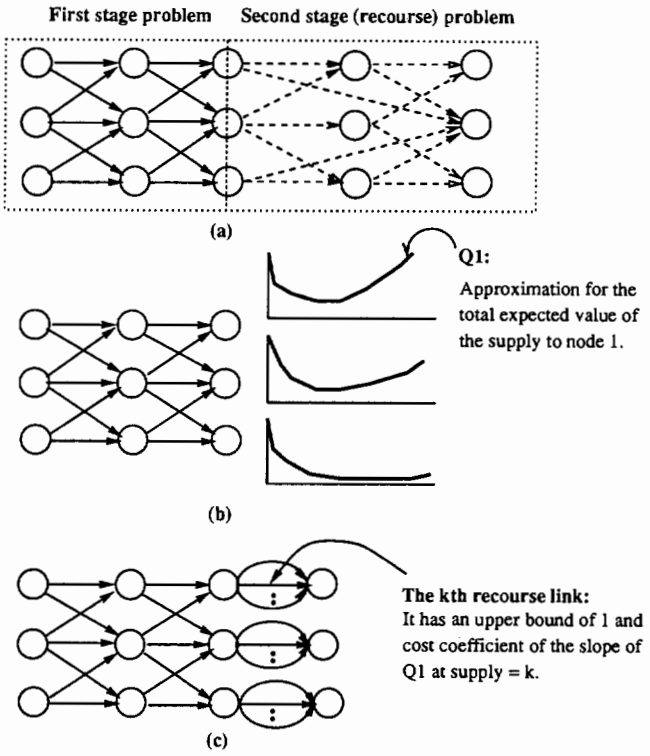



**Fig. 1.** The basic approach. (a) A two-stage dynamic network with random arc capacities. (b) Approximation of the expected total cost of the second stage problem by a piecewise linear, convex and separable function. (c) An augmented stage 1 problem obtained by adding a sets of deterministic links which represent the convex functions in (b).

This paper is organized as follows: In Section 1, we review the tree recourse problem that lays the foundation of our decomposition method. Section 2 introduces a structural decomposition scheme to formulate the network recourse problem as a multicommodity flow problem involving tree subproblems. We introduce in Section 3 a stochastic version of Lagrangian relaxation and a column generation-based method to iteratively update our

approximations. Section 4 summarizes the steps of the decomposition algorithm. Section 5 is devoted to numerical experiments. Concluding remarks are given in Section 6. In the remainder of this paper, we are concerned only with approximating the expected recourse function for the recourse problem. Therefore, we drop the time index unless otherwise specified.

## 1. TREE RECOURSE PROBLEM

Powell and Cheung [10] studied a class of stochastic programs where the recourse problem consists of a forest of trees with random arc capacities. The calculation of the exact expected recourse function is simplified by using the structure imposed by the trees. An efficient algorithm is developed to find the expected recourse function exactly for each individual tree. In this section, we first summarize the key ideas of this work. We then show how this method can be used to find the expected flow on each link, a result that we need later in the development of the decomposition algorithm. Finally, we work through an example to illustrate these results.

We begin by considering a tree recourse problem that consists of a directed tree $T = (\mathcal{N}, \mathcal{A})$ with deterministic arc costs and random arc capacities. Let $r$ be the only supply node of $T$ with supply $s$. There is no specific amount of flow that must be shipped from node $r$ to each ending node. For a fixed $s$ and a given realization of $\xi(\omega)$, the tree recourse problem is a deterministic minimization problem whose path-flow formulation is

$$Q(s, \xi(\omega)) = \min_{f_n(\omega)} \sum_{n=1}^{N_p} \tilde{c}_n f_n(\omega), \tag{12}$$

subject to

$$\sum_{n=1}^{N_p} f_n(\omega) = s \tag{13}$$

$$\sum_{n=1}^{N_p} f_n(\omega)\delta_{ij}^n \le \xi_{ij}(\omega) \quad \forall i, j \in \mathcal{N} \tag{14}$$

$$f_n(\omega) \ge 0, \tag{15}$$

where

$N_p$ = the number of paths in $T$,

$\delta_{ij}^n = \begin{cases} 1 & \text{if link } (i, j) \text{ is on path } n \\ 0 & \text{otherwise,} \end{cases}$

$\tilde{c}_n$ = the cost of path $n$, and

$f_n(\omega)$ = the flow on path $n$ for a realization $\xi(\omega)$.

An example of a two-level tree with random arc capacities is depicted in Figure 2(a), whereas a possible realization is shown in Figure 2(b). Suppose that $s = 1$. The optimal solution of this particular example is to put this unit of flow on path $1 \to 2 \to 4$ with a total cost of 13. However, this unit of flow may use different paths in different realizations, resulting in different total costs. Our aim is to find the expected total cost of this tree problem as a function of the scalar supply.

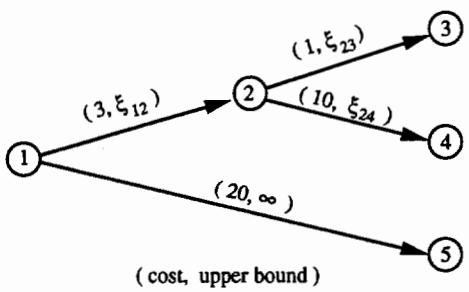
A major step in finding $E_\omega Q(s, \xi(\omega))$ is to compute the probability that a particular unit of flow entering the tree uses a particular path. Once this probability is established for each incremental unit of supply, the value of $E_\omega Q(s, \xi(\omega))$ immediately follows. To see this, let

$\phi(k, n) = P\{$the $k$th unit of flow entering node $r$ takes the $n$th path$\}$,

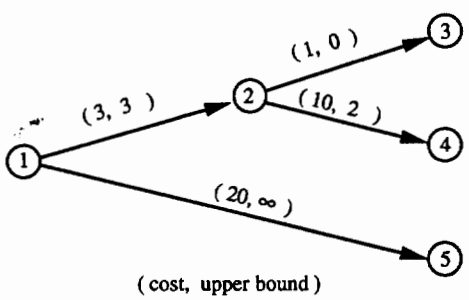
$G(k)$ = expected marginal cost for the $k$th unit of flow entering node $r$.

Then, the expected recourse function is obtained by

$$E_\omega Q(s, \xi(\omega)) = \sum_{k=1}^{s} G(k) = \sum_{k=1}^{s} \sum_{n=1}^{N_p} \tilde{c}_n \cdot \phi(k, n). \tag{16}$$




(a)




(b)

**Fig. 2.** An example of trees with random arc capacities. Figure (b) shows a possible realization of the tree in (a).

Furthermore, Powell and Cheung [10] showed that assigning flow to the least-cost available paths using a greedy algorithm is an optimal strategy for the tree recourse problem, both in the deterministic and stochastic cases. This result leads to an efficient algorithm for computing the routing probabilities $\phi(k, n)$ when all paths are ranked according to their costs. Numerical experiments in [10] show that for trees with more than 2000 random arc capacities that take an average of about four different values the exact expected recourse functions can be found parametrically as a function of $s$ for $s = 0, 1, \ldots, 200$ in a few seconds.

We know that the $\phi(k, n)$ is not only sufficient to establish the expected recourse function, but is also sufficient to obtain the expected link flows for the tree recourse problem. We state this result in Proposition 1.1. This property is crucial when we refine our approximations of the expected recourse function, as described in Section 3.

**Proposition 1.1.** *The expected optimal flow of problem* (12)–(15) *can be obtained by*

$$E_\omega x_{ij}(\omega) = \sum_{k=1}^{s} \sum_{n=1}^{N_p} \delta_{ij}^n \phi(k, n) \quad \forall (i, j) \in \mathcal{A}, \quad (17)$$

*where s is the supply.*

*Proof.* Define an indicator variable by

$$I_{ij}^k(\omega) = \begin{cases} 1 & \text{if the } k\text{th unit of flow uses link } (i, j) \\ 0 & \text{otherwise.} \end{cases}$$

Then, the flow on link $(i, j)$ for a realization $\omega$ is

$$x_{ij}(\omega) = \sum_{k=1}^{s} I_{ij}^k(\omega).$$

The result follows since

$$E_\omega I_{ij}^k(\omega) = \sum_{n=1}^{N_p} \delta_{ij}^n \phi(k, n). \qquad \blacksquare$$

**Example.** Consider the tree in Figure 2(a) again. Assume that the random arc capacities are all 0-1 random variables with probabilities: $P\{\xi_{12} = 1\} = 0.9$, $P\{\xi_{23} = 1\} = 0.5$, and $P\{\xi_{24} = 1\} = 0.6$. If we rank the paths according to their cost, then $p_1 = (1 \to 2 \to 3)$ with cost $\tilde{c}_1 = 4$, $p_2 = (1 \to 2 \to 4)$ with cost $\tilde{c}_2 = 13$, and $p_3 = (1 \to 5)$ with cost $\tilde{c}_3 = 20$. Let $Z_k$ be the total capacity of the first $k$ paths. Consider the first unit of supply to this tree. It will use $p_1$ only if its capacity is at least one, i.e.:

$$\begin{aligned} \phi(1, 1) &= P\{Z_1 \geq 1\} \\ &= P\{\min\{\xi_{12}, \xi_{23}\} \geq 1\} \\ &= P\{\xi_{12} \geq 1\} P\{\xi_{23} \geq 1\} \\ &= 0.45. \end{aligned}$$

On the other hand, this unit will use $p_2$ only if $Z_1 = 0$ and $Z_2 \geq 1$, meaning that

$$\begin{aligned} \phi(1, 2) &= P\{Z_1 = 0 \cap Z_2 \geq 1\} \\ &= P\{\xi_{12} = 1, \xi_{23} = 0, \xi_{24} = 1\} \\ &= 0.27, \end{aligned}$$

and, hence, $\phi(1, 3) = 0.28$. Similarly, we can obtain the routing probabilities for the second unit of supply: $\phi(2, 1) = 0$, $\phi(2, 2) = 0.27$, and $\phi(2, 3) = 0.73$. Note that $Z_2$ is at most 2, so $\phi(k, 3) = 1 \forall k \geq 3$. Using Eq. (16), the expected recourse function is given by

$$E_\omega Q(s, \xi(\omega)) = \begin{cases} 10.91 & \text{if} \quad s = 1 \\ 29.02 & \text{if} \quad s = 2 \\ 29.02 + 20k & \text{if} \quad s \geq 3. \end{cases}$$

Assume that $s = 2$; then by Equation (17), the expected flows on the capacitated links are

$$E_\omega x_{12}(\omega) = 0.99, \quad E_\omega x_{23}(\omega) = 0.45, \quad \text{and} \quad E_\omega x_{24}(\omega) = 0.44.$$

## 2. STRUCTURAL DECOMPOSITION

In general, network recourse problems do not exhibit treelike structures. However, dynamic trees with random arc capacities can be important subproblems when using a particular structural decomposition. In Section 2.1, we formulate the network recourse problem as a multicommodity problem involving trees. Then, Section 2.2 shows that our ability to solve tree subproblems can lead to a lower bound on the original expected recourse function.

### 2.1. Multicommodity Formulation

In this section, we describe a two-level relaxation scheme to decompose a network recourse problem to a number of trees with random arc capacities. In the first level, we decompose the network recourse problem by origins, producing a number of single-source network subproblems. In the second level, we create a tree representation for each single-source network using the notion of the

*extended graph* introduced in Powell and Cheung [10]. These relaxations are described as follows:

Let us recast the network recourse problem again [assume that $R(2) = 0$ for simplicity]:

$$Q(S, \xi(\omega)) = \min_{x(\omega)} c^T x(\omega), \tag{18}$$

subject to

$$\sum_{j \in \mathcal{N}_t \cup \mathcal{N}_d} x_{ij}(\omega) = S_i \qquad \forall i \in \mathcal{N}_s \tag{19}$$

$$\sum_{i \in \mathcal{N}_s \cup \mathcal{N}_t} x_{ij}(\omega) - \sum_{l \in \mathcal{N}_t \cup \mathcal{N}_d} x_{jl}(\omega) = 0 \qquad \forall j \in \mathcal{N}_t \tag{20}$$

$$0 \le x_{ij} \le \xi_{ij}(\omega) \quad \forall i, j \in \mathcal{N}. \tag{21}$$

Suppose that we define a *commodity* to be a unit of flow from a particular origin and let

$K$ = the number of origins,
$r(k)$ = origin node for commodity $k$, and
$x_{ij}^k(\omega)$ = flow of commodity $k$ from point $i$ to point $j$ for a given realization.

Then, we have

$$x_{ij}(\omega) = \sum_{k=1}^{K} x_{ij}^k(\omega),$$

and the multicommodity flow formulation of problem (18)–(21) is

$$Q(S, \xi(\omega)) = \min_{x^k(\omega)} \sum_{k=1}^{K} c^T x^k(\omega), \tag{22}$$

subject to

$$\sum_{j \in \mathcal{N}_t \cup \mathcal{N}_d} x_{r(k)j}^k(\omega) = S_{r(k)} \qquad \forall k = 1, \ldots, K, \tag{23}$$

$$\sum_{i \in \mathcal{N}_s \cup \mathcal{N}_t} x_{ij}^k(\omega) - \sum_{l \in \mathcal{N}_t \cup \mathcal{N}_d} x_{jl}^k(\omega) = 0 \qquad \forall k = 1, \ldots, K, \quad \forall j \in \mathcal{N}_t \tag{24}$$

$$0 \le x_{ij}^k \le \xi_{ij}(\omega) \qquad \forall k = 1, \ldots, K, \quad \forall i, j \in \mathcal{N} \tag{25}$$

$$\sum_{k=1}^{K} x_{ij}^k \le \xi_{ij}(\omega) \qquad \forall i, j \in \mathcal{N}. \tag{26}$$

In this formulation, if constraints (26) are relaxed, then this problem becomes separable by commodity, with the looser constraint (25). We refer to it as the first level of relaxation. Note that the constraints (23)–(25) indicate that each commodity follows a single-source network. We now see how to reformulate a single-source network as a tree problem, again using relaxation.

Given an acyclic, single-source network $G$, let $M = (\mathcal{N}_M, \mathcal{A}_M) \subseteq G$ be the network formed by a collection of paths $\mathcal{P}_M = \{p_1, p_2, \ldots, p_m\}$ in $G$. Note that different paths may share an arc that may produce a cycle in $M$. If this type of "shared" arc is duplicated to eliminate path dependencies, then the resulting graph is a tree. We call it the *extended graph* with respect to $\mathcal{P}_M$, denoted by $T_M$. The second level of relaxation is to obtain the extended graph with respect to the paths of each commodity network. This type of relaxation has been employed in other stochastic network problems (see, e.g., Wallace [17]) to establish bounds on the corresponding objective values.

The two-level relaxation is illustrated in Figure 3. Consider the two-source network in Figure 3(a). In the first level of relaxation, we treat the flows entering this network from different origins as different commodities. Figure 3(b) shows the two commodity subnetworks associated with the two origins. These subnetworks induce coupling on links (3, 5), (3, 4), and (4, 7). In the second level of relaxation, we apply the extended graph concept to each commodity subnetwork. Hence, node 5 in the first subnetwork and link (4, 7) in the second one are duplicated. Then, the resulting graph of the original network consists of two trees as shown in Figure 3(c).

We are now ready to write the network recourse problem as a multicommodity flow problem, where each commodity follows a particular tree. Let

$\mathcal{P}_k$ = a collection of paths used by commodity $k$,
$\mathcal{T}_k$ = the extended graph with respect to $\mathcal{P}_k$,
$B_{ij}$ = set of links in $\cup_k \mathcal{T}_k$, corresponding to the link $(i, j) \in \mathcal{A}$,
$\xi_{i'j'}(\omega) = \xi_{ij}(\omega)$ if $(i', j') \in B_{ij}$,
$y_{i'j'}^k(\omega)$ = the flow on link $(i', j') \in \mathcal{T}_k$ for realization $\xi_{i'j'}(\omega)$,

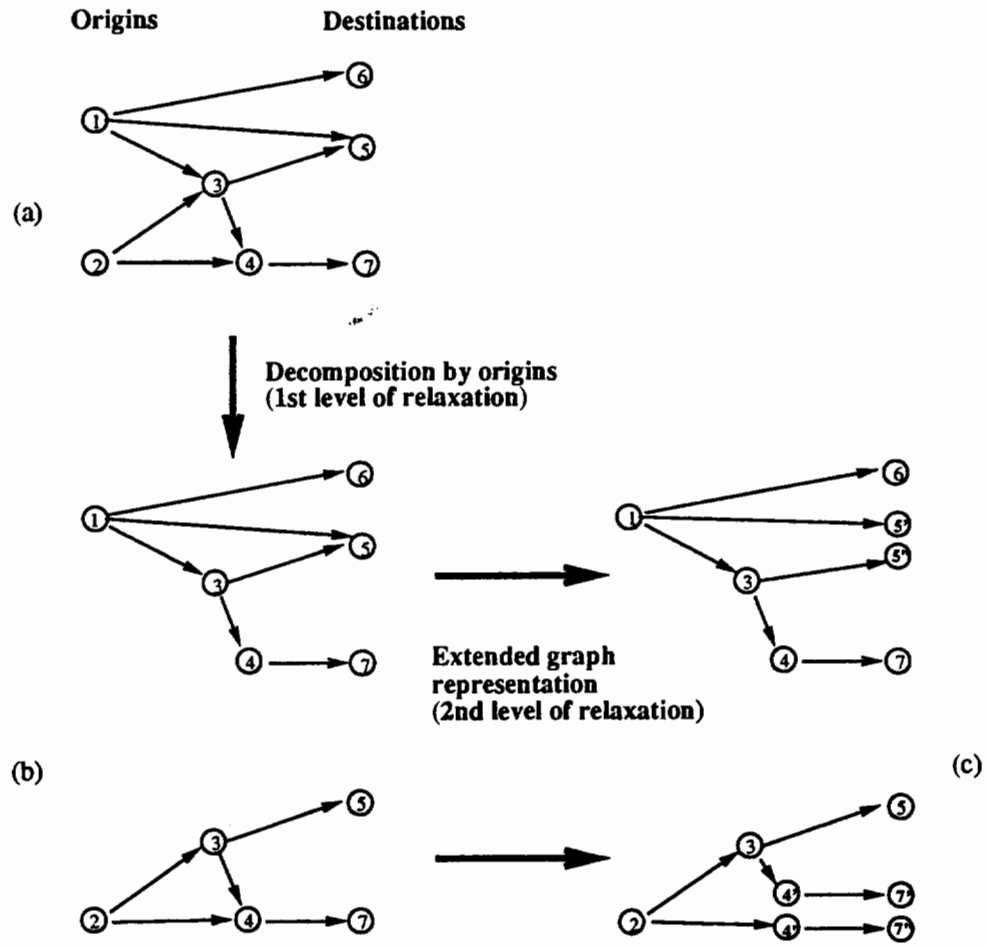



**Fig. 3.** The two-level decomposition of a two-origin transshipment into two tree subproblems.

$c'_{i'j'} = c_{ij}$ if $(i', j') \in B_{ij}$,
$\mathcal{N}^k$ = set of nodes in $\mathcal{T}_k$,
$\mathcal{N}^k_s$ = set of supply nodes in $\mathcal{T}_k$,
$\mathcal{N}^k_t$ = set of pure transshipment nodes in $\mathcal{T}_k$, and
$\mathcal{N}^k_d$ = set of demand nodes in $\mathcal{T}_k$.

Assuming that $\mathcal{P}_k$ includes all possible paths for each commodity $k$, problem (22)–(26) can be rewritten as

$$Q(S, \xi(\omega)) = \min_{y(\omega)} \sum_{k=1}^{K} \sum_{(i,j)\in\mathcal{A}} \sum_{(i',j')\in B_{ij}} c'_{i'j'} y^k_{i'j'}(\omega), \quad (27)$$

subject to

$$\sum_{j'\in\mathcal{N}^k_t\cup\mathcal{N}^k_d} y^k_{r(k)j}(\omega) = S_{r(k)} \qquad \forall k = 1, \ldots, K \quad (28)$$

$$\sum_{i'\in\mathcal{N}^k_s\cup\mathcal{N}^k_t} y^k_{i'j'}(\omega) - \sum_{l'\in\mathcal{N}^k_t\cup\mathcal{N}^k_d} y^k_{j'l'}(\omega) = 0 \qquad \forall k = 1, \ldots, K, \forall j' \in \mathcal{N}^k_t \quad (29)$$

$$0 \le y^k_{i'j'}(\omega) \le \xi_{i'j'}(\omega) \qquad \forall k = 1, \ldots, K, \forall i', j' \in \mathcal{N}^k \quad (30)$$

$$\sum_{k=1}^{K} \sum_{(i'j')\in B_{ij}} y^k_{i'j'}(\omega) \le \xi_{ij}(\omega) \qquad \forall (i, j) \in \mathcal{A}. \quad (31)$$

In this formulation, constraints (28)–(30) represent the feasible set of the tree flows where each tree is rooted at an origin [tree $k$ is rooted at node $r(k)$]. Then, the network recourse problem is separable by tree flows up

to the bundle constraints (31). Therefore, if constraints (31) are relaxed, this problem consists of a set of trees with random arc capacities. In the next section, we describe a method to handle the bundle constraints in a stochastic context.

## 2.2. Relaxation

We now show how a network recourse problem can be decomposed into tree subproblems by relaxing the stochastic bundle constraints. To simplify our notation, let

$$\hat{y}_{ij}^k(\omega) = \sum_{(i'j')\in B_{ij}} y_{i'j'}^k(\omega),$$

$$T_k(\omega) = \{y_{i'j'}^k(\omega) \text{ such that (28)–(30) is satisfied for commodity } k\}.$$

Note that $T_k(\omega)$ is the feasible set for commodity $k$ following the tree $\mathcal{T}_k$ for a given realization.

We start by considering the *Lagrangian relaxation* $L(S, \lambda(\omega))$ of problem (27)–(31) for a given realization $\xi(\omega)$:

$$L(S, \lambda(\omega)) = \min_{y^k(\omega)\in T_k(\omega)} \sum_k c'^T \hat{y}^k(\omega) + \lambda(\omega)^T\Big(\sum_k \hat{y}^k(\omega) - \xi(\omega)\Big), \quad (32)$$

where $\lambda(\omega)$ is the vector of Lagrange multipliers associated with the bundle constraints (31). Assume that we can find $\lambda^*(\omega)$ such that

$$\lambda^*(\omega) = \arg\max_{\lambda(\omega)\geq 0} L(S, \lambda(\omega)). \quad (33)$$

From duality, problem (27)–(31) is equivalent to problem (32) with $\lambda(\omega) = \lambda^*(\omega)$, meaning that

$$Q(S, \xi(\omega)) = L(S, \lambda^*(\omega)). \quad (34)$$

By taking expectations of both sides, we get

$$E_\omega Q(S, \xi(\omega)) = E_\omega\Big\{\min_{y^k(\omega)\in T_k(\omega)} \sum_k (c' + \lambda^*(\omega))^T \hat{y}^k(\omega)\Big\} - E_\omega\{\lambda^*(\omega)^T \xi(\omega)\}. \quad (35)$$

Notice that the Lagrange multipliers $\lambda^*(\omega)$ defined in (33) are optimal with respect to the realization $\xi(\omega)$ only. Therefore, they may not be unique among different realizations. If we replace them by a vector $\lambda \geq 0$ and let

$$\bar{\xi} = E_\omega \xi(\omega),$$

then we have

$$E_\omega Q(S, \xi(\omega)) \geq E_\omega\Big\{\min_{y^k(\omega)\in T_k(\omega)} \sum_k (c' + \lambda)^T \hat{y}^k(\omega)\Big\} - \lambda^T \bar{\xi} \quad (36)$$

$$= \sum_k E_\omega\Big\{\min_{y(\omega)\in T_k(\omega)} (c' + \lambda)^T \hat{y}^k(\omega)\Big\} - \lambda^T \bar{\xi}. \quad (37)$$

Inequality (36) follows the definition of $\lambda^*(\omega)$ in (33) and Equation (37) is by separability in commodities. Hence, we obtain an approximation (which is actually a lower bound) of the expected recourse function $E_\omega Q(S, \xi(\omega))$, denoted by

$$\hat{Q}(S, \lambda) = \sum_k \hat{Q}_k(S_k, \lambda) - \lambda^T \bar{\xi}, \quad (38)$$

where

$$\hat{Q}_k(S_k, \lambda) = E_\omega\Big\{\min_{y^k(\omega)\in T_k(\omega)} (c + \lambda)^T \hat{y}^k(\omega)\Big\}. \quad (39)$$

The functions $\hat{Q}_k(S_k, \lambda)$ represent the solutions to tree recourse problems. For a given $\lambda$, the minimization problem embedded in the expectation of (39) is a tree with random arc capacities. Consequently, we can use the techniques of Section 1 to calculate $\hat{Q}_k(S_k, \lambda)$ parametrically as a function of $S_k$ for each commodity (tree) $k$.

## 3. UPDATING PROCEDURE

There are two issues in calculating the function $\hat{Q}_k(S_k, \lambda)$ defined in (39), namely, the determination of $\lambda$ and the generation of trees. For any given $\lambda$, the function $\hat{Q}(S, \lambda)$ is a lower bound of the expected recourse function. Therefore, we could maximize this function with respect to $\lambda$ to get a tighter bound. On the other hand, the collection of paths $\mathcal{P}_k$ for each commodity $k$ can be very large due to the total enumeration of paths. Therefore, we wish to restrict the flow to a relatively small number of paths. In the following, we first show how to obtain $\lambda$

in Section 3.1. Next, we describe a method to dynamically generate trees in Section 3.2.

### 3.1. Updating the Multipliers

The multipliers $\lambda$ could be iteratively updated to provide better approximations to the expected recourse function $E_\omega Q(S, \xi(\omega))$. Recall from (36) that

$$\hat{Q}(S, \lambda) \le E_\omega\{Q(S, \omega)\}. \tag{40}$$

We would like to find $\lambda^*$ such that

$$\lambda^* = \arg\max_{\lambda \ge 0} \hat{Q}(S, \lambda), \tag{41}$$

since $\hat{Q}(S, \lambda^*)$ would be the tightest lower bound to $E_\omega Q(S, \omega)$ among all $\lambda$. Let $\partial_\lambda \hat{Q}(S, \lambda)$ be the set of subgradients of $\hat{Q}(S, \lambda)$ with respect to $\lambda$ and let

$$g(S, \lambda) \in \partial_\lambda \hat{Q}(S, \lambda).$$

Then, we can perform standard subgradient methods to obtain $\hat{Q}(S, \lambda^*)$ (see Shor [13] for a review). The basic step of a subgradient method is to iteratively update $\lambda$ using, for iteration $l$:

$$\lambda^{l+1} = \max\{\lambda^l + \rho_l \cdot g(S, \lambda^l), 0\}, \tag{42}$$

where the step sizes $\rho_l$, $l = 1, 2, \ldots$, satisfy

$$\rho_l > 0, \quad \sum_{l=1}^{\infty} \rho_l = \infty \quad \text{and} \quad \sum_{l=1}^{\infty} (\rho_l)^2 < \infty. \tag{43}$$

Notice that the decision variables $y^k$ are now dependent on $S$, $\lambda$, and $\omega$, so we can write $y^k = y^k(S, \lambda, \omega)$. Define

$$\hat{y}_{ij}^k(S, \lambda, \omega) = \sum_{(i'j') \in B_{ij}} y_{i'j'}^k(S, \lambda, \omega),$$

$$\bar{y}_{ij}(S, \lambda) = E_\omega\{\sum_k \sum_{(i'j') \in B_{ij}} y_{i'j'}^k(S, \lambda, \omega)\},$$

where $\hat{y}_{ij}^k(S, \lambda, \omega)$ is the amount of commodity $k$ using link $(i, j)$ in the original network and $\bar{y}_{ij}(S, \lambda)$ represents the total expected flow on $(i, j)$. Then, we show in the following proposition that the vector

$$g(S, \lambda) = \bar{y}(S, \lambda) - \bar{\xi} \tag{44}$$

is a valid subgradient of $\hat{Q}(S, \lambda)$ at $\lambda$.

**Proposition 3.1.** *Let $\hat{Q}(S, \lambda)$ and $g(S, \lambda)$ be defined by (38)–(39) and (44), respectively. Then, $g(S, \lambda)$ is a subgradient of $\hat{Q}(S, \lambda)$ at $\lambda$.*

*Proof.* For any given realization $\omega$, let

$$L(S, \lambda, \omega) = \min_{y^k(S,\lambda,\omega) \in T_k(\omega)} \sum_k c^T \hat{y}^k(S, \lambda, \omega) + \lambda^T(\sum_k \hat{y}^k(S, \lambda, \omega) - \xi(\omega)). \tag{45}$$

Let $\sum_k \hat{y}^{k*}(S, \lambda, \omega)$ be the solution of (45). It is easy to see that $\sum_k \hat{y}^{k*}(S, \lambda, \omega) - \xi(\omega)$ is a subgradient of $L(S, \lambda, \omega)$ for any given $\lambda$. Hence, the following inequality holds:

$$L(S, \lambda, \omega) + (\sum_k \hat{y}^{k*}(S, \lambda, \omega) - \xi(\omega))^T(\hat{\lambda} - \lambda) \ge L(S, \hat{\lambda}, \omega) \quad \forall \hat{\lambda} \ge 0.$$

Taking expectations of both sides, we get

$$\hat{Q}(S, \lambda) + g(S, \lambda)^T(\hat{\lambda} - \lambda) \ge \hat{Q}(S, \hat{\lambda}). \qquad \blacksquare$$

We know from (17) in Proposition 1.1 that the expected link flow of each commodity can be obtained as a byproduct of the tree recourse algorithm while finding the expected recourse function of each individual tree. Therefore, the subgradient $g(S, \lambda)$ immediately follows.

### 3.2. Generation of Trees

In Section 2.1, we assume that $\mathcal{P}_k$ is the set of all possible paths for commodity $k$. However, the cardinality of this set can grow exponentially as the problem size increases. A common strategy to deal with this difficulty is using column generation techniques (see Ford and Fulkerson [4]): We start with a small number of paths and successively generate new paths using the shortest path trees derived from the modified arc costs in each iteration. This strategy avoids the total enumeration of paths. Although adding one path to $\mathcal{P}_k$ in each iteration is not efficient, we can add the $m$ shortest paths to $\mathcal{P}_k^{(l)}$, i.e.:

$$\mathcal{P}_k^{(l+1)} = \mathcal{P}_k^{(l)} \cup \hat{\mathcal{P}}_k^{(l)}(\lambda^{(l)}), \tag{46}$$

where $\hat{\mathcal{P}}_k^{(l)}(\lambda^{(l)})$ is the set of shortest paths added for commodity $k$ in iteration $l$ when the arc costs are $c_{ij} + \lambda^{(l)}$.

Since $\mathcal{P}_k^{(l)}$ is a subset of $\mathcal{P}_k$, we are always dealing with a restricted problem. Therefore, inequality (40) does not hold when partial trees are used, meaning that the ap-

proximation function obtained in each iteration is not necessarily a lower bound for $E_\omega(S, \xi(\omega))$.

## 4. NETWORK RECOURSE DECOMPOSITION ALGORITHM

In this section, we first summarize the steps in the decomposition method and then make some remarks on the implementation.

**Algorithm NRD**

STEP 1. Initialization:
Obtain $S^{(0)}(1)$ and $\lambda^{(0)}$; set iteration counter $l = 0$;

STEP 2. Define the trees:
For given $\lambda^{(l)}$, find a set of shortest paths to update $\mathcal{P}_k^{(l)}$ and $\mathcal{T}_k^{(l)}$;

STEP 3. Obtain an approximation of $E_\omega Q(S(1), \xi(2, \omega))$:

$$\hat{Q}^{(l)}(S^{(l)}(1), \lambda^{(l)}) = \sum_k \hat{Q}_k^{(l)}(S_k^{(l)}(1), \lambda^{(l)}) - \lambda^{(l)T}\bar{\xi},$$

by finding the expected recourse function for each tree subproblem:

$$\hat{Q}_k^{(l)}(S_k^{(l)}(1), \lambda^{(l)}) = E_\omega\{\min_{y^k(S_k^{(l)}(1),\lambda^{(l)},\omega)\in T_k^{(l)}(\omega)} (c(2) + \lambda^{(l)})^T y^k(S_k^{(l)}(1), \lambda^{(l)}, \omega)\}.$$

STEP 4. Update $\lambda$ by

$$\lambda^{(l+1)} = \lambda^{(l)} + \rho_l \cdot g(S^{(l)}(1), \lambda^{(l)}),$$

where $g(S^{(l)}(1), \lambda^{(l)})$ is found using (44) and $\rho_l$ satisfies (43).

STEP 5. Obtain a new solution $x^{(l+1)}(1)$, $S^{(l+1)}(1)$ by solving

$$\min_{x(1)} c^T(1)x(1) + \hat{Q}^{(l)}(S^{(l)}(1), \lambda^{(l)})$$

subject to (2)–(5).

STEP 6. $l = l + 1$; if not terminated, go to step 2.

A numerical example of applying this algorithm to a small network can be found in the next section. In the remainder of this section, we comment on the steps in this algorithm that are not discussed in previous sections.

**Remarks**

1. In step 1, we can simply set $\lambda^{(0)}$ to 0. The initial supply $S^0(1)$ can be obtained by various means. For example, we can randomly draw a realization of the random arc capacities or replace the random arc capacities with their means and solve the resulting deterministic network problem.
2. A possible termination criterion in step 6 is that the algorithm is stopped whenever the relative change of the multipliers $\lambda$ is smaller than a value $\epsilon$, i.e.:

$$\frac{\|\lambda^{(l)} - \lambda^{(l-1)}\|}{\|\lambda^{(l)}\|} < \epsilon.$$

3. Since $\hat{Q}_k^{(l)}(S_k^{(l)}(1), \lambda^{(l)})$ (in step 3) are convex, piecewise linear functions, the minimization problem in step 5 becomes a pure network (see Fig. 1).
4. In this algorithm, the refinement of the approximation $\hat{Q}^{(l)}(S^{(l)}(1), \lambda^{(l)})$ keeps pace with obtaining a new first stage solution $x^{(l)}(1)$. However, if we replace $S^{(l)}(1)$ with a constant vector $\bar{S}(1)$ in the algorithm, then step 5 can be performed only after the approximations are established. This makes the estimation of the expected recourse function independent of the first-stage solution. This property allows us to easily extend the NRD method to multistage problems. In an upcoming paper [2], we attempt to apply the NRD method to a real-world problem in a multistage setting.

## 5. NUMERICAL ILLUSTRATIONS

In this section, we first use a small numerical example to illustrate the steps in the NRD algorithm. Then, we evaluate the quality of the NRD approximations with respect to the true expected recourse function using some randomly created dynamic networks.

### 5.1. A Numerical Example

Consider the network recourse problem given in Figure 4. The random arc capacities are assumed to follow truncated Poisson distributions. We truncate a Poisson variable $\xi_{ij}$ at $k$ if $k$ is the largest integer such that $P\{\xi_{ij} \geq k\} < 0.0001$. The means of the original Poisson variables are shown in Figure 4. We apply the NRD algorithm to this network and show the first two iterations. In this example, we use a fixed supply vector $S$ and skip step 5 (solving the first stage problem). Furthermore, we use the step sizes given by

$$\rho_l = \frac{2.0}{l},$$

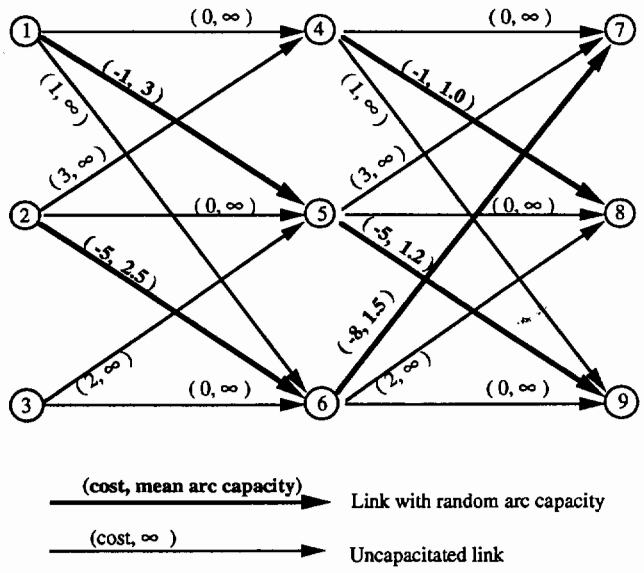



**Fig. 4.** An example of the network recourse problem with 3 origins.

which satisfies condition (43). Moreover, since the uncapacitated links [e.g., link (1, 4)] do not induce bundling constraints in the multicommodity formulation, we show only the expected link flows and the multipliers for the capacitated links.

*Initialization:* We set

$$S^T = [S_1 \quad S_2 \quad S_3] = [4 \quad 4 \quad 4]$$

$$\lambda^{(0)T} = [\lambda_{15}^{(0)} \quad \lambda_{26}^{(0)} \quad \lambda_{48}^{(0)} \quad \lambda_{59}^{(0)} \quad \lambda_{67}^{(0)}]$$

$$= [0 \quad 0 \quad 0 \quad 0 \quad 0].$$

To ensure feasibility, we initialize each tree by including an uncapacitated path:

$$\mathcal{P}_1^{(0)} = \{(1 \to 4 \to 7)\}, \quad \mathcal{P}_2^{(0)} = \{(2 \to 5 \to 8)\},$$
$$\mathcal{P}_3^{(0)} = \{(3 \to 6 \to 9)\}.$$

*Iteration* 1: We obtain the shortest path for each origin:

$$p_1^{(1)} = (1 \to 6 \to 7), \quad p_2^{(1)} = (2 \to 6 \to 7),$$
$$\text{and} \quad p_3^{(1)} = (3 \to 6 \to 7)$$

and update the sets $\mathcal{P}_1^{(1)}$, $\mathcal{P}_2^{(1)}$, and $\mathcal{P}_3^{(1)}$. The corresponding trees $\mathcal{T}_k^{(1)}$ are shown in Figure 5. Next, we find the expected recourse function for each tree and compute the expected link flows. The expected recourse functions of the trees are shown below:

| $s$ | $\hat{Q}_1^{(1)}(s, \lambda)$ | $\hat{Q}_2^{(1)}(s, \lambda)$ | $\hat{Q}_3^{(1)}(s, \lambda)$ |
|---|---|---|---|
| 1 | −5.44 | −9.27 | −6.21 |
| 2 | −8.53 | −13.37 | −9.75 |
| 3 | −9.87 | −14.50 | −11.28 |
| ⋮ | ⋮ | ⋮ | ⋮ |

Then, we update $\lambda$ by using (42). The expected link flows and the updated $\lambda$ are shown below:

| Arc | $c_{ij} + \lambda_{ij}^{(0)}$ | $\bar{\xi}_{ij}$ | $\bar{y}_{ij}^1$ | $\bar{y}_{ij}^2$ | $\bar{y}_{ij}^3$ | $\bar{y}_{ij}$ | $\lambda_{ij}^{(1)}$ |
|---|---|---|---|---|---|---|---|
| (1, 5) | −1.00 | 3.00 | 0 | 0 | 0 | 0 | 0 |
| (2, 6) | −5.00 | 2.50 | 0 | 1.13 | 0 | 1.13 | 0 |
| (4, 8) | −1.00 | 1.00 | 0 | 0 | 0 | 0 | 0 |
| (5, 9) | −5.00 | 1.20 | 0 | 0 | 0 | 0 | 0 |
| (6, 7) | −8.00 | 1.50 | 1.48 | 1.13 | 1.48 | 4.08 | 5.17 |

where $\bar{y}_{ij}^k$ represents the expected flow on link $(i, j)$ in tree $\mathcal{T}_k^{(1)}$ and $\bar{y}_{ij}$ is the total expected flow on this link.

*Iteration* 2: The shortest paths with modified link costs are

$$p_1^{(2)} = (1 \to 5 \to 9), \quad p_2^{(2)} = (2 \to 6 \to 7),$$
$$\text{and} \quad p_3^{(2)} = (3 \to 5 \to 9).$$

The updated trees $\mathcal{T}_k^{(2)}$ are shown in Figure 6. Note that $\mathcal{T}_2^{(2)}$ has only two paths since $p_2^{(2)} = p_2^{(1)}$. The expected recourse functions of the trees are as follows:

| $s$ | $\hat{Q}_1^{(2)}(s, \lambda)$ | $\hat{Q}_2^{(2)}(s, \lambda)$ | $\hat{Q}_3^{(2)}(s, \lambda)$ |
|---|---|---|---|
| 1 | −4.46 | −5.59 | −2.76 |
| 2 | −6.92 | −8.06 | −4.94 |
| 3 | −8.06 | −8.74 | −6.40 |
| ⋮ | ⋮ | ⋮ | ⋮ |

The expected link flows of the trees and the updated multipliers are shown below:

| Arc | $c_{ij} + \lambda_{ij}^{(1)}$ | $\bar{\xi}_{ij}$ | $\bar{y}_{ij}^1$ | $\bar{y}_{ij}^2$ | $\bar{y}_{ij}^3$ | $\bar{y}_{ij}$ | $\lambda_{ij}^{(2)}$ |
|---|---|---|---|---|---|---|---|
| (1, 5) | −1.00 | 3.00 | 1.02 | 0 | 0 | 1.02 | 0 |
| (2, 6) | −5.00 | 2.50 | 0 | 1.13 | 0 | 1.13 | 0 |
| (4, 8) | −1.00 | 1.00 | 0 | 0 | 0 | 0 | 0 |
| (5, 9) | −5.00 | 1.20 | 1.02 | 0 | 1.19 | 2.21 | 1.01 |
| (6, 7) | −2.83 | 1.50 | 1.34 | 1.13 | 1.29 | 3.76 | 7.43 |

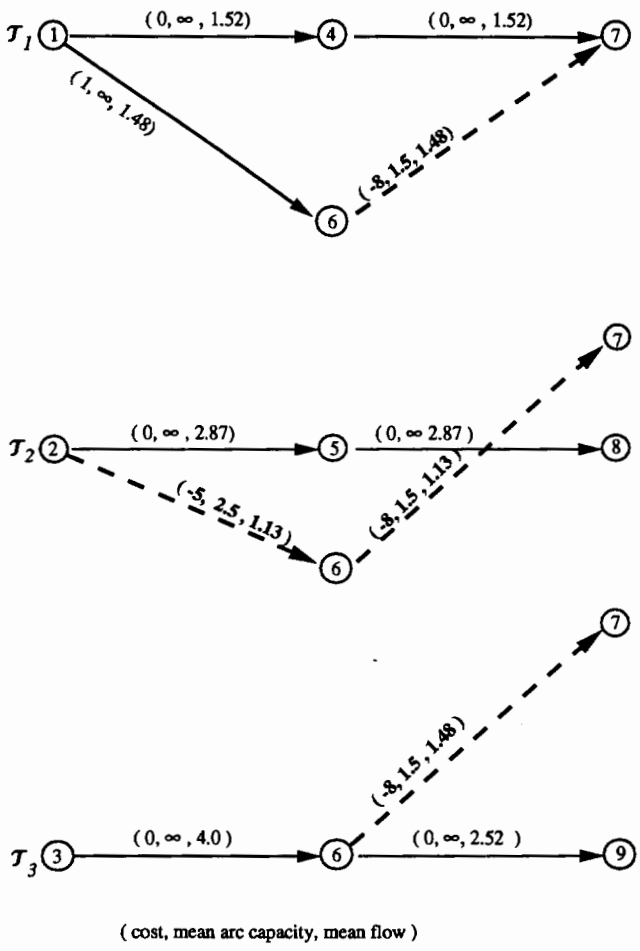



**Fig. 5.** The trees in iteration 1 of the NRD algorithm.

In this example, we see that the overflow on link (6, 7) is due to the relaxation. This results in a relatively high value of the associated multiplier $\lambda_{67}$ that may lower the ranking of the paths containing this link. Consequently, the probability that these paths are being used in the next iteration is reduced. Hence, the expected total flow on this link is decreased. In the next section, we will see how different the approximations with different multipliers could be.

## 5.2. Quality of the Approximations

We have two goals in this section: The first goal is to look at how well the NRD lower bounds are when complete trees are used. The second goal is to compare the shape of the NRD approximations with those obtained by Monte Carlo simulation. In our numerical experiments, the random arc capacities of the randomly generated networks follow truncated Poisson distributions where a random variable $\xi_{ij}$ is truncated at $k$ if $k$ is the largest integer such that $P\{\xi_{ij} \geq k\} < 0.0001$. The means of the untruncated random variables are generated by an exponential distribution with a mean of $v$.

### Lower bound

From Eq. (36), we know that with complete trees the approximation produced by NRD, i.e., $\hat{Q}(S, \lambda)$, is a lower bound of the true expected recourse function $E_\omega Q(S, \omega)$ for a given vector $S$. On the other hand, if we replace the random arc capacities with their expectations and solve the resulting deterministic network, then we obtain another lower bound, known as the *Jensen's bound.* To investigate how tight these bounds are, we compare them with the Monte Carlo estimates using 12 randomly generated networks. For these networks, we use $v = 0.6$ for problems P1–P5 and $v = 0.5$ for problems P6–P12. To obtain the Monte Carlo estimate for a network, we solve, for a fixed vector $S$, the network 400 times with different realizations and obtain the mean total cost. The sizes of the networks and the results of these experiments are shown in Table I.

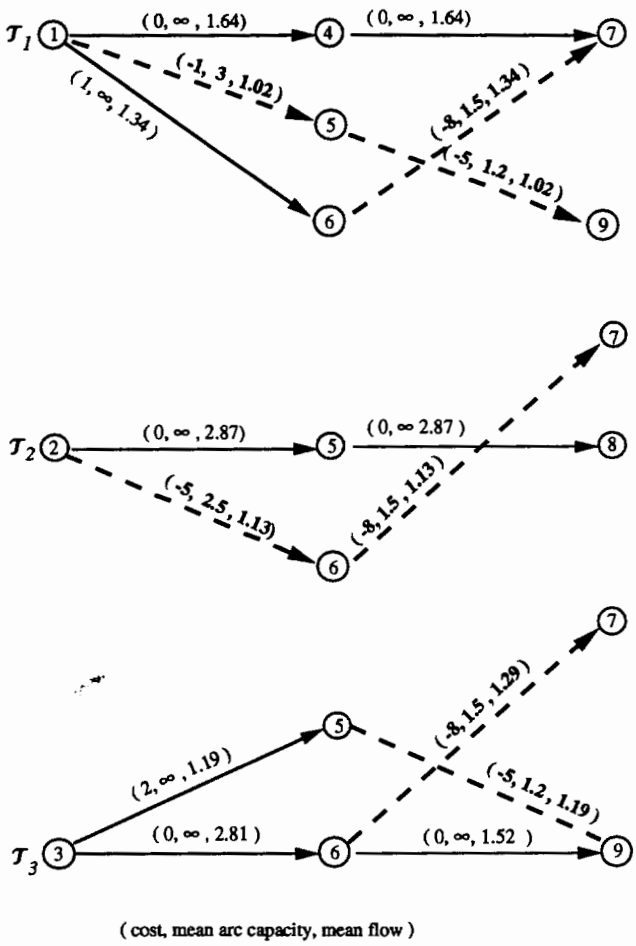



**Fig. 6.** The trees in iteration 2 of the NRD algorithm.

To show how the NRD lower bounds change during the algorithm, we also plot in Figure 7 the values of the Monte Carlo estimate and the lower bounds obtained by NRD on a typical network recourse problem (P9). Both Table I and Figure 7 suggest that NRD can produce reasonably tight lower bounds of expected recourse functions for a fixed $S$. Nevertheless, the main motivation of developing NRD is to obtain separable, convex approximation of the expected recourse function as a parametric function of $S$. Therefore, studying the quality of the NRD lower bound may require further research and is beyond the scope of this paper.

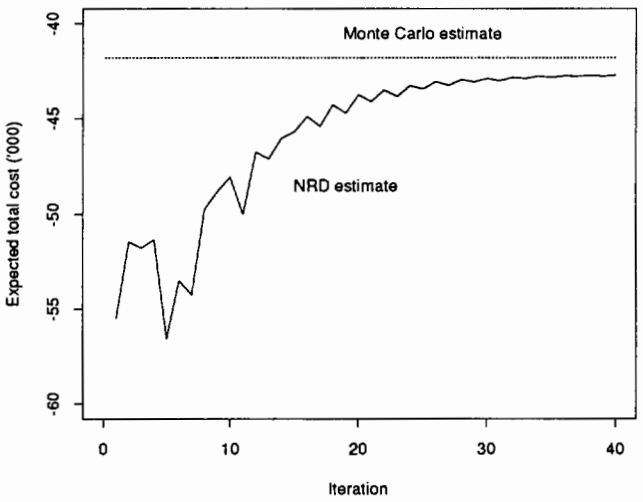



**Fig. 7.** Convergence of NRD lower bounds of the expected recourse function for a network with 60 nodes, 632 arcs (201 random arc capacities).

## Shape of the Approximations

The expected recourse function $E_\omega Q(S, \xi(\omega))$ and its approximations are functions of a multidimensional supply vector $S$. Therefore, to compare their shapes, we look at their rate of change along a specific direction. Let $\bar{Q}_i(S_i)$ be the projection of $E_\omega Q(S, \xi(\omega))$ on the axis defined by $S_i$. We assume that this function has been shifted so that $\bar{Q}_i(0) = 0$. This function measures the expected marginal cost of the each unit of supply entering node $i$ in stage 2. We denote the NRD and Monte Carlo estimates of $\bar{Q}_i(S_i)$ by $\bar{Q}_i^{NRD}(S_i)$ and $\bar{Q}_i^{MC}(S_i)$, respectively. Then, we wish to compare $\bar{Q}_i^{NRD}(S_i)$ and $\bar{Q}_i^{MC}(S_i)$.

Since the NRD approximation is separable in supply nodes, $\bar{Q}_i^{NRD}(S_i)$ is simply the function $\hat{Q}_i(S_i)$ defined in (39). On the other hand, the Monte Carlo estimate, $\bar{Q}_i^{MC}(S_i)$, can be obtained as follows: We vary the amount of $S_i$ while keeping all other supplies $S_j$, $j \neq i$, constant. For each value of $S_i$, we compute the mean objective value of the network recourse problem among $m$ independent realizations. We denote this mean by $\tilde{Q}_i^{MC}(S_i)$. Then, the value of $\bar{Q}_i^{MC}(S_i)$ when $S_i = k$ is given by

$$\bar{Q}_i^{MC}(k) = \tilde{Q}_i^{MC}(k) - \tilde{Q}_i^{MC}(0).$$

To ensure the convexity of the resulting approximation, we use the same set of realizations for different values of $S_i$.

**TABLE I. Comparison of Monte Carlo estimates and NRD lower bounds of the expected recourse functions**

| Problems | | | | Estimates of Expected Total Cost | | | |
|---|---|---|---|---|---|---|---|
| P | No. Nodes | No. Arcs | No. Random Arc Capacities | Monte Carlo Mean | Standard Error | Jensen's Bound | NRD Bound |
| 1 | 30 | 96 | 13 | −2410 | 21 | −2574 | −2465 |
| 2 | 30 | 64 | 24 | −1084 | 20 | −1244 | −1100 |
| 3 | 30 | 156 | 68 | −6243 | 34 | −6864 | −6505 |
| 4 | 30 | 163 | 34 | −6104 | 28 | −6490 | −6266 |
| 5 | 30 | 161 | 34 | −8288 | 34 | −8529 | −8422 |
| 6 | 60 | 530 | 132 | −31391 | 60 | −32039 | −31966 |
| 7 | 60 | 556 | 134 | −40972 | 69 | −41839 | −41334 |
| 8 | 60 | 558 | 266 | −31786 | 90 | −32996 | −32700 |
| 9 | 60 | 632 | 201 | −41795 | 82 | −42391 | −42456 |
| 10 | 60 | 687 | 395 | −42093 | 105 | −43058 | −42912 |
| 11 | 120 | 2320 | 763 | −206738 | 173 | −208852 | −209226 |
| 12 | 120 | 2290 | 742 | −164749 | 162 | −166886 | −166641 |

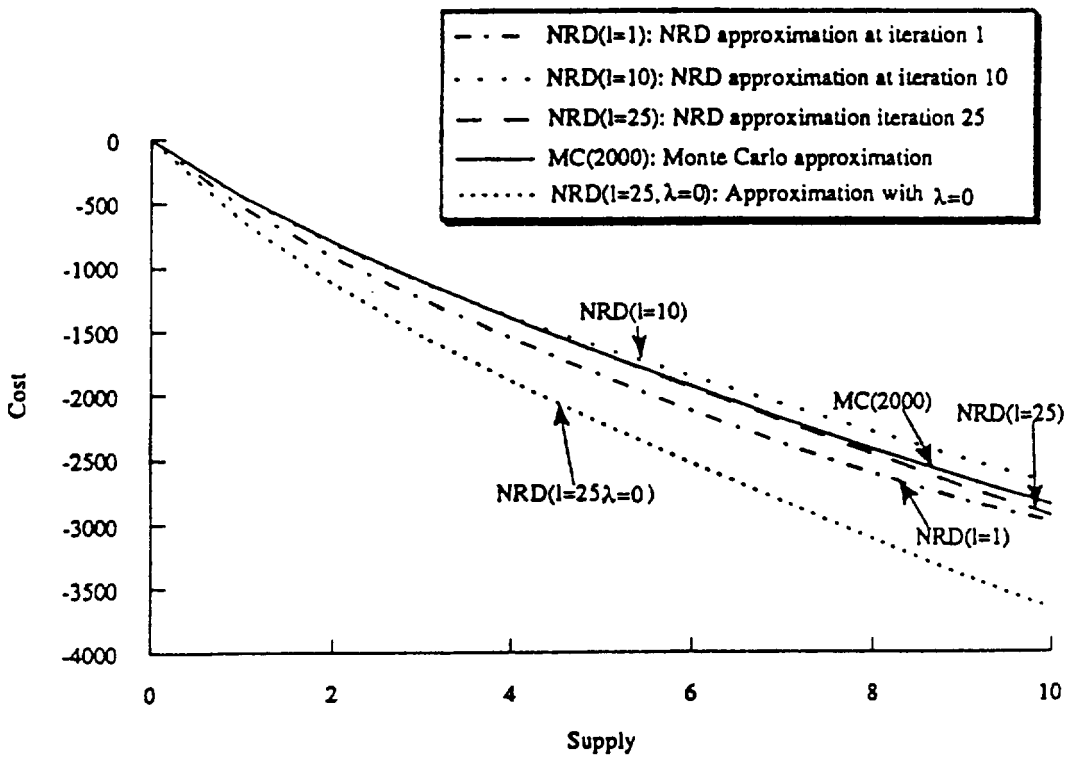



**Fig. 8.** Comparison of the NRD and the Monte Carlo approximations of the network recourse problem with 24 nodes, 101 arcs and 47 random arc capacities.

In our experiments, we apply the NRD algorithm to two randomly generated network recourse problems. The first one has 24 nodes (six are supply nodes), 101 arcs, and 47 random arc capacities. The second one has 80 nodes (20 are supply nodes), 699 arcs, and 483 random arc capacities. For both networks, we use $v = 0.5$ to generate the random arc capacities. In implementing the NRD algorithm, we find 10 shortest paths with modified costs for each supply node in each iteration. Then, we use these paths to update the trees (see step 2 and Section 3.2). We choose $\epsilon = 0.02$ as our stopping criterion. The step size is determined by a modification of the conventional step size rule (see, e.g., [13]): For iteration $l$, we use

$$\rho_l = \gamma_l \cdot \frac{Q^U(S, \lambda) - \hat{Q}^{(l)}(S, \lambda^{(l)})}{\| g(S, \lambda^{(l)}) \|^2},$$

where $\gamma_l \in [0, 2]$ and $Q^U(S, \lambda)$ is a rough estimated upper bound. We choose $\gamma_0 = 2.0$ and decrease it by half after every 10 iterations. For illustration purposes, the number of iterations required for the small and the large problems are set to 25 and 50, respectively. In the Monte Carlo simulation, we use 2000 samples to estimate $\bar{Q}_l^{MC}(s)$. The standard deviation of means are ranging from 26.8 to 34.7 for the smaller problem and from 243 to 260 for the large problem. The approximations for a typical supply node in each problem are shown in Figures 8 and 9. We discuss the results as follows:

Figure 8 shows the Monte Carlo approximation and the NRD approximations in different iterations ($l = 1$, $l = 10$, and $l = 25$). These approximations are labeled MC(2000), NRD(1 = 1), NRD(1 = 10), and NRD(1 = 25), respectively. We also plot the approximation using the set of trees obtained in the 25th iteration, but with $\lambda = 0$ [labeled NRD($1 = 25, \lambda = 0$)]. This variation of the NRD approximation can show the effect of updating $\lambda$ in our algorithm. The NRD approximation with $\lambda = 0$ is clearly shown to be underestimating the true function. This is due to the total relaxation of the bundle constraints without penalizing the constraint violation. However, the NRD approximations with updated $\lambda$ are very close to the Monte Carlo estimates even when $l = 10$. Similar results are obtained for the larger problem as shown in Figure 9. In general, the NRD approximation is neither an upper bound nor a lower bound of the true expected recourse function. For example, NRD(1 = 25) is slightly below MC(2000) in Figure 8 but NRD(1 = 50) is slightly above MC(2000) in Figure 9. It is because we are dealing with a restricted recourse problem in the NRD algorithm. Nevertheless, the approximations depicted in these two

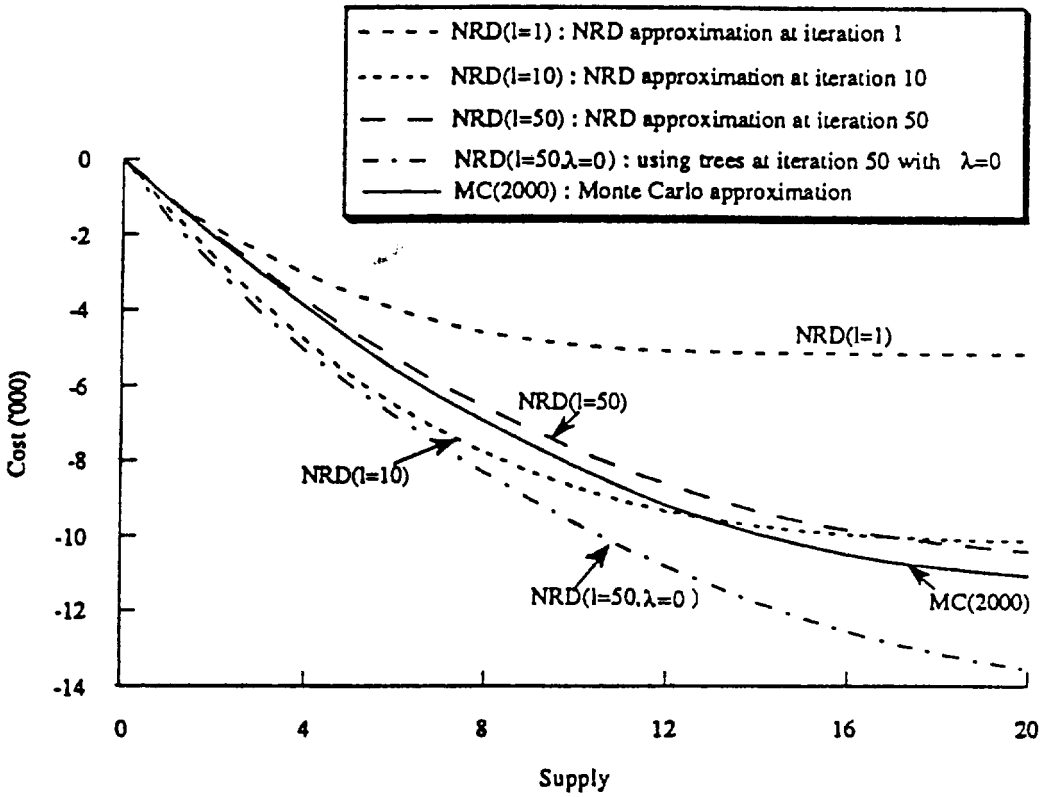



**Fig. 9.** Comparison of the NRD and the Monte Carlo approximations of the network recourse problem with 80 nodes, 699 links and 483 random arc capacities.

figures show that the quality of the NRD approximations can be very high.

## 6. CONCLUSION

We formulate a dynamic network with random arc capacities as a stochastic program with network recourse. Then, we develop a structural decomposition method to approximate the expected network recourse function. In this method, we first use a two-level decomposition scheme on the network recourse problem to produce tree subproblems whose expected recourse functions are easy to compute. These functions provide an approximation of the original expected recourse function. Then, we show how to update the approximation using subgradient optimization and column generation for the trees. Numerical experiments show that the quality of this approximation scheme is quite high. The extension of this method to multistage problems and its application to real-world problems will appear in an upcoming paper.

We would like to thank two anonymous referees for their comments and suggestions which helped to clear up some areas of confusion in an earlier version of this paper. This research was supported in part by Grant DDM-9102134 from the National Science Foundation and in part by Grant F49620-93-1-0098 from the Office of Scientific Research of the Air Force.

## REFERENCES

[1] J. R. Birge and R. J.-B. Wets, Sublinear upper bounds for stochastic programs with recourse. *Math. Program.* **43** (1989) 131–149.

[2] R. K. Cheung and W. B. Powell, An algorithm for multistage dynamic networks with random arc capacitites. Technical Report SOR-92-11, Dept. of Civil Engineering and Operations Research, Princeton University (1992).

[3] Y. Ermoliev, Stochastic quasi-gradient methods. In: Y. Ermoliev and R. Weto (eds.), *Numerical Techniques for Stochastic Optimization.* Springer-Verlag, Berlin (1988).

[4] L. R. Ford and D. R. Fulkerson, A suggested computation for maximal multi-commodity network flow. *Management Sci.* **5** (1958) 97–101.

[5] L. F. Frantzeskakis and W. B. Powell, A successive linear approximation procedure for stochastic, dynamic vehicle allocation problems. *Transport. Sci.* **24**(1) (1990) 40–57.

[6] J. L. Higle and S. Sen, Stochastic decomposition: An algorithm for two stage linear programs with recourse. *Math. Oper. Res.* **16**(3)(1991) 650–669.

[7] P. Kall, A. Ruszczynski, and K. Frauendorfer, Approximation in stochastic programming. *Numerical Techniques for Stochastic Optimization.* Springer-Verlag, Berlin (1988) 313–351.

[8] J. M. Mulvey and H. Vladimirou, Stochastic network programming for financial planning problems. *Management Sci.* **3** (11) (1992) 1642–1664.

[9] W. B. Powell, A comparative review of alternative algorithms for the dynamic vehicle allocation problem. *Vehicle Routing: Methods and Studies.* North-Holland, New York (1988) 249–292.

[10] W. B. Powell and R. K. Cheung, Stochastic programs over trees with random arc capacities. *Networks,* to appear.

[11] L. Qi, An alternating method for stochastic linear programming with simple recourse. *Math. Program. Study* **27** (1986) 183–190.

[12] R. T. Rockafellar and R. J.-B. Wets, Scenarios and policy aggregation in optimization under uncertainty. *Math. Oper. Res.* **16**(1)(1991) 119–147.

[13] N. Z. Shor, *Minimization Methods for Non-Differentiable Function.* Springer-Verlag, Berlin (1985).

[14] R. M. Van Slyke and R. J.-B. Wets, L-shaped linear programs with applications to optimal control and stochastic programming. *SIAM J. Appl. Math.* **17**(4)(1969) 638–663.

[15] S. W. Wallace, Solving stochastic programs with network recourse. *Networks* **16** (1986) 295–317.

[16] S. W. Wallace, A piecewise linear upper bound on the network recourse function. *Math. Program.* **38** (1987) 133–146.

[17] S. W. Wallace, Bounding the expected time-cost curve for a stochastic PERT network from below. *Oper. Res. Lett.* **8** (1989) 89–94.

[18] R. Wets, Programming under uncertainty: The equivalent convex program. *SIAM J. Appl. Math.* **14** (1966) 89–105.

[19] R. J. Wets, Stochastic programming. In: G. Neuhauser, A. H. G. Rinnooy Kan, M. J. Todd (eds.), *Handbooks in OR & MS: Optimization.* Elsevier North-Holland, Amsterdam (1989), Vol. 1, Chap. 8.

[20] W. T. Ziemba, Computational algorithms for convex stochastic programs with simple recourse. *Oper. Res.* **18** (1970) 414–431.

Received May 26, 1992
Accepted February 15, 1994