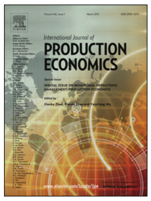
Contents lists available at ScienceDirect

Int. J. Production Economics

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



The dynamic fleet management problem with uncertain demand and customer chosen service level

Ning Shi [a], Haiqing Song [b,*], Warren B. Powell [c]

[a] Business School, Sun Yat-sen University, No. 135 XinGangXi Road, Guangzhou 510275, China
[b] Lingnan College, Sun Yat-sen University, No. 135 XinGangXi Road, Guangzhou 510275, China
[c] Department of Operations Research and Financial Engineering, Princeton University, Princeton, NJ 08544, United States

ARTICLE INFO

Article history:
Received 19 January 2012
Accepted 2 August 2013
Available online 9 October 2013

Keywords:
Stochastic programming
Approximate dynamic programming
Dynamic fleet management
Decomposition method
Multistage networks

ABSTRACT

In this paper, we study a dynamic fleet management problem with uncertain demands and customer chosen service levels. We first show that the problem can be transformed into a dynamic network with partially dependent random arc capacities, and then develop a structural decomposition approach which decomposes the network recourse problem into a series of tree recourse problems (TRPs). As each TRP can be solved by an efficient algorithm, the decomposition approach can solve the problem very efficiently. We conduct numerical experiments to compare its performance with two alternative methods. Numerical experiments show that the performance of our method is quite encouraging.

1. Introduction

We consider the problem faced by a carrier who needs to manage a fleet of homogeneous vehicles over space and time to serve a number of transportation requests. Each transportation request is defined by an origin–destination pair. The service for the request must start at a specific instant in time or be lost. The carrier may provide customers a portfolio of service levels that vary in travel time (24 h, 48 h or 3 days). For example, when customers place orders from E-commercial companies (such as amazon.com, 360buy.com, Taobao.com), they can choose delivery service level ranged from 3 days, 2 days and 1 day with different rates. Carriers, given orders with various service level and demand, need to allocate their fleets to fulfill the tasks. If customer demands a service level with shorter travel time, the carrier may need to pay additional transportation cost and ask for a higher rate. Customers select one service level when they place the order. This problem belongs to a class of dynamic fleet management problems (DFMPs), which can be viewed as a type of spatial, dynamic inventory management problem with reusable vehicles. At any point in space and time, a vehicle may be assigned to satisfy a revenue generating activity. It may be repositioned empty to another point in space and time or held in inventory.

The DFMP has received a lot of attention from the research community. Comprehensive reviews can be found in Dejax and Crainic (1987), Crainic and Laporte (1998), Powell and Topaloglu (2005) and Flatberg et al. (2007). We focus on the most relevant literature. Early fleet management models are deterministic and appear as the first applications of linear programming and min-cost network flow algorithms (see Dantzig and Fulkerson, 1954; White, 1972). These models formulate the problem over a time-space network (also called a *dynamic network*, or a *time-staged network*). A number of authors have studied the problem of random demands. The first to explicitly incorporate uncertainty in demands is Jordan and Turnquist (1983) in the context of the allocation of rail freight cars. Powell (1986) formulates the problem with random demands as a dynamic network with random arc capacities. Powell (1988) provides an overview of alternative modeling and algorithmic strategies for the stochastic fleet management problem. This model has been applied to dynamic truck allocation (Powell, 1987; Powell et al., 1988), empty container repositioning (Crainic et al., 1993; Chu, 1995) and transportation planning (Barbarosoglu and Arda, 2004). Godfrey and Powell (2002) and Topaloglu and Powell (2006) introduce the idea of adaptively estimating piecewise-linear approximations of the value of vehicles in the future.

All of this literature assumes that the time for completing the transportation request is fixed. In our motivating application, the time is not fixed but determined by the service level chosen by the customer. Therefore, the service time for a particular demand can be regarded as a random variable, and it becomes known as soon as the demand is known. But for orders moving in the future, we do not yet know the customer's choice, and for this reason the travel time is stochastic. We note that this type of randomness in the time

* Corresponding author. Tel.: +86 20 84113598; fax: +86 20 84114823.
E-mail address: songhq@mail.sysu.edu.cn (H. Song).

0925-5273/$ - see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.ijpe.2013.09.010

to complete a service contrasts with randomness resulting from delays due to weather, congestion and equipment failures. In these settings, the service completion time only becomes known after the demand has been served.

A few authors have considered the problem of random travel times. Cheung et al. (2005) proposes a labeling method to handle uncertain service times. This method is able to handle a number of operational constraints, but does not scale to larger problems. Glockner and Nemhauser (2000) use scenario trees in a stochastic programming model to handle uncertainty. Simao et al. (2009) use approximate dynamic programming to model truckload operations which includes random travel times which become known only after a trip is completed.

There is a separate literature that includes customer choice. For example, Zhang and Adelman (2009) incorporate customer choice in airline revenue management. However, customer choice has not been considered in the fleet management literature.

Our solution strategy extends a line of research in stochastic vehicle allocation using separable approximations. This problem class has been most widely studied using the framework of two-stage stochastic programs with network recourse (Wallace, 1986, 1987; Birge and Wallace, 1988). Wallace (1986) introduces a piecewise linear upper bound for networks and provides a result that is generalized in Birge and Wallace (1988) for stochastic programs. Independently, separable, piecewise linear approximations have been proposed for discrete vehicle allocation problems that arise in the context of fleet management (Powell, 1986). Frantzeskakis and Powell (1990) presents a method called the *Successive Linear Approximation Procedure* (SLAP) for approximating the expected recourse function. It is generalized by the *Successive Convex Approximation Method* (SCAM) in Cheung and Powell (1996). Similar to SLAP, SCAM decomposes the network into a series of trees and expresses the expected recourse function in terms of trees in the network. Furthermore, SCAM generalizes SLAP by using convex, instead of linear, approximations of the expected recourse function. The advantage of the SCAM is that it works quite well even with problems with huge sample space of random outcomes. SCAM and SLAP both estimate piecewise linear functions in a preprocessor. Godfrey and Powell (2002) and Topaloglu and Powell (2006) propose adaptive approximation procedures to estimate piecewise linear, separable approximations.

In this paper, we show that the problem of managing a fleet of vehicles where customers can choose the service level for orders can be formulated as a multistage dynamic network model with partially dependent random variables by an arc transformation. The contributions of this paper are:

1. We formulate for the first time the fleet management problem with customer-chosen service levels. We formulate it as a dynamic network model with partially dependent random arc capacities. As this model retains the network structure, it enables us to apply structural decomposition techniques. Unlike previous dynamic fleet management models where random variables are usually assumed independent, our model allow random variables representing the customer selection of service level be partially dependent. It enables us to consider customer behaviors, which usually introduce a dependency in random demands, in fleet management field.
2. We show that with slight modifications, SCAM works for DFMP with customer chosen service levels. We present a new structural decomposition approach: the *Successive REsource directive Decomposition Method* (SREDM). This approach provides a search mechanism which takes the advantage of the efficiency of solving the sub-problems. Both decomposition methods explicitly take the advantage of the network structure.
3. We evaluate the efficiency of the decomposition method numerically. The results demonstrate that our approach is superior to the modified SCAM, and the gaps between the results of our approach and the lower bound are very tight.

The rest of this paper is organized as follows. Section 2 introduces how to transform the problem to a dynamic network model with partially dependent random arc capacities. Then it provides a multistage stochastic programming formulation. Next, Section 3 presents the structural decomposition approach that formulates the problem as a *Discrete Resource Allocation Model* (DRAM) and decomposes the problem into a number of stochastic tree recourse problems. Section 4 compares this approach with the alternative methods on a set of test problems. The results in Section 4 show the superiority of the new method over the alternative methods. Finally, Section 5 gives some concluding remarks.

2. Problem formulation

In this section, we first introduce an arc transformation. By this transformation, any arc with a random travel time can be transformed into a set of arcs with deterministic travel times and dependent random arc capacities. Then, the problem is formulated as a dynamic network model with dependent arc capacities.

2.1. Arc transformation

The left part of Fig. 1 is a typical time-space network. The vertical dimension represents the geographical locations and the horizonal dimension represents the discrete times. In the example on the left of Fig. 1, there are two locations i and j, and there are three time periods. Let us define,

$\mathcal{L}$	the set of locations
Γ	the set of discrete time periods
u_{ijt}	the demand, in terms of the number of movements, from location i to location j starting at time t

We use i_t to denote a node representing a location i at time t. Accordingly, $(i_t, j_{t'})$ denotes the arc from i_t to $j_{t'}$.

Now we are ready to illustrate the transformation by Fig. 1. Assume that the arc $(i_t, j_{t'})$, where $t' = t + \tau_{ijt}$, represents the movement from location i to location j starting at time t, where

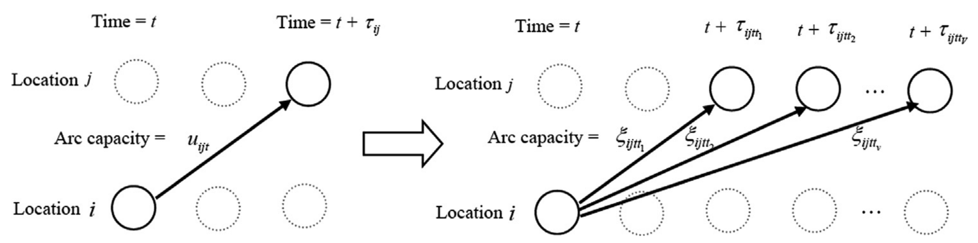


Fig. 1. Transformation of an arc with random travel time and random capacity to a set of arcs with deterministic travel times and random arc capacities.

τ_{ijt} is the travel time from location i to location j starting at time t. If the travel time is deterministic, the destination of this movement can be represented by the node $j_{t'}$. When τ_{ijt} is a random variable, such a destination cannot be presented by a single node in the time-space network. Suppose that τ_{ijt} is a discrete random variable with V possible integer values taken from the set $\mathcal{T}_{ijt} = \{\tau_{ijtt_1}, \ldots, \tau_{ijtt_v}, \ldots, \tau_{ijtt_V}\}$. Each realization of the travel time corresponds to a service level.

Let

$$\delta_{ijtt_v} = \begin{cases} 1 & \text{if the realization of } \tau_{ijt} \text{ is } \tau_{ijtt_v}, \\ 0 & \text{otherwise.} \end{cases}$$

Since the indicator depends on τ_{ijt}, it is a random variable. When the travel time is random, corresponding to the arc $(i_t, j_{t'})$, we generate a node set $N_j[i,t] = \{j_{t_v}, v \in [1,V]\}$ and an arc set $A_j[i,t] = \{(i_t, j_{t_v}), v \in [1,V]\}$. Each element in this arc set can now represent one possible movement from i to j for the realized travel time τ_{ijt_v}. For the deterministic case, $|A_j[i,t]| = 1$. Similarly, we define $O_i[j,t] = \{i_{\bar{t}_1}, \ldots, i_{\bar{t}_v}, \ldots, i_{\bar{t}_{V'}}\}$ as the set of nodes which can reach location j at time t, where V' is the total number of such kind of nodes. Accordingly, the arc set $D_i[j,t] = \{(i_{\bar{t}}, j_t) : i_{\bar{t}} \in O_i[j,t]\}$.

The travel time for arc (i_t, j_{t_v}) is fixed at τ_{ijt_v}, and the arc capacity of (i_t, j_{t_v}), denoted by ξ_{ijtt_v}, is defined as

$$\xi_{ijtt_v} = u_{ijt} \cdot \delta_{ijtt_v}. \tag{1}$$

From (1), we can see that when an arc $(i_t, j_{t'})$, $j_{t'} \in N_j[i,t]$, has a positive arc capacity, then the rest of the arcs in $A_j[i,t]$ must have a capacity of 0. This dependency leads to

$$\sum_{j_{t'} \in N_j[i,t]} \xi_{ijtt'} = u_{ijt}. \tag{2}$$

By this transformation, a transportation request with an uncertain demand and an uncertain travel time can be represented by a bundle of arcs with dependent arc capacities. The dynamic fleet management problem with uncertain demands and uncertain travel times, thus, can be modeled by dynamic network with partially dependent arc capacities. In this paper, we assume that the travel time of empty move, $\tau_{ijt}{}^{e}$ is deterministic.

To facilitate the presentation, in the remaining part of this paper, we define the set of outgoing arcs from node i_t as $A[i,t] = \bigcup_{j \in \mathcal{L},\, j \neq i} A_j[i,t]$. Correspondingly, we define the end nodes of the arcs in $A[i,t]$ as $N[i,t] = \bigcup_{j \in \mathcal{L}, j \neq i} N_j[i,t]$. Similarly, we define the set of arcs entering node j_t as $D[j,t] = \bigcup_{i \in \mathcal{L}, i \neq j} D_i[j,t]$. And, we define the origin nodes of the arcs in $D[j,t]$ as $O[j,t] = \bigcup_{i \in \mathcal{L}, i \neq j} O_i[j,t]$.

2.2. Formulation

Let $(\Omega, \mathcal{F}, P)$ be a probability space and $\omega \in \Omega$ an outcome. For outcome ω, the vector of the realized arc capacities is $\xi(\omega)$. Define:

$\mathcal{N}_t$	the set of nodes from stage t to the end stage
ξ_t	the realized capacities up to stage t
R_{it}	the total supply at location i at time t
$x_{ijtt'}$	flow of loaded vehicles in the arc $(i_t, j_{t'})$
y_{ijt}	flow of empty vehicles from location i to location j at time t
$r_{ijtt'}$	net profit per loaded vehicle in the arc $(i_t, j_{t'})$
c_{ijt}	cost per vehicle for empty moving from location i to location j at time t

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

$S_{tjt'}$	internal supply to location j at time t' resulting from decisions up to stage t

Let the vector $S_t = \{S_{tjt'} : \forall j_{t'} \in \mathcal{N}_t\}$ summarize the states of the system just after stage t, $t = 1, 2, \ldots, T$. It is used to communicate the decisions in stage t with the decisions in later stages. The complete information of the total supply at one location j at time t, i.e., R_{jt}, is known at stage t (that is, it is only incompletely known before stage t) and can be obtained by

$$R_{jt} = S_{(t-1)jt}. \tag{3}$$

We assume that the values of $R_{j1}, j \in \mathcal{L}$ are given.

The problem can now be formulated as the following stochastic optimization problem:

$$\min \sum_{i \in \mathcal{L}} \sum_{j_t \in N[i,1]} -r_{ij1t} x_{ij1t} + \sum_{i \in \,\in \mathcal{L}} c_{ij1} y_{ij1} + \mathbb{E}[Q_2(S_1, \xi_2)] \tag{4}$$

subject to

$$\sum_{j_t \in N[i,1]} (x_{ij1t} + y_{ij1}) = R_{i1} \quad \forall\ i \in \mathcal{L}, \tag{5}$$

$$\sum_{i_1 \in O[j,t]} x_{ij1t} + y_{ij1} = S_{1jt} \quad \forall j_t \in N[i,1], 1 + \tau^{e}_{ij1} = t, \tag{6}$$

$$\sum_{i_1 \in O[j,t]} x_{ij1t} = S_{1jt} \quad \forall j_t \in N[i,1], 1 + \tau^{e}_{ij1} \neq t, \tag{7}$$

$$0 \le x_{ij1t} \le \xi_{ij1t} \quad \forall i \in \mathcal{L},\ \forall j_t \in N[i,1], \tag{8}$$

$$\sum_{j_t \in N[i,1]} \xi_{ij1t} = u_{ij1} \quad \forall j_t \in N[i,1], \tag{9}$$

where for a given supply vector S and a particular realization $\xi_t(\omega)$, $Q_t(S_{t-1}, \xi_t)$ is the value of a minimization problem which is defined recursively as follows:

$$Q_t(S_{t-1}, \xi_t(\omega)) = \min \sum_{i \in \mathcal{L}} \sum_{j_{t'} \in N[i,t]} -r_{ijtt'} x_{ijtt'} + \sum_{i \in \mathcal{L}} \sum_{j \in \mathcal{L}} c_{ijt} y_{ijt} + \mathbb{E}[Q_{t+1}(S_t, \xi_{t+1})] \tag{10}$$

subject to

$$\sum_{j_{t'} \in N[i,t]} (x_{ijtt'} + y_{ijt}) = R_{it} \quad \forall i \in \mathcal{L}, \tag{11}$$

$$S_{(t-1)jt'} + \sum_{i_t \in O[j,t']} x_{ijtt'} + y_{ijt} = S_{tjt'} \quad \forall j_t \in N[i,t], t + \tau^{e}_{ijt} = t' \tag{12}$$

$$S_{(t-1)jt'} + \sum_{i_t \in O[j,t']} x_{ijtt'} = S_{tjt'} \quad \forall j_{\hat{t}} \in N[i,t], t + \tau^{e}_{ijt} \neq t' \tag{13}$$

$$0 \le x_{ijtt'} \le \xi_{ijtt'} \quad \forall i \in \mathcal{L},\ \forall j_{t'} \in N[i,t], \tag{14}$$

$$\sum_{j_{t'} \in N[i,t]} \xi_{ijtt'} = u_{ijt} \quad \forall j_{t'} \in N[i,t], \tag{15}$$

where $t = 2, 3, \ldots, T$ and $Q_{T+1}(\cdot, \cdot) = 0$.

The assumption $Q_{T+1}(\cdot, \cdot) = 0$ means that all vehicles arriving at a time beyond the planning horizon are considered to have a salvage value of 0. The study of the end effects is beyond the scope of this study. Also, the decision variables $x_{ijtt'}$ and y_{ijt} are integer variables.

The problem defined by (4)–(15) is a typical multistage stochastic program with network recourse where (4)–(9) are called the *first stage* problem and (10)–(15) are called the *stage t recourse* problem. Eq. (5) shows that the flow out of each node i_1, $i \in \mathcal{L}$ should be equal to the supply at this node. Eqs. (6) and (7) show that the flow out of the nodes in stage 1 into any node $j_{t'} \in \mathcal{N}_1$ is recorded as $S_1(j, t')$. Eq. (8) shows that the flow on each revenue arc is bounded by its capacity. Eq. (9) comes from the definition of the arc capacities and show the dependency of the random arc capacities.

The expectation function $\mathbb{E}[Q_{t+1}(S_t,\xi_{t+1})]$ is called the *stage* $t+1$ *expected recourse function*. The nested expectation functions which appear in the objective functions make the problem extremely complicated. In order to solve the first stage problem, we need to determine the expected recourse function $\mathbb{E}[Q_2(S_1,\xi_2)]$ as a function of S_1. However, in stage 2, we again need to evaluate the function $\mathbb{E}[Q_3(S_2,\xi_3)]$ and so forth. Except for a few special cases, the exact calculation of the expected recourse functions is numerically intractable, even for moderate problems with a small number of stages. Therefore, we rely on approximation methods. An overview of various approximation schemes for general stochastic programs can be found in Birge and Louveaux (1997) and Kall and Mayer (2005) and the references cited there. In our problem, we have a very special structure in each stage, namely a *transportation problem* with random arc capacities where no specific amount of flow must be shipped to the demand points. The special structure allows the development of specialized method to approximate the expectation functions using some simpler functions. Both the SLAP algorithm in Frantzeskakis and Powell (1990) and the SCAM algorithm in Cheung and Powell (1996) are successful approximation approaches. However, one important assumption of SLAP and SCAM is that all arc capacities are independent. Therefore, both SLAP and SCAM cannot solve the problem directly. Glockner and Nemhauser (2000) attempt to tackle the problems where the dependency of arc capacities is defined by a set of scenarios. Then they propose a scenario aggregation approach. The efficiency of this approach depends on the number of scenarios.

3. Successive resource-directive decomposition method

Recently, Shi et al. (2007) introduce a new specialized algorithm to solve the problem where the network has a directed tree structure with partially dependent arc capacities. Our ability to solve tree problems motivates us to solve problems with general network structure. In this section, we introduce a new decomposition approach: the Successive REsource-Directive Decomposition Method (SREDM). Similar to SCAM in Cheung and Powell (1996), SREDM also uses a backward recursion framework to successively provide convex approximations of the expected recourse functions at each stage. At each stage, the approximation can be achieved by structurally decomposing the underlying network into tree problems. Then, each tree problem can be solved by the algorithm in Shi et al. (2007). The tree problem here is much more difficult than that in Cheung and Powell (1996), SREDM uses a different decomposition approach to decompose the network into tree problems. The decomposition approach in SCAM falls into the category of *Price-directive* decomposition that penalizes the violation of the bundle constraints (it will be introduced in detail later) and obtains a lower bound of the recourse function, while SREDM belongs to *Resource-directive* decomposition approach that re-formulates the problem as a *Discrete Resource Allocation Model* (DRAM) and achieves an upper bound of the recourse function.

SREDM successively generates a sequence of functions starting from the last stage, that is,

$$\hat{Q}_T(S_{T-1}),\ldots,\hat{Q}_{t+1}(S_t),\quad \hat{Q}_t(S_{t-1}),\ldots,\hat{Q}_2(S_1),$$

where the function $\hat{Q}_t(S_{t-1})$ is used to approximate the expected total cost for stages from t to T as a parametric function of S_t. Furthermore, each function $\hat{Q}_t(S_{t-1})$ is convex and has a *separable* form of

$$\hat{Q}_t(S_{t-1})=\sum_{j_{t'}\in\mathcal{N}_t}\hat{Q}_{j,t'}(S_{(t-1)jt'}) \tag{16}$$

where $\hat{Q}_{j,t'}(S_{(t-1)jt'})$ is piecewise linear and convex. This *separable* form is very important to SREDM. The physical meaning of the function $\hat{Q}_{j,t'}(S_{(t-1)jt'})$ will be introduced in Section 3.1.

Below, we summarize the major steps of SREDM.

*Step*1: *Initialization*
Set $t=T$. $\hat{Q}_{t+1}(S(t))=0$.

*Step*2: *Network augmentation and the modified stage t problem*
Approximate the stage t problem by replacing $\mathbb{E}[Q_{t+1}(S(t),\xi_{t+1})]$ with the approximation $\hat{Q}_{t+1}(S(t))$. Let us denote the modified stage t problem as $[MP_t]$. Represent $\hat{Q}_{t+1}(S(t))$ by a set of links such that $[MP_t]$ is a network with random arc capacities (that is, augment the original stage t network with additional links).

*Step*3: *Re-formulate as a multi-commodity flow problem*
Reformulate $[MP_t]$ as a multi-commodity flow problem with bundle constraints.

*Step*4: *Re-formulate as a DRAM*
To tackle the bundle constraints, we re-formulate the problem further as a DRAM.

*Step*5: *Solve DRAM by Local Search Procedure*
We solve the DRAM by a local search procedure that is implemented by an Arc-Switch Procedure. By solving a series of tree recourse problems, the method produces a function, denoted by $\hat{Q}_t(S(t-1))$, that approximates the exact objective value of $[MP_t]$ as a function of $S(t-1)$.

*Step*6: *Termination*
Set $t=t-1$. If $t=0$, terminate the algorithm. Otherwise, repeat Step 2 to Step 5.

We describe the network augmentation and the modified stage t problem (Step 2), the multi-commodity flow formulation (Step 3) and DRAM (Step 4) in the following. The local search procedure for solving the DRAM model and the Arc-Switch Procedure are put in the appendix. Note that, in the local search procedure, we need to solve a number of tree recourse problems. The tree recourse problem itself is a very difficult problem and a summary can be found in the appendix.

3.1. Network augmentation and the modified stage t problem

By replacing $\mathbb{E}[Q_{t+1}(S(t),\xi_{t+1})]$ with the approximation $\hat{Q}_{t+1}(S(t))$ in (10), the stage t problem is modified as

$$[MP_t]\quad \hat{Q}_t(S(t-1))=E(\tilde{Q}_t(S(t-1),\xi_t)) \tag{17}$$

where

$$\tilde{Q}_t(S(t-1),\xi_t)=\min\left\{-r_tx_t+c_ty_t+\sum_{j_{t'}\in\mathcal{N}_{t+1}}\hat{Q}_{j,t'}(S_{j,t'}(t)).\quad \text{subject to : (11)–(15)}\right\} \tag{18}$$

Let $S_{j,t'}(t)=s$. From this formulation, we see that $\hat{Q}_{j,t'}(s)$ has an intuitive interpretation: it approximates the expected marginal value of having s vehicles in location j at time t'. Suppose these vehicles are ordered such that the first one takes the most profitable movement, the second takes the second best, and so forth. Let $q^k_{jt'}$ be the expected marginal contribution of the kth vehicle. Then $\hat{Q}_{j,t'}(s)$ can be written as

$$\hat{Q}_{j,t'}(s)=\sum_{k=1}^{s}q^k_{jt'}. \tag{19}$$

As shown in Cheung and Powell (1996), the functions $\hat{Q}_{j,t'}(s)$ are convex piecewise linear on s, that is,

$$q^k_{jt'} \le q^{k'}_{jt'} \quad \text{if } k \ge k'. \tag{20}$$

This inequality reflects the diminishing return of the incremental unit of supply. In other words, the first vehicle available in location j should have a higher expected marginal value (or lower expected marginal cost) than the second one.

Since each function $\hat{Q}_{j,t'}(s)$ is convex and piecewise linear, it can be represented by a set of deterministic links, which we call “recourse links” (see Fig. 2). The kth recourse link out of the source node has an arc capacity of 1 and an arc cost of $q^k_{jt'}$. The last recourse link has infinite arc capacity with zero cost. This link is used to represent the “holding” option in case where a large number of vehicles are being sent to node j. We call this link “feasibility link”.

As a result, the impacts of current decisions are captured by the recourse links. Mathematically speaking, we let,

$z^k_{jt'}$	the flow on the kth recourse link out of node j for a realization ξ_t
$\mathfrak{L}_{jt'}$	the set of recourse links out of node $j_{t'}$, and
$\mathfrak{L}'_{jt'}$	the set of recourse links out of node $j_{t'}$ excluding the feasibility link

Therefore $\tilde{Q}_t(S(t-1),\xi_t)$ can be reformulated as

$$\tilde{Q}_t(S(t-1),\xi_t(\omega)) = \min \sum_{i \in \mathcal{L}} \sum_{j_{t'} \in N[i,t]} \left(-r_{ijtt'}x_{ijtt'} + c_{ijt}y_{ijt} + \sum_{k \in \mathfrak{L}_{jt'}} q^k_{jt'} z^k_{jt'} \right) \tag{21}$$

subject to

$$\sum_{j_{t'} \in N[i,t]} x_{ijtt'} + \sum_{j \in \mathcal{L}} y_{ijt} = R_{it} \quad \forall i \in \mathcal{L}, \tag{22}$$

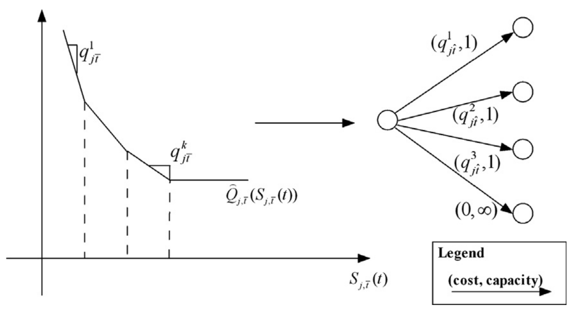


Fig. 2. Piecewise linear approximation and recourse links.

$$\sum_{i_t \in O[j,t']} x_{ijtt'} + y_{ijt} - \sum_{k \in \mathfrak{L}_{jt'}} z^k_{jt'} = 0 \quad \forall j_{t'} \in N[i,t], t+\tau^e_{ij1} = t' \tag{23}$$

$$\sum_{i_t \in O[j,t']} x_{ijtt'} - \sum_{k \in \mathfrak{L}_{jt'}} z^k_{jt'} = 0 \quad \forall j_{t'} \in N[i,t], t+\tau^e_{ij1} \ne t' \tag{24}$$

$$0 \le x_{ijtt'} \le \xi_{ijtt'} \quad \forall i \in \mathcal{L},\ j_{t'} \in N[i,t], \tag{25}$$

$$0 \le z^k_{jt'} \le 1 \quad \forall j_{t'} \in N[i,t],\ k \in \mathfrak{L}'_{jt'}, \tag{26}$$

$$\sum_{j_{t'} \in N[i,t]} \xi_{ijtt'} = u_{ijt} \quad \forall i \in \mathcal{L}. \tag{27}$$

Clearly, Problem (21)–(27) is an acyclic network. Such a network consists of a number of two-level trees as the first-level arcs have partially dependent arc capacities and the second-level arcs represent the recourse links.

3.2. Multi-commodity flow problem formulation (MCPF)

To approximate the expected value of problem (21)–(27) as a separable function of the supply (meaning the number of vehicles available in each location), we can decompose the network by the origins. By reviewing any flow originating from one origin as one commodity, we can re-formulate the problem as a multi-commodity flows problem with dependent random arc capacities. Taking the example in Fig. 3(a) for instance, the flow entering node 1 is considered as commodity 1 and the flow entering node 2 is considered as commodity 2. It suggests that we should differentiate the flows by their origins. Let

$z^k_{ijtt'}$: the flow of commodity i on the kth recourse link out of node $j(t')$.

Therefore, the multi-commodity formulation of problems (21)–(27) can be written as

$$\tilde{Q}_t(S(t-1),\xi_t) = \min \sum_{i \in \mathcal{L}} \sum_{j_{t'} \in N(i,t)} \left(-r_{ijtt'}x_{ijtt'} + c_{ijt}y_{ijt} + \sum_{k \in \mathfrak{L}_{jt'}} q^k_{j,t'} z^k_{ijtt'} \right) \tag{28}$$

subject to

$$\sum_{j_{t'} \in N[i,t]} x_{ijtt'} + y_{ijt} = R_{it} \quad \forall i \in \mathcal{L}, \tag{29}$$

$$x_{ijtt'} + y_{ijt} - \sum_{k \in \mathfrak{L}_{jt'}} z^k_{ijtt'} = 0 \quad \forall i \in \mathcal{L},\ j_{t'} \in N[i,t], t+\tau^e_{ij1} = t' \tag{30}$$

$$x_{ijtt'} - \sum_{k \in \mathfrak{L}_{jt'}} z^k_{ijtt'} = 0 \quad \forall i \in \mathcal{L},\ j_{t'} \in N[i,t], t+\tau^e_{ij1} \ne t' \tag{31}$$

$$0 \le x_{ijtt'} \le \xi_{ijtt'} \quad \forall i \in \mathcal{L},\ j_{t'} \in N[i,t], \tag{32}$$

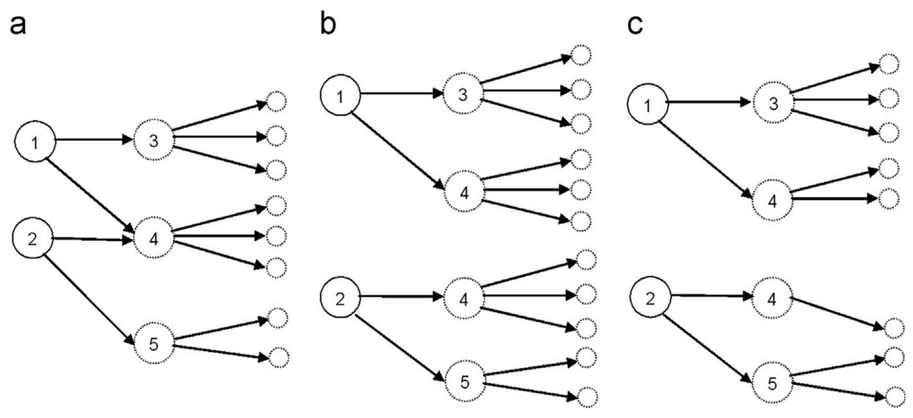


Fig. 3. Two stage network. (a) Two level trees. (b) Price-directive decomposition. (c) Resource-directive decomposition.

$$\sum_{j_{t'}\in N[i,t]} \xi_{ijtt'} = u_{ijt} \quad \forall i \in \mathcal{L}, \tag{33}$$

$$0 \le \sum_i z^k_{ijtt'} \le 1 \quad \forall i \in \mathcal{L},\ j_{t'} \in N[i,t],\ k \in \mathfrak{L}'_{jt'}. \tag{34}$$

In this formulation, if we relax the constraint (34), then this problem becomes separable by commodity. Furthermore, constraints (28)–(33) indicate that each commodity i follows a two-level tree which is rooted at supply node i. Note that the links in the second-level of these trees may be shared by different trees (see Fig. 3(a)), inducing the bundle constraints (34). To tackle these bundle constraints, the SCAM in Cheung and Powell (1996) relaxes these constraints by using a set of deterministic multipliers $\lambda^k_{jt'}$. Then, the network is decomposed into trees with overlapped arcs. If the flows in one arc exceed the capacity (namely, the constraint (34) is violated), a penalty price will be added to that arc. Then, the Lagrangian relaxation of the problems (21)–(27) , is

$$L(S,\lambda) = \min \sum_{i\in\mathcal{L}} \sum_{j_{t'}\in N(i,t)} \left(-r_{ijtt'}x_{ijtt'} + c_{ijt}y_{ijt} + \sum_{k\in\mathfrak{L}_{jt'}} q^k_{jt'} z^k_{ijtt'} \right) + \sum_{j_{t'}\in N(i,t)} \sum_{k\in\mathfrak{L}'_{jt'}} \left(\lambda^k_{jt'} \left(\sum_i z^k_{ijtt'} - 1 \right) \right)$$

By relaxing the constraints (34), a lower bound of $E(\tilde{Q}_t(S(t-1),\xi_t))$ can be achieved. By updating the vector of λ, the bound can be tightened. Different rules of updating λ have been discussed in Cheung and Powell (1996). Using the algorithm in Shi et al. (2007), the SCAM can work for dynamic networks with partially random arc capacities as well. This modified SCAM will be served as a benchmark in Section 4.

To tackle the bundle constraints, in this paper, we propose a new decomposition approach that falls into the category of *Resource-directive* decomposition approach. This new approach re-formulates the problem as a DRAM that is solved by an *Arc-Switch Procedure*. The objective function achieved by the new approach is an upper bound of the exact recourse function.

One can also derive the MCPF's equivalent deterministic formulation (Glockner and Nemhauser, 2000) which can be directly solved by commercial solver as CPLEX. However as the problem size increases exponentially with the number of random variables, it is not practical to solve the MCPF by CPLEX.

3.3. Discrete resource allocation model

In this subsection, because we are only concerned with the stage t problem, without introducing ambiguity, we suppress the time index t by writing: $\hat{Q}_t$ as $\hat{Q}$, $S(t-1)$ as S, ξ_t as ξ. To ease the presentation burden, we suppress the time index t' from $j_{t'}$ such that $z^k_{ijtt'}$ is simplified as $z_{ij}{}^k$, $q^k_{jt'}$ as $q_j{}^k$. In other words, the notation j represents one node in $N[i,t]$. By interpreting the empty movement as a special load movement, the notation y_{ijt} is eliminated.

By regarding each recourse link as one vehicle, the bundle constraints can be interpreted as “each vehicle can only be allocated to one commodity”. Thus, we can define a binary variable:

$$b^k_{ij} = \begin{cases} 1 & \text{if recourse link } (j,k) \text{ is assigned to commodity } i, \\ 0 & \text{otherwise.} \end{cases}$$

By definition, we know that,

$$\sum_i b^k_{ij} = 1 \quad \forall j \in N[i,t]. \tag{35}$$

Let $\mathbf{b} = \{b^k_{ij} : \forall i \in \mathcal{L},\ \forall j \in N[i,t]\}$ be one feasible allocation and Λ be the set of all feasible allocations. Given one allocation $\mathbf{b}$, the bundle constraints (34) can be replaced by the following constraint:

$$0 \le z^k_{ij} \le b^k_{ij} \quad \forall i \in \mathcal{L},\ j \in N[i,t],\ k \in \mathfrak{L}'_j. \tag{36}$$

The new constraint (36) together with the constraints (29)–(33) now are all separable for the commodity. Therefore, we can define the problem with a given vector $\mathbf{b}$ as

$$\tilde{Q}(S,\mathbf{b},\xi) = \left\{ \min \sum_{i\in\mathcal{L}} \sum_{j\in N(i,t)} \left(-r_{ij}x_{ij} + \sum_{k\in\mathfrak{L}_j} q^k_j z^k_{ij} \right) : \text{ subject to constraints (29)–(33)} \right\}$$

Therefore, the expected value of $\tilde{Q}(S,\mathbf{b},\xi)$ can be written as $\hat{Q}(S,\mathbf{b},\xi) = \mathbb{E}[\tilde{Q}(S,\mathbf{b},\xi)]$.

We further define the feasible set for commodity i for a given $\mathbf{b}$ as

$T_i(\mathbf{b}) = \{x_{ij}, z^k_{ij} | \text{constraints (29)–(36) are satisfied}\}$.

Now let,

$$\hat{Q}_i(S_i,\mathbf{b}) = \mathbb{E}\left\{ \min_{(x_{ij},z^k_{ij})\in T_i(\mathbf{b})} \left(\sum_{j\in N(i,t)} \left(-r_{ij}x_{ij} + \sum_{k\in\mathfrak{L}_j} q^k_j z^k_{ij} \right) \right) \right\}. \tag{37}$$

Since for any given $\mathbf{b}$, constraints (29)–(36) are separable, we can see that,

$$\hat{Q}(S,\mathbf{b}) = \sum_{i\in\mathcal{L}} \hat{Q}_i(S_i,\mathbf{b}). \tag{38}$$

The embedded minimization problem in (37) is a directed tree with partially dependent random arc capacities. As shown in Shi et al. (2007), the function $\hat{Q}(S,\mathbf{b})$ is piecewise linear and convex in the scalar supply S, and the function can be computed by a pseudo-polynomial algorithm parametrically. Then according to (38), we can obtain $\hat{Q}(S,\mathbf{b})$ easily. Therefore, the problem of $[MP_t]$ defined in (17) now is equivalent to find a best “$\mathbf{b}$” from all feasible ones. That is,

$$\hat{Q}(S) = \min_{\mathbf{b}\in\Lambda} \mathbb{E}[Q(S,\mathbf{b},\xi)]. \tag{39}$$

Note that by definition, any $\mathbf{b}\in\Lambda$ needs to satisfy the constraints defined in (35). This constraint is a typical “budget constraint” that appears in a DRAM (Federgruen and Groenevelt, 1986). The DRAM is in general difficult to solve. There are several polynomial solvable DRAM problems in literature. When the objective function is separable in terms of the “resource”, Hochbaum and Shanthikumar (1990) prove that a greedy algorithm can obtain the optimal solution. Federgruen and Groenevelt (1986) propose the necessary and sufficient conditions for the optimality of the greedy algorithm for DRAM. These conditions do not require the objective functions to be separable. Recently, Murota (1998) generalizes a group of non-separable solvable problems. The objective function of these problems has to be “M-Convex”. Unfortunately, in general our problem does not fall into any category of these solvable problems. Therefore, in this paper, we turn to a local search approach that achieves a local optimal solution. Also, it is straightforward to see that the approximation achieved by such an approach is an upper bound of the exact expected recourse function. Therefore, in this paper, we provide an arc-switch procedure to search an optimal arc allocation. The technical details are put in the appendix.

3.4. Algorithm

In summary, the pseudo algorithm for SRIEM is as follows,
SRIEM

```
Step1: Compute the approximations in stage $N$, $\hat{Q}_N(S_N)$
Step2: Augmenting stage $N-1$ problem by recourse links corresponding to $\hat{Q}_N(S_N)$.
Step3: For $t = N-1$ down to 2,
Step3a: Set $n=0$ and $\mathbf{b}_n$. Decompose the augmented stage $t$ problem by origins.
Step3b: Produce and solve tree subproblems $\hat{Q}_i(S_i, \mathbf{b}_n)$.
Step3c : Update $n = n+1$,
    $\mathbf{b}_n = \arg \min_{\mathbf{b}_n \in N(\mathbf{b}_{n-1})} \{\hat{Q}(\mathbf{b})\}$.
Step3d : Repeat Step 3c until $\hat{Q}(\mathbf{b}_n) - \hat{Q}(\mathbf{b}_{n-1}) \le \varepsilon$.
Step3e : Augment stage $t-1$ problem by the recourse links corresponding to $\hat{Q}(\mathbf{b}_n)$.
Step4 : Solve the augmented first stage problem.
```

To have a clearer picture of this method, we illustrate a step of it using Fig. 4. Fig. 4(a) shows the $t-1$, t and $t+1$ recourse problems. Approximate functions for stage $t+1$ are shown in Fig. 4(b) and they can be represented by a number of recourse links in Fig. 4(c). The augmented stage t problem can be decomposed by a number of tree problems by allocating recourse links to different trees as shown in Fig. 4(d). Each tree problem can be solved by algorithm in Shi et al. (2007). By an arc-switch step, a better approximation can be achieved, shown in Fig. 4(e). With the approximation for stage t problem, we can get the augmented $t-1$ problem shown in Fig. 4(f).

4. Numerical experiments

In this section, we evaluate the SREDM algorithm by comparing it with the alternative methods. We conduct two sets of experiments. In the first set, the demands are deterministic and we compare the SREDM algorithm with a labeling algorithm. In the second set, both demands and service levels are random variables, and we compare the SREDM algorithm with the modified SCAM algorithm. We first introduce the design of the experiment and then report the numerical results.

4.1. Experimental design

The algorithms are implemented in Java and the experiments are conducted on a PC with 2.4 GHz and 1.0 G memory. In our results, we use a posterior lower bound by optimizing the problem after all information becomes known.

4.2. Comparison between SREDM and a labeling algorithm

We generate a set of test problems based on some real experience from a Chinese logistics company and use a planning horizon of 36 periods (representing 36 days with 24-h intervals). There are 10 locations whose positions are randomly located in a 100-by-100 unit square. The origin and the destination of a task are randomly located in those locations. The starting time of a request is uniformly generated in the period [0, 36]. The empty moving cost is set as 15 per unit distance. The holding cost is set to 8 per unit time. The profit for each task is set as a linear function of distance and service level: $r_{ij} = 70d_{ij} - 10\tau_{ij}$ where d_{ij} is the distance from location i to j. Customers have two possible service levels to choose for each task. One service level is set as two periods (2 days) and the other is set as four periods(four days). The customers choose the first service level with a probability p_1 that is drawn from a uniform distribution between 0 and 1. The second service level is chosen with a probability $p_2 = 1 - p_1$.

The labeling algorithm in Cheung et al. (2005) is the first attempt to solve DFMP with uncertain service times. It can be easily modified to solve the problem here. The labeling algorithm

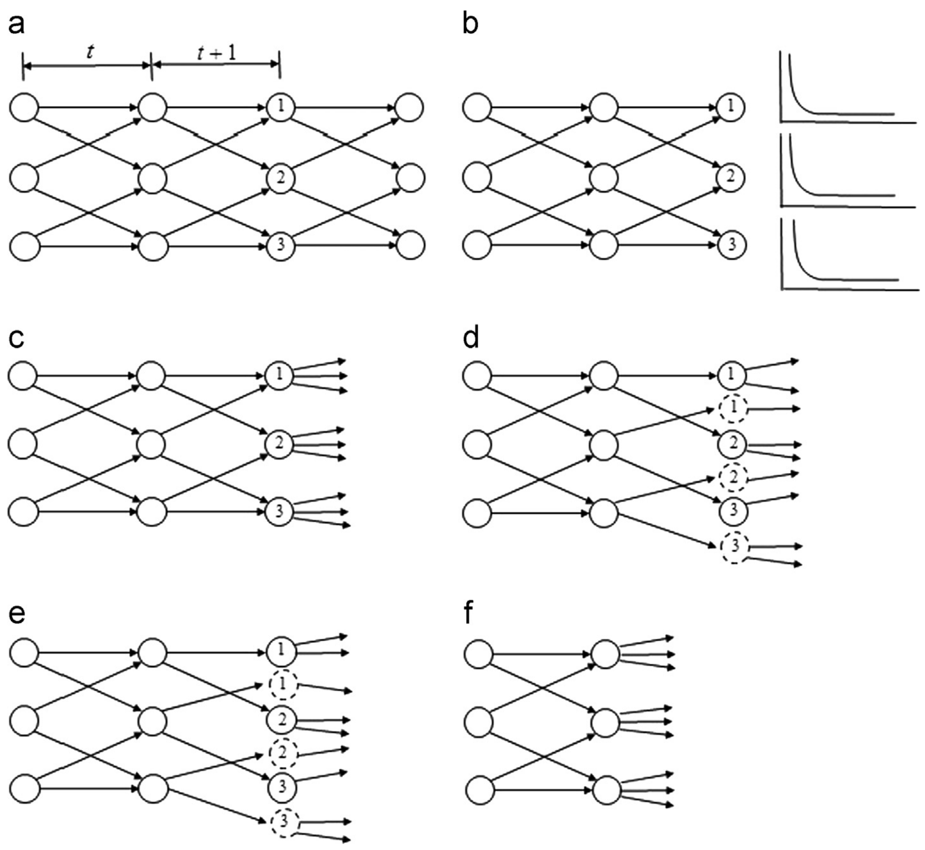


Fig. 4. A step in SRIEM. (a) Recourse problems in stage $t-1$, t and $t+1$. (b) Approximations for stage $t+1$ problem. (c) Augmented stage t problem. (d) Decomposition by origins, producing trees. (e) Arc-switch for better approximation. (f) Augmented $t-1$ problem.

approximates the expected recourse function by generating a set of virtual routes and labels. The advantage of the labeling algorithm is the capability and flexibility of dealing with complex work rules including task time window, drivers' working hours and preferences, etc. However, the labeling algorithm assumes that the demands are deterministic. Due to this limitation, in this set of experiments, we set the demands as deterministic. We fix the number of locations as 10 and the number of vehicles is 20. We considers 8 instances where the number of tasks (in each stage) is increased from 20 to 1000. For each instance, we run 2000 samples and achieved the sample means of poster-bounds, solutions of SREDM and solutions of the labeling algorithm.

The comparison result is listed in Table 1. Column 1 shows the problem size (denoted by N_T). Column 2 records the poster-bound (PB). Column 3 records the solutions achieved by the labeling algorithm and Column 4 records its computational time. Column 5 records the solutions achieved by the SREDM algorithm and column 6 records its computational time. It demonstrates that the SREDM algorithm can achieve much better solution quality than the labeling algorithm. Especially for large scale problems (400 tasks and 1000 tasks), the labeling algorithm cannot achieve a satisfactory solution in 2 h while the SREDM algorithm can obtain one good solution in less than 5 min.

4.3. Comparison between SREDM and a modified SCAM

To evaluate the performance of the SREDM algorithm under uncertainty, we use the similar test problems similar to those used in Cheung and Powell (1996). The problem generator creates the locations in a 100-by-200 rectangle. We simply use the Euclidean distance between each pair of locations as the corresponding travel distance. The net profit for each loaded tractor movement is 70 cents per mile, while the cost of an empty movement is 40 cents per mile. The holding cost is set to 10 cents per time unit. The demand $u_{ij}{}^t$ between location i and location j is assumed to follow a Poisson distribution with mean $m_{ij}{}^t$ which is calculated by

$$m_{ij}^t = \alpha_i \cdot \beta_j \cdot v_t$$

where,

α_i	inbound potential for location i
β_j	outbound potential for location i
v_t	an exponential random variable

The inbound potential and outbound potential for each location capture the location's ability to attract the inbound flows or generate the outbound flows. The inbound potential for a location i, α_i, is drawn uniformly between 0.2 and 1.8. The corresponding outbound potential β_i is obtained by $\beta_i = 2 - \alpha_i$. Therefore, these two potentials are negatively correlated. The motivation for this setting is partly because regions with large inbound flows often have small outbound flows in real world applications. More importantly, under this setting, a myopic algorithm may produce a poor solution since a vehicle may be sent to a location with high inbound potential but with very low outbound demand. (The myopic algorithm simply solves a static deterministic assignment problem at the current stage.) Finally, to capture the randomness in demand, we also include an exponentially distributed random number v_t with mean 1.5, which is the typical daily demand between each pair of locations.

We consider in total 12 periods. Each period represents one day. For each task, the customers have three service levels to choose: 1 periods (one day), 2 periods (two days) and 4 periods (four days). The customer chooses service level 1, service level 2, and service level 3 with probability p_1, p_2 and p_3 respectively. The values of p_1 and p_2 are randomly drawn from a uniform distribution in the range [0, 0.5]. And $p_3 = 1 - p_1 - p_2$.

The results are shown in Table 2. Column 1 and Column 2 show the problem size (N_L represents the number of locations and N_R represents the number of resources). Column 3 records the PB bound. Columns 4, 5 and 6 record the solutions achieved by the myopic algorithm, the modified SCAM and the SREDM respectively. Columns 7, 8 and 9 record the gap of the solutions to the PB. An interesting observation is that when the service level is uncertain, the myopic method has a significant gap with PB. It clearly demonstrates that both the SREDM algorithm and the modified SCAM algorithm significantly outperform the myopic algorithm, and the SREDM algorithm performs slightly better than the modified SCAM.

Table 1
Comparison between SREDM and labeling algorithm.

N_T	PB	Labeling algorithm			SREDM		
		Solution	Gap (%)	CPU (s)	Solution	Gap (%)	CPU (s)
20	−70 184	−54 919	21.75	3.7	−59 951	14.58	3.1
40	−148 887	−113 154	24.00	9.2	−124 470	16.40	6.1
60	−217 507	−169 112	22.25	31.7	−180 966	16.80	8.2
80	−252 516	−205 800	18.50	32.9	−212 618	15.80	8.2
100	−370 599	−293 699	20.75	49.8	−317 233	14.40	18.1
200	−712 004	−582 063	18.25	1491.2	−612 323	14.00	33.4
400	−133 8236	*	*	*	−1 118 765	16.40	68.3
1000	−1 846 001	*	*	*	−1 524 797	17.40	214.1

Table 2
Comparison between the SREDM and the modified SCAM.

N_L	N_R	Net contribution				Gap to PB		
		PB	Myopic	SCAM	SREDM	Myopic (%)	SCAM (%)	SREDM (%)
10	50	−1 432 137	−471 780	−899 382	−928 024	67.06	37.20	35.20
15	75	−2 235 212	−1 188 201	−1 439 477	−1 546 767	46.84	35.60	30.80
20	100	−3 647 817	−1 247 668	−2 320 012	−2 451 333	65.80	36.40	32.80
30	150	−6 584 076	−2 391 967	−4 477 172	−4 319 154	63.67	32.00	31.40
35	175	−7 906 730	−2 634 798	−5 186 815	−5 781 743	66.68	34.40	26.88
40	200	−9 641 711	−3 373 945	−6 479 229	−6 845 614	65.36	32.80	29.00

5. Conclusion

In this paper, we propose the first model and algorithm for the dynamic fleet management problem with uncertain demand and customer chosen service levels. Then, we develop the SREDM algorithm for the resulting stochastic networks, which structurally decomposes the network into trees, whose expected recourse functions are obtained by a pseudo-polynomial algorithm in Shi et al. (2007). Numerical experiments show that the use of the SREDM methodology is very encouraging. In future, it is interesting to investigate the pricing issues for difference service levels and the impacts of other customer behaviors.

Acknowledgment

This research is supported by the Natural Science Foundation of China via funding nos. 71171204, 71171205, and the Fundamental Research Funds for the Central Universities via no. 1209073.

Appendix A. Local search procedure

In this section, we regard the supply S as given, thus we suppress it from the functions by writing $\hat{Q}(S,\mathbf{b})$ as $\hat{Q}(\mathbf{b})$, $\hat{Q}_i(S,\mathbf{b})$ as $\hat{Q}_i(\mathbf{b})$.

Let us associate an indicator vector ${e_{ij}}^k$ with each variable ${b_{ij}}^k$. In the vector ${e_{ij}}^k$, only the element representing ${b_{ij}}^k$ is equal to 1 and all the others are 0. We define the set of its neighbors as

$$N(\mathbf{b}) = \{\mathbf{b}+e^k_{ij}-e^k_{uj} : b^k_{uj}=0, b^k_{ij}=1\}.$$

To obtain a neighbor of $\mathbf{b}$, we can remove one arc that is previously allocated to tree i (i.e., the arc is in tree i), to another tree. From each tree, let us define the arcs that can be re-allocated as *supply arcs* of this tree. Mathematically, it is defined as

$$G_i = \{(j,k) : b^k_{ij}=1 \quad \forall j \in \mathcal{L}\}.$$

The *supply arcs* from the same branch node j of tree i will be grouped in $G_{i,j}$

$$G_{i,j} = \{(j,k) : b^k_{ij}=1\}.$$

Now, we are ready to present the local search procedure. Let ε be a small solution tolerance.

Local Search Procedure

*Step*1: Set $n=0$ and $\mathbf{b}_n$.
*Step*2: Update $n=n+1$,

$$\mathbf{b}_n = \arg\min_{\mathbf{b}_n \in N(\mathbf{b}_{n-1})} \{\hat{Q}(\mathbf{b})\}.$$

*Step*3: If $\hat{Q}(\mathbf{b}_n)-\hat{Q}(\mathbf{b}_{n-1}) \le \varepsilon$, stop. Otherwise, go to Step 2.

Let $\mathbf{b}+e^k_{uj}-e^k_{ij}$ be one neighbor of $\mathbf{b}$. It can be obtained by removing the *supply arc* (j, k) in $G_{i,j}$ to the tree u that shares the arc (i,j) with tree i (let us call such a movement *arc-switch*). We define $\hat{Q}_u(\mathbf{b}+e^k_{uj})$ as the objective value of tree u after adding one arc (j,k); $\hat{Q}_i(\mathbf{b}-e^k_{ij})$ as the objective value of tree i after removing the arc (j,k) from the tree; $\delta^{k+}_{uj}(\mathbf{b})$ as the difference between $\hat{Q}_u(\mathbf{b}+e^k_{uj})$ and $\hat{Q}_u(\mathbf{b})$; and $\delta^{k-}_{ij}(\mathbf{b})$ the difference between $\hat{Q}_i(\mathbf{b}-e^k_{ij})$ and $\hat{Q}_i(\mathbf{b})$. In this following, we regard $\mathbf{b}$ as given, thus we suppress it from the functions by writing $\delta^{k+}_{uj}(\mathbf{b})$ as δ^{k+}_{uj}, $\delta^{k-}_{ij}(\mathbf{b})$ as δ^{k-}_{ij}. Step 2 of the local search procedure can be implemented by the following procedure.

The efficiency of this step depends on two factors: (1) The size of the neighborhood and (2) the efficiency of performing each switch. The size of the neighborhood depends on how many arcs are shared by different trees. Let $|\mathcal{L}|$ be the number of trees, and K be the maximum number of *supply arcs* in one tree. In the worst case, there will be $K|\mathcal{L}|^2$ switches.

To facilitate the search process, we develop an *Arc-Switch Procedure* that can search the local optimal solution in less than $K|\mathcal{L}|$ steps. This procedure will be described later. We first introduce some propositions of the functions $\hat{Q}(\mathbf{b}+e^k_{uj}-e^k_{ij})$, $\hat{Q}_u(\mathbf{b},+e^k_{uj})$, and $\hat{Q}_i(\mathbf{b},-e^k_{ij})$. With these propositions, we can design an *Arc-Switch Procedure* which significantly reduces the computational effort of Step 2 in the local search procedure.

Proposition 1. *The objective function of $\hat{Q}(\mathbf{b}+e^k_{uj}-e^k_{ij})$ can be calculated by*

$$\hat{Q}(\mathbf{b}+e^k_{uj}-e^k_{ij})-\hat{Q}(\mathbf{b}) = \delta^{k-}_{ij}+\delta^{k+}_{uj}.$$

This proposition reflects the fact that each arc-switch involves only two trees and the benefit of an arc-switch is the summation of the "gain" of adding one arc and the "loss" of removing that arc. With this proposition, to evaluate all $K|\mathcal{L}|^2$ arc switches, we only need to compute all $K|\mathcal{L}|$ "gains" and "losses".

To further reduce the computational burden, we develop both lower and upper bound for the "gains" and "losses" such that some arc switches can be excluded from the candidates of the "best" without calculating their benefits.

Suppose the *supply arcs* in each $G_{i,j}$ are ranked in ascending costs and all the arcs' costs are non-positive since they measure the future marginal "profits". We consider two arcs (j,k) and (j,k'), in $G_{i,j}$, and $k<k'$.

Proposition 2. *The values of δ^{k+}_{uj} and δ^{k-}_{ij} satisfy, $\delta^{k'+}_{uj} \le \delta^{k+}_{uj}$, $\delta^{k'-}_{ij} \le \delta^{k-}_{ij}$.*

This is a direct result of the convexity of the objective function. It is intuitive that adding the arc with higher profit results in more "gain" and removing the arc with higher profit results in more "loss".

Note that the only difference from adding the arc (j,k') and the arc (j,k) is the difference in their costs. Thus, adding the arc (j,k') to tree u is equivalent to adding the arc (j,k) to tree u first and then change the arc's cost from ${q_j}^k$ to $q^{k'}_j$. It leads to Proposition 3.

Proposition 3. *The difference between $\delta^{k'+}_{uj}$ and δ^{k+}_{uj} is bounded by*

$$\delta^{k'+}_{uj}-\delta^{k+}_{uj} \le f(\mathbf{b},+e^k_{uj})(q^{k'}_j-q^k_j),$$

where the $f(\mathbf{b},+e^k_{uj})$ represents the expected flows in the arc (j,k) when it is added to the tree u. The computation of $f(\mathbf{b},+e^k_{uj})$ and the proof of this proposition is stated as follows.

Proof. According to the definition, we have,

$$\Delta\tilde{Q}_u(\mathbf{b},+e^{k'}_{uj},\omega)-\Delta\tilde{Q}_u(\mathbf{b},+e^k_{uj},\omega)$$
$$=\tilde{Q}_u(\mathbf{b}+e^{k'}_{uj},\xi(\omega))-\tilde{Q}_u(\mathbf{b}+e^k_{uj},\omega).$$

According to (2.38), we know that $\tilde{Q}_u(\mathbf{b}+e^{k'}_{uj},\omega)$ can be rewritten as

$$\tilde{Q}_u(\mathbf{b}+e^{k'}_{uj},\omega) = \sum_{n=1}^{N_p} c_n f_n(s,\omega).$$

Let $f(j,v,\omega)$ denote the flow through the second-level arc (j, v) in the tree u after the arc (j,k') is added. Since each second-level arc corresponds to one path, the function $\tilde{Q}_u(\mathbf{b}+e^{k'}_{uj},\omega)$ can be rewritten as

$$\tilde{Q}_u(\mathbf{b}+e^{k'}_{uj},\omega) = \sum_{(j,v)\in G_u} q^v_j f(j,v,\omega),$$

where G_u is the set of second-level arcs in tree u after the arc (j,k') is added.

Accordingly, let $\hat{f}(j,v,\omega)$ denote the flow through the second-level arc (j_v) in the tree u after the arc (j,k) is added. The function $\tilde{Q}_u(\mathbf{b}+e^k_{uj},\omega)$ can be rewritten as

$$\tilde{Q}_u(\mathbf{b}+e^k_{uj},\omega)=\sum_{(j,v)\in G_u} q^v_j \hat{f}(j,v,\omega)$$

where G'_u is the set of second-level arcs in tree u after the arc (j,k) is added.

Note that the only difference between these two functions is that the arc (j,k) may have a different cost compared to the arc (j,k'). Thus the optimal flows for the function $\tilde{Q}_u(\mathbf{b}+e^k_{uj},\omega)$ will be feasible for the function $\tilde{Q}_u(\mathbf{b}+e^{k'}_{uj},\omega)$. Therefore, we have,

$$\tilde{Q}_u(\mathbf{b}+e^{k'}_{uj},\omega)\le \sum_{(j,v)\in G_u} q^v_j \hat{f}(j,v,\omega),$$

and,

$$\begin{aligned}\tilde{Q}_u(\mathbf{b}+e^{k'}_{uj},\omega)-\tilde{Q}_u(\mathbf{b}+e^{k}_{uj},\omega)&=\sum_{(j,v)\in G'_u} f(j,v,\omega)q^v_j-\sum_{(j,v)\in G_u}\hat{f}(j,v,\omega)q^v_j\\
&\le \sum_{(j,v)\in G'_u, v\ne k'}\hat{f}(j,v,\omega)q^v_j-\sum_{(j,v)\in G_u,\ v\ne k}\hat{f}(j,v,\omega)q^v_j\\
&\quad+\hat{f}(j,k,\omega)q^{k'}_j-\hat{f}(j,k,\omega)q^k_j\\
&=\hat{f}(j,k,\omega)(q^{k'}_j-q^k_j).\end{aligned}$$

By taking the expectation on all samples, we have

$$\begin{aligned}\Delta\hat{Q}_u(\mathbf{b},+e^{k'}_{uj})-\Delta\hat{Q}_u(\mathbf{b},+e^{k}_{uj})&=E_{\omega\in\Omega}(Q_u(\mathbf{b}+e^{k'}_{uj},\omega)-Q_u(\mathbf{b}+e^{k}_{uj},\omega))\\
&\le E_{\omega\in\Omega}(\hat{f}(j,k,\omega)(q^{k'}_j-q^k_j))\\
&=f(\mathbf{b},e^k_{uj})(q^{k'}_j-q^k_j).\quad\square\end{aligned}$$

Since the arc (j,k) has lower cost than the arc (j,k'), we have the following proposition:

Proposition 4. *The expected flow in the arc (j,k) is monotone decreasing with the cost of the arc (j,k). That is,*

$$f(\mathbf{b},+e^{k'}_{uj})\le f(\mathbf{b},+e^{k}_{uj}),\quad k<k'.$$

Similarly, adding the arc (j,k) to the tree u is equivalent to adding the arc (j,k') to the tree u first and then change its cost from $q^{k'}_j$ to $q_j{}^k$. This results in the following proposition:

Proposition 5. *The difference between δ^{k+}_{uj} and $\delta^{k'+}_{uj}$ is bounded from below.*

$$\delta^{k+}_{uj}-\delta^{k'+}_{uj}\le f(\mathbf{b},+e^{k'}_{uj})(q^k_j-q^{k'}_j).\tag{40}$$

As a summary of the above propositions, we have that,

Proposition 6. *The difference between δ^{k+}_{uj} and $\delta^{k'+}_{uj}$ is bounded by*

$$f(\mathbf{b},+e^{k'}_{uj})(q^{k'}_j-q^k_j)\le\delta^{k'+}_{uj}-\delta^{k+}_{uj}\le f(\mathbf{b},+e^{k}_{uj})(q^{k'}_j-q^k_j).$$

This proposition shows that the difference in the “gains” by adding two different arcs of the same $G_{i,j}$ to tree u is actually bounded both from above and from below.

Similarly, we can show that the difference of the “losses” by removing two different arcs in the same $G_{i,j}$ is also bounded both from above and from below. Let $f(\mathbf{b},-e^k_{ij})$ be the expected flow of the arc (j,k) in tree i before the arc is removed.

Proposition 7. *The difference between two “losses” is bounded by*

$$-f(\mathbf{b},-e^k_{ij})(q^{k'}_j-q^k_j)\le\delta^{k'-}_{uj}-\delta^{k-}_{uj}\le -f(\mathbf{b},-e^{k'}_{ij})(q^{k'}_j-q^k_j).$$

These bounds can help us exclude the switches that cannot achieve enough cost improvement, and we can perform a recursive procedure to search the best switch. For each tree i and each set of *supply arcs* in $G_{i,j}$, let $T(i,j)$ be the set of trees that share the arcs from node j with tree i. Let I be the index of *supply arcs* that have not been tested so far. For any tree $u\in T(i,j)$, we perform the following recursive procedure.

Arc-Switch Procedure (i,u,I)

*Step*1: *Initialization*
Let $s=1$, $\ell=|G_{i,j}|$. Let Δ_{min} be the best cost improvement achieved so far.

*Step*2: *Evaluate the arc-switch for the first and the last arcs*
For each set of *supply arcs* in $G_{i,j}$, obtain the first arc (j,s) and the last arc (j,ℓ). Compute δ^{s-}_{ij}, δ^{s+}_{uj}, $\delta^{\ell-}_{ij}$, $\delta^{\ell+}_{uj}$, $f(\mathbf{b},-e^s_{ij})$ and $f(\mathbf{b},-e^\ell_{ij})$, $f(\mathbf{b},+e^s_{uj})$ and $f(\mathbf{b},+e^\ell_{uj})$. Let $\Delta_{min}=\min\{\delta^{\ell+}_{uj}+\delta^{\ell-}_{uj},\delta^{s+}_{uj}+\delta^{s-}_{uj}\}$

*Step*3: *Evaluate the arc-switch for the supply arc k*
Let $k=[(s+\ell)/2]$, compute δ^{k+}_{uj}. If $f(\mathbf{b},+e^k_{uj})-f(\mathbf{b},-e^s_{ij})\ge 0$, remove the indices in the $[s,k]$ from I. If $f(\mathbf{b},+e^k_{uj})-f(\mathbf{b},-e^\ell_{ij})\ge 0$, remove the indices in the $[k,\ell]$ from I. Otherwise, go to Step 4.

*Step*4: *Compare the bounds*
Let $LB_1=(f(\mathbf{b},+e^k_{uj})-f(\mathbf{b},-e^s_{ij}))(q^k_j-q^s_j)$ and $LB_2=(f(\mathbf{b},+e^k_{uj})-f(\mathbf{b},-e^\ell_{ij}))(q^k_j-q^\ell_j)$. If $LB_1+\hat{Q}(\mathbf{b},+e^s_{uj}-e^s_{ij})-\hat{Q}(\mathbf{b})>\Delta_{min}$, remove the indices in $[s,k]$ from I. Otherwise, compute δ^{k-}_{ij}, δ^{k+}_{ij}, and set $\Delta_{min}=\min\{\delta^{k-}_{ij}+\delta^{k+}_{ij},\Delta_{min}\}$, set $I'=[s,k]$, and perform **Arc Switch Procedure** $(i,\ u,\ I')$. If $LB_2+\hat{Q}(\mathbf{b},+e^\ell_{uj}-e^\ell_{ij})-\hat{Q}(\mathbf{b})>\Delta_{min}$, remove the indices in $[k,\ell]$ from I. Otherwise, set $I''=[k,\ell]$, and perform **Arc Switch Procedure** $(i,\ u,\ I'')$.

Explanation: In Step 3, if $f(\mathbf{b},+e^k_{uj})-f(\mathbf{b},-e^s_{ij})\ge 0$, according to Proposition 1, $\hat{Q}(\mathbf{b},+e^k_{ij}-e^k_{uj})-\hat{Q}(\mathbf{b},+e^s_{ij}-e^s_{uj})\ge 0$. Thus the switch (i,u,s) is better than the switch (i,u,k). According to Proposition 4, we know that for any $k'<k$, $f(\mathbf{b},+e^{k'}_{uj})-f(\mathbf{b},-e^s_{ij})\ge 0$, that implies that the switch (i,u,s) is the best one for all switches (i,u,k'), $k'\in[s,k]$, and the switches (i,u,k'), $k'\in[s,k-1]$, can be excluded.

Similarly, by examining whether $f(\mathbf{b},+e^k_{uj})-f(\mathbf{b},-e^\ell_{ij})\ge 0$, we can judge whether the switch (i,u,ℓ) is the best switch in all switches (i,u,k'), $k'\in[k,\ell]$.

According to Proposition 4, LB_1 and LB_2 are two lower bounds of the function $\hat{Q}(\mathbf{b},+e^k_{uj}-e^k_{ij})-\hat{Q}(\mathbf{b})$. By comparing with $LB1$ with Δ_{min}, we can judge whether the switch (i,u,k) is the best one among all (i,u,k'), $k'\in[s,k]$. Similarly, by comparing with LB_2 with Δ_{min}, we can judge whether the (i,u,k) is the best one among all (i,u,k'), $k'\in[k,\ell]$.

The critical step of the arc-switch procedure is to solve the subproblems $\hat{Q}_u(\mathbf{b})$, $\hat{Q}_i(\mathbf{b})$ and to compute the cost changes $\hat{Q}_u(\mathbf{b},+e^k_{uj})$ and $\hat{Q}_i(\mathbf{b},-e^k_{ij})$. To solve the subproblems, one can apply the tree algorithm in Shi et al. (2007). This algorithm can compute the exact optimal objective value of the subproblems $\hat{Q}_u(\mathbf{b})$ efficiently. The essential of this algorithm is to compute a number of *dispatch probabilities*. A summary of this algorithm can be found in Appendix B.

Appendix B. Propositions and algorithm for the modified TRP

In this appendix, we first provide a summary of the main results in Shi et al. (2007) and then describe the algorithm for the modified TRP.

Given a two-level tree with random arc capacities and deterministic arc costs, our objective is to calculate the expected cost function in terms of the supply s at the root node. Let $(\Omega, P, \mathcal{F})$ be a probability space and $\omega \in \Omega$ an outcome. For outcome ω, the vector of realized arc capacities is $\xi(\omega)$. Suppose that the paths, each consisting of two arcs (from i_t to a branch node and then to a leaf node), are ranked in an ascending order according to their costs. We call the nth ranked path as path n and define

N_p	the number of paths in the tree
c_n	the cost of path n
$f_n(s, \xi(\omega))$	the flow on path n when the supply is s and the realized set of arc capacities is $\xi(\omega)$.

Then, the problem can be stated as determining

$$\overline{Q}(s) = E_{\omega \in \Omega} Q(s, \xi(\omega)),$$

where $Q(s, \xi(\omega))$ is defined as

$$Q(s, \xi(\omega)) = \min \sum_{n=1}^{N_p} c_n f_n(s, \xi(\omega)), \tag{41}$$

To obtain $\overline{Q}(s) = E_{\omega \in \Omega} Q(s, \xi(\omega))$, let us begin by defining the *dispatch probabilities* and the expected path flow as

$\phi(k,n)$	the probability that the kth unit of flow entering node $i(t)$ takes path n
$\overline{f}_n(s)$	the expected flow on path n when the supply to the root node is s.

From the definitions, we can see that

$$\overline{f}_n(s) = \sum_{k=1}^{s} \phi(k,n) \tag{42}$$

and the total expected cost for a given s is

$$\overline{Q}_i(s) = E_{\omega \in \Omega} Q_i(s, \xi(\omega)) = \sum_{n=1}^{N_p} c_n \overline{f}_n(s) = \sum_{n=1}^{N_p} \sum_{k=1}^{s} c_n \phi(k,n). \tag{43}$$

Thus, $\overline{Q}(s)$ can be obtained by calculating $\phi(k,n)$ for $k = 1, 2 \ldots, s$ and $n = 1, \ldots, N_p$. In the following sections, we provide an algorithm to calculate $\phi(k,n)$ for $n = 1, \ldots, N_p - 1$. For $n = N_p$,

$$\phi(k, N_p) = 1 - \sum_{n=1}^{N_p - 1} \phi(k,n). \tag{44}$$

Let us define

$\sigma^n_{jt'}$	the maximum possible outgoing flow from branch node $j_{t'}$.
$b^n_{jt'}$	the number of paths covering the branch node $j_{t'}$ in $\mathcal{T}_n$.
χ^n_j	the maximum possible outgoing flow from the set of branch nodes in $N_j[i,t]$.

The amount of $\sigma^n_{jt'}$ is limited by the maximum amount of inbound flow to node $j_{t'}$. Furthermore, this amount cannot exceed the number of paths passing through $j_{t'}$ since we can have at most one unit of flow per path. Therefore, $\sigma^n_{jt'}$ can be written as

$$\sigma^n_{jt'} = \min\{b^n_{jt'}, \xi_{ijtt'}\}. \tag{45}$$

On the other hand, the random variable $\chi_j{}^n$ is the sum of $\sigma^n_{jt'}$, i.e.,

$$\chi^n_j = \sum_{j(t') \in N_j[i,t]} \sigma^n_{jt'}. \tag{46}$$

Due to the dependency of $\xi_{ijtt'}$, finding the distributions of $\chi_j{}^n$ is not trivial. However, if they are found, then finding the distribution of $\phi(k,n)$ is straightforward.

Let us call $\chi_j{}^n$ the *bundle capacity* as it measures the maximum possible flow through a bundle of nodes representing a particular location j in the graph. Their distributions, as shown later, determine the values of $\phi(k,n)$. To establish their relationship, we assume that the ranked path n covers arc $(i_t, j_{t'})$ and consider the kth unit of flow entering the root node i_t.

Theorem 1. *Suppose that path n covers arc $(i_t, j_{t'})$, the dispatch probability $\phi(k,n)$ can be calculated as*

$$\phi(k,n) = \Pr(\sigma^n_{jt'} = b^n_{jt'}) \Pr\left(\sum_{w \in \mathcal{L},\ w \neq j} \chi^n_w = k - b^n_{jt'} \right). \tag{47}$$

Proof. See Theorem 1 in Shi et al. (2007). □

Theorem 2. *Suppose that path n covers arc $(i_t, j_{t'}) \in A_j[i,t]$. Then*

(a) *for any $j_{\hat{t'}} \neq j_{t'}$ and $(i_t, j_{\hat{t'}}) \in A_j[i,t]$, $\sigma^n_{j\hat{t'}}$ has the same distribution as $\sigma^{n-1}_{j\hat{t'}}$,*
(b) *for any $w \neq j$, $\chi_w{}^n$ has the same distribution as χ^{n-1}_w,*
(c) *for $w{=}j$, and for $y = 0, 1, \ldots, s$, we have*

$$\Pr(\chi^n_w = y) = \begin{cases} \Pr(\chi^{n-1}_w = y) - \Pr(\xi_{ijt'} > b^{n-1}_{jt'}) & \text{if } y = b^{n-1}_{jt'}, \\ \Pr(\chi^{n-1}_w = y) + \Pr(\xi_{ijt'} > b^{n-1}_{jt'}) & \text{if } y = b^{n}_{jt'}, \\ \Pr(\chi^{n-1}_w = y) & \text{otherwise}. \end{cases}$$

Proof. See Theorem 2 in Shi et al. (2007). □

The recursive relationship provided in Theorem 2 allows us to compute the distribution of $\chi_j{}^n$ without explicitly knowing the distributions of $\sigma^n_{jt'}$, thereby reducing the computational effort. Using Theorems 1 and 2, we can compute the expected cost function for the modified TRP as follows.

```
Step1: Initialization
    Set $n = 0, k = 0$, $\phi(0,0) = 0$ and $\Pr(\chi^0_j = 0) = 1$ for $i = 1, \ldots, M$.
Step2: Obtain the distributions for the bundle capacities
    Set $n = n + 1$. Obtain the distribution of $\chi_j{}^n$ by Theorem 2 for all $j \in \mathcal{L}$.
Step3: Compute the dispatch probabilities
    If $n < N_p$, calculate $\phi(k,n)$ according to (47) of Theorem 1, for all $k = 1, \ldots, s$. If $n = N_p$, calculate $\phi(k,n)$ according to (44).
Step4: Test for termination
    If $n < N_p$, go to Step 2; otherwise, go to Step 5.
Step5: Compute the expected total cost function
    Calculate $\overline{Q}(s)$ by (43) and then terminate.
```

The algorithm is actually a pseudo-polynomial time algorithm. The computational effort depends on several parameters. The first is the amount of supplies at the root node which bounds the possible realizations of $\chi_j{}^n$ in $[0, s]$ and limits the breadth of the third-level arcs (for each branch node, we only need s outward second-level arcs). The second is the total number of locations, M. The third is the number of possible realizations of the travel time, V. The second and third parameters together determine the size of the tree. All three parameters together determine the total number of paths, that is, $N_p = MVs$. Thus, the total number of steps to obtain the distributions of all χ_i^n for $n = 1, 2, \ldots, MVs$ is bounded by $O(MVs^2)$. In the future research, we expect that this approach can be used to study more customer behaviors (besides customer chosen service level) in the context of fleet management.

References

Barbarosoglu, G., Arda, Y., 2004. A two-stage stochastic programming framework for transportation planning in disaster response. Journal of the Operational Research Society 55.

Birge, J.R., Louveaux, F., 1997. Introduction to Stochastic Programming. Springer.

Birge, J.R., Wallace, S.W., 1988. A separable piecewise linear upper bound for stochastic linear programms. SIAM Journal on Control and Optimization 26, 725–739.

Cheung, R.K., Hang, D.D., Shi, N., 2005. A labeling method for dynamic driver-task assignment with uncertain task durations. Operations Research Letters 33, 411–420.

Cheung, R.K., Powell, W.B., 1996. An algorithm for multistage dynamic networks, with random arc capacities, with an application to dynamic fleet management. Operations Research 44, 951–963.

Chu, Q., 1995. Dynamic and Stochastic Models for Container Allocation (Ph.D. thesis), Massachusetts Institute of Technology, USA.

Crainic, T.G., Gendreau, M., Dejax, P., 1993. Dynamic and stochastic models for the allocation of empty containers. Operations Research 41, 102–126.

Crainic, T.G., Laporte, G., 1998. Fleet Management and Logistics (Center for Research on Transportation 25th Anniversary Series, 1971–1996), 1st ed., Springer.

Dantzig, G.B., Fulkerson, D.R., 1954. Minimizing the number of tankers to meet a fixed schedule. Naval Research Logistics Quarterly 1, 217–222.

Dejax, J., Crainic, T.G., 1987. Survey paper—a review of empty flows and fleet management models in freight transportation. Transportation Science 21, 227–248.

Federgruen, A., Groenevelt, H., 1986. The greedy procedure for resource allocation problems: necessary and sufficient conditions for optimality. Operations Research 34, 909–918.

Flatberg, T., Hasle, G., Kloster, O., Nilssen, E., Riise, A., 2007. Dynamic and stochastic vehicle routing in practice, pp. 41–63.

Frantzeskakis, L.F., Powell, W.B., 1990. A successive linear approximation procedure for stochastic, dynamic vehicle allocation problems. Transportation Science 24, 40–57.

Glockner, G.D., Nemhauser, G.L., 2000. A dynamic network flow problem with uncertain arc capacities: formulation and problem structure. Operations Research 48, 233–242.

Godfrey, G.A., Powell, W.B., 2002. An adaptive dynamic programming algorithm for dynamic fleet management, i: single period travel times. Transportation Science 36, 21–39.

Hochbaum, D.S., Shanthikumar, J.G., 1990. Convex separable optimization is not much harder than linear optimization. Journal of the ACM 37, 843–862.

Jordan, W.C., Turnquist, M.A., 1983. A stochastic dynamic network model for railroad car distribution. Transportation Science 17, 123–145.

Kall, P., Mayer, J., 2005. Stochastic Linear Programming: Models, Theory, and Computation, International Series in Operations Research & Management Science, vol. 80, Springer.

Murota, K., 1998. Discrete convex analysis. Mathematical Programming 83, 313–371.

Powell, W., Sheffi, Y., Nickerson, K., Butterbaugh, K., Atherton, S., 1988. Maximizing profits for North American van lines' truckload division: a new framework for pricing and operations. Interfaces 18, 21–41.

Powell, W.B., 1986. A stochastic model of the dynamic vehicle allocation problem. Transportation Science 20, 117–129.

Powell, W.B., 1987. An operational planning model for the dynamic vehicle allocation problem with uncertain demands. Transportation Research 21, 217–232.

Powell, W.B., 1988. A comparative review of alternative algorithms for the dynamic vehicle allocation problem. In: Golden, B., Assad, A. (Eds.), Vehicle Routing: Methods and Studies, vol. 62. North Holland, pp. 249–291.

Powell, W.B., Topaloglu, H., 2005. Fleet management. In: Applications of Stochastic Programming, MPS/SIAM Series on Optimization, vol. 5, SIAM, Philadelphia, PA, pp. 185–215.

Shi, N., Cheung, R.K., Song, H.Q., 2007. On stochastic programs over trees with partially dependent arc capacities. Networks 50, 157–163.

Simao, H.P., Day, J., George, A., Gifford, T., Nienow, J., Powell, W.B., 2009. An approximate dynamic programming algorithm for large-scale fleet management: a case application. Transportation Science 43, 178–197.

Topaloglu, H., Powell, W.B., 2006. Dynamic-programming approximations for stochastic time-staged integer multicommodity-flow problems. INFORMS Journal of on Computing 18, 31–42.

Wallace, S., 1987. A piecewise linear upper bound on the network recourse function. Mathematical Programming 38, 133–146.

Wallace, S.W., 1986. Solving stochastic programs with network recourse. Networks 16, 225–317.

White, W.W., 1972. Dynamic transshipment networks: an algorithm and its application to the distribution of empty containers. Networks 2, 211–236.

Zhang, D., Adelman, D., 2009. An approximate dynamic programming approach to network revenue management with customer choice. Transportation Science 43, 381–394.