IIE Transactions (2003) **35**, 433–444
0740-817X © 2003 "IIE"
0740-817X/03 $12.00 + .00
DOI: 10.1080/07408170390187889

A time-window sliding procedure for driver-task assignment with random service times

RAYMOND K. CHEUNG[1] and DARREN D. HANG[2]

[1] *Department of Industrial Engineering and Engineering Management, Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong*
E-mail: rcheung@ust.hk
[2] *Carmen Systems, Odinsgatan 9, 41103 Gothenburg, Sweden*
E-mail: darren@carmen.se

Received November 2000 and accepted April 2002

This paper addresses the problem of assigning drivers to cover tasks when the tasks are the movements of containers. The tasks must be started within certain time windows and their servicing times are uncertain while the decisions are made over time. These decisions can be changed if new information is received. The problem is formulated in a stochastic optimization framework with the objective of minimizing the costs of current driver-task assignment and the expected future costs. A time-window sliding solution procedure is developed to estimate the expected future costs by solving the minimum cost flow problems iteratively. Furthermore, the results of the numerical experiments that assess the efficiency of the algorithm and the benefits of considering uncertainty in the model are reported.

1. Introduction

This paper addresses the problem of assigning drivers to cover a set of tasks that represent short-haul in-land container movements. The research is motivated by the *drayage problem* in Hong Kong, a city that handled over 16 million 20-foot equivalent units of containers in 1999. In the drayage problem, a task can be a *container yard* move that involves moving a loaded container out of the container yard at a terminal to a consignee's warehouse, waiting for the unloading of goods from the container, and taking the empty container to a container storage site. This type of task involves a movement and some intermediary activities. Another task can be *shuttling* that represents a movement of a container between two locations (such as the container terminal and the rail yard) without involving any intermediary activity. There are several characteristics of a drayage problem. First, tasks have to be started within some time windows. Because of the very limited parking space in Hong Kong, there is a waiting cost for a driver who arrives at a task's starting location before the task can be started. This cost is location dependent. Second, task service times can be uncertain and have various degrees of uncertainty. For example, the container yard move involves many intermediary activities and the service time involves a large time variation whereas the shuttling is relatively simple and its service time has a small variation. Third, not all the tasks have to be covered by a company's own fleet because uncovered tasks can be outsourced to a third party carrier through payment of a premium. Finally, the drivers' routes are not fixed and the decisions on assigning drivers to tasks are made over time.

In this paper, the problem is formulated in a framework of stochastic dynamic optimization. The cost of assigning a driver to a task is the sum of the transportation cost from the finishing location of a task to the starting location of another task, the waiting cost before starting the task, minus the reward obtained by covering the task. The objective is to minimize the driver-task assignment costs for the currently available drivers and the expected future driver-task assignment costs. In this study, the planning horizon is divided into evenly distributed time intervals (for example, 30 minutes). Time parameters and variables are multiples of these intervals.

The literature related to this research can be grouped into three areas. The first area is the literature on deterministic task scheduling with and without a time window. Fischetti *et al.* (1987, 1989) consider the bus driver scheduling problem and formulate the problem in the framework of machine scheduling. They show that the problem is NP-hard and develop lower bounds and dominance criteria that are embedded in a branch-and-bound process for obtaining the optimal solution. Desrosiers *et al.* (1984) study the routing and scheduling problem with time window constraints and propose a column

generation method. In their method, columns (of the constraint matrix) represent vehicle paths that can be obtained by solving time-constrained shortest path problems. Desrosiers *et al.* (1995) present a unified solution framework for various deterministic routing and scheduling problems that is based on path-flow formulations and the Dantzig-Wolfe decomposition. Path-flow formulations allow us to incorporate complex work rules. In the context of driver-task assignment, Powell *et al.* (2000) develop an adaptive labelling algorithm in which a set of labels are used to represent the possible histories of drivers and another set of labels are used to represent some possible tours of the drivers in the future. The matching of the two sets of labels can produce new tours with lower costs. The algorithm, however, does not consider the uncertainty of service times. In practice, the uncertainty can make a previously determined tour infeasible.

The second related research area is the research on Stochastic Vehicle Routing Problems (SVRP) as our problem can be considered as a special type of SVRP in which vehicles have unit capacity and the service times are random. In SVRP, the stochastic component can be the travel time (Laporte *et al.*, 1992), the customer demand (Bertsimas, 1992), or the existence of customers (Gendreau *et al.*, 1995; 1996a). If our problem is formulated as SVRP, then we need to consider the service time-window constraint, permit tasks to be rejected, penalize early arrivals at customer sites, and allow routes to be formed dynamically. To our knowledge, such a problem has not been studied explicitly in the literature on SVRP. The problem can also be considered as a stochastic program with recourse (see Birge and Louveaux (1997) for a review). Solution methods for general stochastic programs with recourse include scenario-based methods (Rockafellar and Wets, 1991) such as the integer L-shaped method (Laporte and Louveaux, 1993), sampling-based methods (Ermoliev, 1988) such as the Bayesian approach (Jagannathan, 1985) for two-stage stochastic programs, dynamic programming (Secomandi, 1998), and tabu search methods such as the one for stochastic vehicle routing (Gendreau *et al.*, 1996b). Similar to Jagannathan (1985), the method proposed in this paper involves sampling. However, unlike Jagannathan (1985), this paper deals with a multistage stochastic program in which even the start time of a stage is a random quantity. Furthermore, the probability mass functions used for generating samples are not given in advance but are iteratively estimated throughout the solution process.

The third related area is the research on dynamic vehicle allocation problems (Frantzeskakis and Powell, 1990) where a number of vehicles need to cover a set of loads (or tasks, in our terminology) over time. In the problem, the availability of a load is a discrete random variable. The start times of the loads are fixed. Cheung and Powell (1996) formulate the problem in a dynamic network with random arc capacities and develop a decomposition method to solve the problem. Different from the dynamic vehicle allocation problem, our problem has random task service times and tasks can be started within some time windows.

In the motivating application, immediate solutions are required. Therefore, our focus is on approximation techniques. Our contributions lie in formulating the problem in a stochastic dynamic optimization framework and in developing a new solution procedure that assigns drivers to tasks at decision time points over the planning horizon. This procedure depends on the ability to estimate the expected costs of covering tasks in the future. The concept of assignment time windows is presented that mimics the actual practices in making driver-task assignments. By sliding such a time-window backward from the end of the planning horizon to the current time, the costs of starting tasks at different time points are estimated. These estimates are then used for making the dynamic driver-task assignments.

The paper is organized as follows. Section 1 provides the problem formulation. Section 2 presents the time-window sliding solution procedure. Section 3 describes the steps of cost estimation for a given time-window. Section 4 reports some numerical experiments and Section 5 provides concluding remarks.

2. Formulation

To simplify the discussion, first consider the example of a deterministic driver-task assignment problem with two drivers and three tasks, as shown in Fig. 1. The available time and the remaining duty time for each driver are given. For each task, the starting time-window, the revenue generated by this task, and the service duration are shown. Solid arcs are used to represent the deadhead movements between the current locations of the drivers and the starting locations of the tasks to be covered. Dashed arcs are used to represent the returns of drivers to the depot. The numbers next to the solid arcs are the transportation costs and the travel times. If the objective is to minimize the cost or to maximize the revenue, then the optimal solution is to let driver 1 take tasks 1, 2, and 3 in sequence, finishing at 16:30 and to let driver 2 go back to the depot directly.

The mathematical formulations of the deterministic model and the stochastic dynamic model are described next.

2.1. *The deterministic model*

Assume that there are N drivers to cover I tasks over a planning horizon of T periods. Define

$\mathcal{N} = \{1, \ldots, N\}$ = the set of drivers;
$\mathcal{I} = \{1, \ldots, I\}$ = the set of tasks.

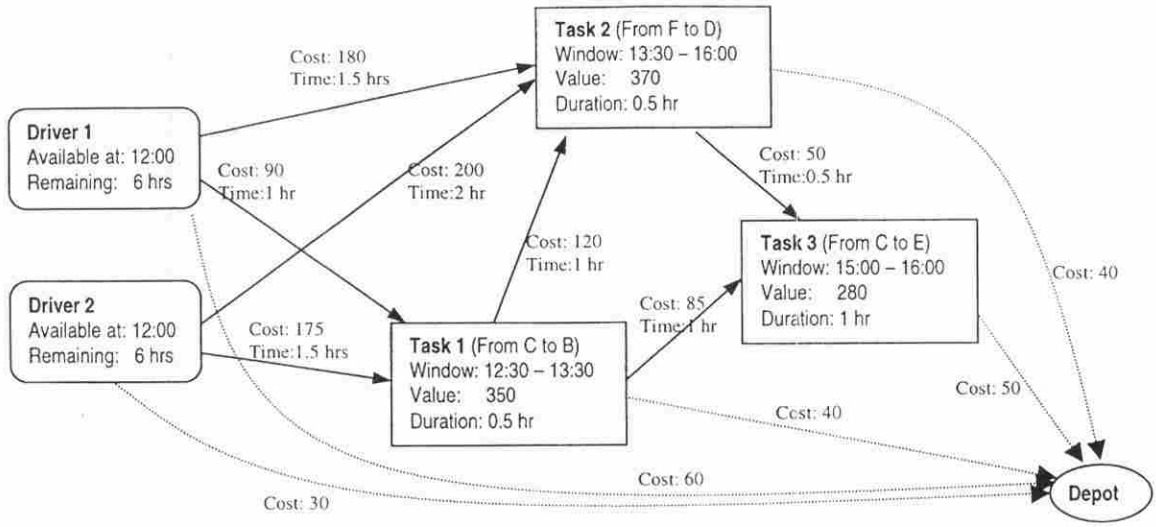

Fig. 1. A deterministic driver-task assignment example.

In the problem, after a driver covers a task, the driver will become available again. This newly available driver is called a *future* driver and we say that the driver is *generated* by the task just covered. The drivers initially available are assumed to be *generated* by some dummy tasks whose finishing locations are at the depot. The ending of the driver routes can likewise be represented by dummy tasks whose starting locations are at the depot. Let

$\mathscr{I}'$ = the set of dummy tasks for the initial drivers;
$\mathscr{I}''$ = the set of dummy tasks that represent the end of driver routes.

For each task i, let

a_i = the earliest possible starting time;
b_i = the latest possible starting time;
$\mathscr{S}_i$ = the set of possible starting times = $\{a_i, \ldots, b_i\}$;
τ_i = the service duration;
$\mathscr{F}_i$ = the set of possible finishing times;
r_i = the reward attained from covering the task.

For the tasks in $\mathscr{I}'$ and $\mathscr{I}''$, a_i and b_i are the beginning and the end of the planning horizon respectively. The service times and rewards for tasks in $\mathscr{I}'$ or $\mathscr{I}''$ are zero.

The distances and costs are defined as follows:

t_{ij} = the travel time from the finishing location of task i to the starting location of task j, $i \in \mathscr{I}' \cup \mathscr{I}$, $j \in \mathscr{I} \cup \mathscr{I}''$;
c_{ij} = the travel cost from the finishing location of task i to the starting location of task j, $i \in \mathscr{I}' \cup \mathscr{I}$, $j \in \mathscr{I} \cup \mathscr{I}''$;
c_{ij}^{w} = the minimum of the waiting cost per unit time at task i's finishing location and that at task j's starting location.

The decision variables are:

$$x_{ij}^{fs} = \begin{cases} 1 & \text{if the driver that finishes task } i \text{ at time } f \\ & \text{starts task } j \text{ at time } s, \\ & f \in \mathscr{F}_i, s \in \mathscr{S}_j,\ i \in \mathscr{I}' \cup \mathscr{I},\ j \in \mathscr{I} \cup \mathscr{I}'',\ i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

The following three variables are defined for the convenience of describing the model:

s_j = the starting time of task j, $j \in \mathscr{I} \cup \mathscr{I}''$ (if task j is covered);
f_i = the finishing time of task i, $i \in \mathscr{I}' \cup \mathscr{I}$ (if task i is covered);

$$x_{ij} = \begin{cases} 1 & \text{if the driver that finishes task } i \text{ starts task} \\ & j, i \in \mathscr{I}' \cup \mathscr{I},\ j \in \mathscr{I} \cup \mathscr{I}'',\ i \neq j, \\ 0 & \text{otherwise.} \end{cases}$$

In terms of the decision variables, the dependent variables can be expressed as

$$s_j = \sum_{i \in \mathscr{I}' \cup \mathscr{I}} \sum_{f \in \mathscr{F}_i} \sum_{s \in \mathscr{S}_j} s\, x_{ij}^{fs}, \tag{1}$$

$$f_i = s_i + \tau_i, \tag{2}$$

$$x_{ij} = \sum_{f \in \mathscr{F}_i} \sum_{s \in \mathscr{S}_j} x_{ij}^{fs}. \tag{3}$$

The constraints of the problem are:

$$\sum_{i \in \mathscr{I}' \cup \mathscr{I}} x_{ij} \leq 1 \quad \forall j \in \mathscr{I}, \tag{4}$$

$$\sum_{i \in \mathscr{I}' \cup \mathscr{I}} x_{ij} = \sum_{l \in \mathscr{I} \cup \mathscr{I}''} x_{jl} \quad \forall j \in \mathscr{I}, \tag{5}$$

$$\sum_{i \in \mathscr{I}'} \sum_{j \in \mathscr{I} \cup \mathscr{I}''} x_{ij} \leq N, \tag{6}$$

$$f_i + t_{ij} - s_j \le (1 - x_{ij})M \quad \forall i \in \mathscr{I}' \cup \mathscr{I}, j \in \mathscr{I} \cup \mathscr{I}'', \quad (7)$$

$$a_i \le s_i \le b_i \quad \forall i \in \mathscr{I}, \quad (8)$$

$$x_{ij}^{fs} \in \{0, 1\} \quad \forall i \in \mathscr{I}' \cup \mathscr{I}, j \in \mathscr{I} \cup \mathscr{I}'', f \in \mathscr{F}_i, s \in \mathscr{S}_j. \quad (9)$$

Constraint (4) ensures that each task is covered at most once. Constraint (5) is the flow conservation constraint: a driver who covers a task will be available later. Constraint (6) indicates that the number of drivers that cover tasks cannot exceed the number of available drivers. Constraint (7) requires that a task's starting time be later than the time when the driver arrives at the task's starting location. Constraint (8) ensures that all the starting time windows of tasks are satisfied. Constraint (9) is the integrality requirement.

The objective of the problem is to minimize the total assignment costs minus the total rewards. The assignment cost for the driver who finishes task i and then starts task j, denoted by $\bar{c}_{ij}$, consists of the travel cost and the waiting cost:

$$\bar{c}_{ij} = c_{ij} + \max\{s_j - (f_i + t_{ij}), 0\} c_{ij}^{w}.$$

Therefore, the problem can mathematically written as:

$$\min \sum_{i \in \mathscr{I}' \cup \mathscr{I}} \sum_{j \in \mathscr{I} \cup \mathscr{I}''} (\bar{c}_{ij} - r_j) x_{ij}, \quad (10)$$

subject to (1)–(9).

2.2. *Stochastic dynamic model*

If the service times are random quantities, then a static route obtained by solving the integer program defined by (1)–(10) can become infeasible because constraint (7) could be violated. Consider the revised example shown in Fig. 2 where each of the five tasks has several possible service durations with some known probabilities. Suppose all tasks take the minimum service durations, then the route, driver 1 → task 1 → task 2 → task 3 → depot, is feasible. However, if the service times for both task 1 and task 2 are 1 hour, then the driver will arrive at the starting location of task 3 at 16:30, 30 minutes later than the latest start time and the route is not feasible. Therefore, under an uncertain environment, the routes for the drivers need to be formed in different time stages.

To describe the model better, consider the time-space representation of the dynamic assignment problem shown in Fig. 2. Assume that driver 1 is assigned to cover task 2 and driver 2 is assigned to cover task 1 and both the actual service times for tasks 1 and 2 are 0.5 hour. In Fig. 3, the vertical axis is for location, the horizon axis is for time and the planning horizon has 12 intervals (30 minutes each). Assume that all drivers are initially available at time 0. For uncovered tasks, small triangles are used to represent their possible starting times and circles are used to represent their possible finishing times.

2.2.1. *Decision buffer and assignment window*

There are two special characteristics of the driver-task assignment in the motivating application. First, when a driver becomes available, the dispatcher will make an assignment not only for this driver, but also for the drivers who will be available within a time buffer. This buffer is referred to as the *decision buffer*. The model assumes that at a given time if the remaining time of a task is expected to be shorter than the length of the decision buffer, then the actual finishing time of the task is known. In practice, the actual remaining time can be estimated quite accurately if the real-time information to trace the status of the task is available and when the length of the buffer is short. This is particularly true for the model since decisions are made on discrete time points. For example, in Fig. 3, if the decision buffer is two time units

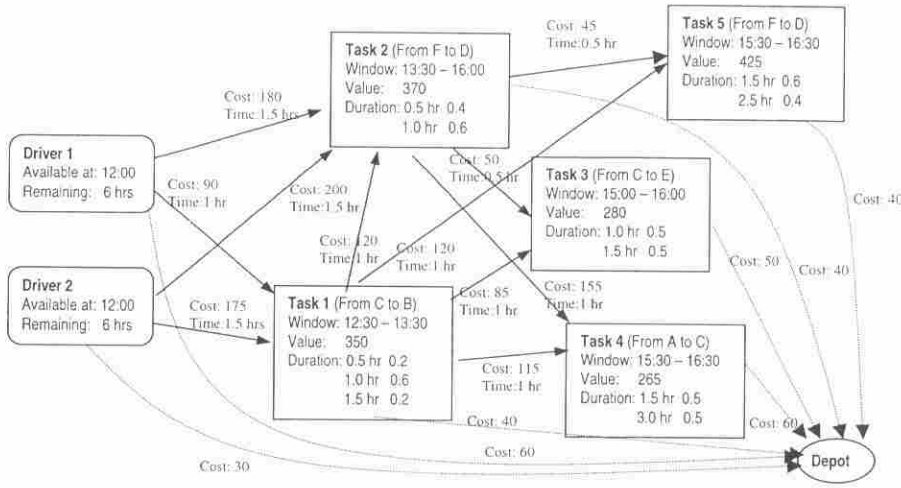

Fig. 2. A stochastic driver-task assignment example.

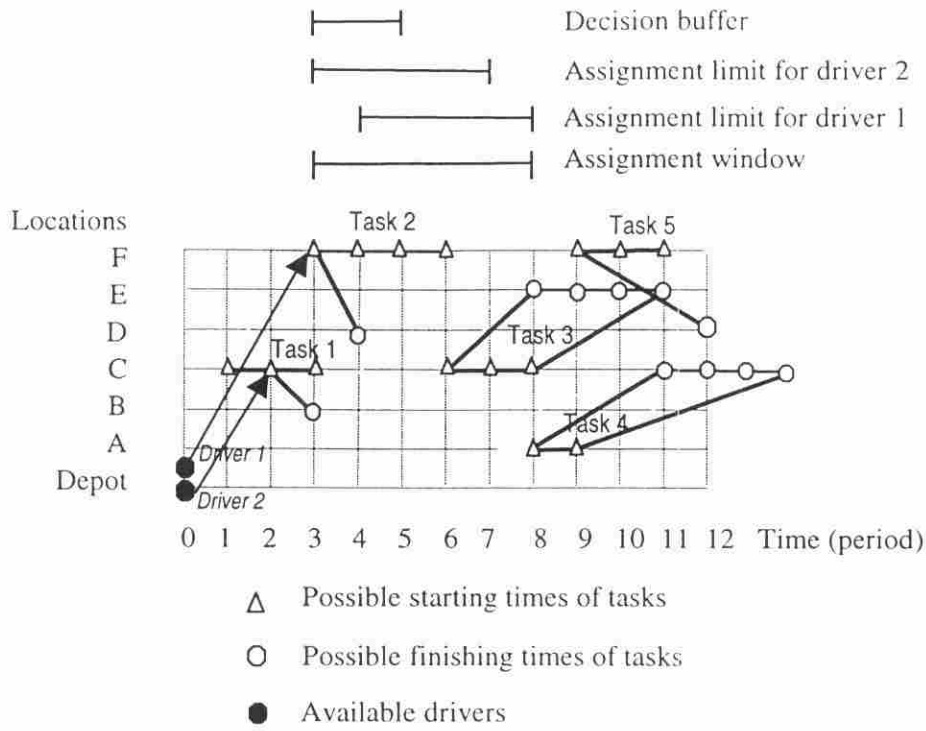

Fig. 3. A time-space representation of the driver-task assignment problem.

(1 hour), then both drivers 1 and 2 will be in the decision buffer starting at time 3. When the length of this buffer is zero, decisions are made for each individual driver (or multiple drivers if they are available at the same time).

Second, if there is no task that can be started within a certain time limit after a driver becomes available, for example 2 hours, then the driver will go back to the depot. The reason is that the driver can perform other internal tasks or take rests at the depot. In the motivating application, the depot is located at the container terminal where drivers can help handle some internal movements of containers at the terminal. This time period is referred to as the *assignment limit*. In Fig. 3, if the assignment limit is 2 hours, then task 4 is not within the assignment limit for driver 1 but it is within the assignment limit for driver 2, whereas task 5 is not within any of the assignment limits. We define the *assignment time window*, or simply the *assignment window*, as the period from the beginning of the decision buffer to the latest task starting time (latest available time of the drivers in the decision buffer plus the assignment limit). In the example, the assignment window is from three to eight. Thus, task 4 is within the assignment window whereas task 5 is not.

2.2.2. *Formulation*

In the stochastic model, τ_i is a random variable. Let

$\mathscr{T}_i$ = the set of possible service times of task i;
$p_{i,\tau}$ = the probability that the service time of task i is $\tau, \tau, \in \mathscr{T}_i$;
l = the time index that represents the beginning of an assignment window;
w_D = the length of the decision buffer.

For the assignment window starting at time l, define

$w_{A,l}$ = the length of the assignment window;
$\mathscr{I}_l^D$ = the set of tasks that may be finished within the decision buffer;
$\mathscr{I}_l^A$ = the set of tasks that may be started within the assignment window;
$\mathscr{S}_{j,l}$ = the set of possible starting times of task j (that is, the elements of $\mathscr{S}_j$) that are within the assignment window;
$\mathscr{F}_{i,l}$ = the set of possible finishing times of task i (that is, the elements of $\mathscr{F}_i$) that are within the decision buffer.

The set $\mathscr{I}_l^D$ generates drivers who will be available in the decision buffer. These drivers will cover the tasks in $\mathscr{I}_l^A$ within the assignment window. Notice that if the end of the decision buffer is beyond the end of the planning horizon, T, the length of the decision buffer will be shortened (if $l + w_D > T$, we set $w_D = T - l$).

Let Ω_l be the probability space for the decision stage starting from time l. An outcome $\omega_l \in \Omega_l$ represents the history of the drivers' available times before time $l + w_D$ (the end time of the current decision buffer). Suppose the duration of task i for a given ω_l is written as $\tau_i(\omega_l)$. The objective function of the optimization problem in this

stage consists of two parts. The first part is the total assignment cost minus the total reward within this stage and the second part is the expected total cost from the next stage to the end of the planning horizon. The start time of the next stage, denoted as l', is the finishing time of the first task that finishes after $l + w_D$. Mathematically speaking, the time is given by

$$l' = \min\left\{ f_j \text{ where } f_j \geq l + w_D \text{ and } j \in \bigcup_{\tilde{l} \leq l} \mathscr{I}_{\tilde{l}}^A \right\}. \quad (11)$$

Since $f_i = s_i + \tau_i(\omega_l)$, the start time of the next stage depends on the outcome ω_l. Therefore, unlike many multistage problems where the start times of the stages are known in advance, the start time of a stage in our problem is a random quantity and is known only after the decisions in previous stages are made and some random service times are realized.

The optimization problem of the current stage for the given ω_l is written as:

$$Q(l, \omega_l) = \min \sum_{i \in \mathscr{I}_l^D} \sum_{j \in \mathscr{I}_l^A} (\bar{c}_{ij} - r_j) x_{ij} + EQ(l', \omega_{l'}), \quad (12)$$

subject to constraints, (1)–(6), (8)–(9) and

$$s_i + \tau_i(\omega_l) + t_{ij} - s_j \leq (1 - x_{ij})M \quad \forall i \in \mathscr{I}_l^D, j \in \mathscr{I}_l^A. \quad (13)$$

Notice that constraint (7) is replaced by (13) in this stochastic formulation. The last term in (12) represents the expected future cost from the stage starting at time l'. If a driver finishes a task and becomes available at l' after time T, this driver is assumed to return to the depot. Thus, for $l' > T$, $EQ(l', \omega_{l'})$ are set to the travel costs of the returning trips. Finally, the objective function for the initial stage problem can be written as:

$$\min \sum_{i \in \mathscr{I}'} \sum_{j \in \mathscr{I}_0^A} (\bar{c}_{ij} - r_j) x_{ij} + EQ(l', \omega_{l'}), \quad (14)$$

where $l' = \min\{f_j \text{ where } f_j \geq w_D\}$.

3. The window sliding procedure

The deterministic model has a large number of integer decision variables and is difficult to solve computationally. In the stochastic model, the objective function in (12) involves a recursive embedding of the optimization in expectations, making the model even more difficult to solve.

We now consider a procedure that approximates the complex function $EQ(l', \omega_{l'})$ by a linear term. This linear term captures the impact of starting a particular task at a particular time. Let

v_j^t = the expected future cost of starting task j at time t.

The term $EQ(l', \omega_{l'})$ is then approximated by

$$\sum_{i \in \mathscr{I}_{l'}^D} \sum_{j \in \mathscr{I}_{l'}^A} \sum_{s \in \mathscr{S}_j} v_j^t x_{ij}.$$

Consequently, the objective function (12) is approximated by

$$\min \sum_{i \in \mathscr{I}_l^D} \sum_{j \in \mathscr{I}_l^A} (\bar{c}_{ij} - r_j) x_{ij} + \sum_{i \in \mathscr{I}_{l'}^D} \sum_{j \in \mathscr{I}_{l'}^A} \sum_{s \in \mathscr{S}_j} v_j^t x_{ij}. \quad (15)$$

The future value of starting a task at a given time depends on whether this task will be covered, when it is to be covered, and the future value that the driver who covers this task will generate.

Define for each task i, $i \in \mathscr{I}_l^D$:

u_i^t = the expected future cost of finishing task i at time t and $t \in \mathscr{T}_{i,l}$;
q_i^t = the estimated probability that task i is finished at time t and $t \in \mathscr{T}_{i,l}$;
p_i^t = the estimated probability that task i is started at time t and $t \in \mathscr{S}_i$;
p_i = the estimated probability that task i is covered by a driver;
$\hat{x}_{s_i}$ = the number of times that the starting time of task i is s_i in our samples.

For given p_i^t, the probability q_i^t can be determined as

$$q_i^t = \sum_{\tau \in \mathscr{T}_i} p_{i,\tau} \times p_i^{t-\tau}. \quad (16)$$

The procedure iteratively estimates the values of v_j^t and the probabilities p_i^t and p_i. We assume that all tasks are to be covered initially and each task has the equal chance to start at each of its possible starting times (see Step WS1 below for the corresponding mathematical expressions). The procedure consists of multiple passes of sliding the assignment window from the end of the planning horizon towards the beginning of the planning horizon one period at a time. In each pass, we estimate v_i^t, $i \in \mathscr{I}_l^D$ during the sliding process and update p_i^t and p_i at the end of the pass. The estimation of v_i^t involves drawing samples on possible task finishing times within the decision buffer and solving a network flow problem for each of the samples. The details of how the network is constructed and how the values of v_i^t are computed are provided in Section 4.

To update the probabilities p_i (the likelihood of when task i will be covered), we consider the ratio of $\sum_{s_i \in \mathscr{S}_i} \hat{x}_{s_i}$ to $\sum_{i \in \mathscr{I}_l^D} \sum_{s_i \in \mathscr{S}_i} \hat{x}_{s_i}$. The quantity $\sum_{s_i \in \mathscr{S}_i} \hat{x}_{s_i}$ represents the number of times that task i is covered in the samples. If this number is larger for task i than for task i', then task i is more likely to be covered than is task i'. On the other hand, to update the probabilities p_i^t, we consider the ratio of $\hat{x}_t$ to $\sum_{t \in \mathscr{S}_i} \hat{x}_t$. This ratio indicates the number of times that a task is started at a particular time t over its possible start times. The weighted averages of these two ratios over the previous passes are then used to update p_i and p_i^t

through two smoothing parameters, β_p and $\beta_{p'}$, that are between zero and one as follows:

$$p_i^t = \beta_p p_i^t + (1-\beta_p)\frac{\hat{x}_t}{\sum_{t\in\mathscr{S}_i}\hat{x}_t}, \quad \forall t \in \mathscr{S}_i, \ \forall i \in \mathscr{I}, \quad (17)$$

$$p_i = \beta_{p'} p_i + (1-\beta_{p'})\frac{\sum_{s_i\in\mathscr{S}_i}\hat{x}_{s_i}}{\sum_{i\in\mathscr{I}_l^{\mathrm{D}}}\sum_{s_i\in\mathscr{S}_i}\hat{x}_{s_i}} \quad \forall i \in \mathscr{I}. \quad (18)$$

The procedure is summarized below.

Procedure: Assignment Window Sliding (WS)

Step WS1. Set $n = 0$,
$v_i^t = 0, \ \forall i \in \mathscr{I}, t = 1, \ldots, T$,
$p_i^t = \frac{1}{|\mathscr{S}_i|}, \forall i \in \mathscr{I}, t \in \mathscr{S}_i$,
$p_i = 1, \forall i \in \mathscr{I}$.

Step WS2. Set $n = n + 1$.
Set $\hat{x}_{s_i} = 0, \forall i \in \mathscr{I}, s_i \in \mathscr{S}_i$.
Reset the starting time assignment window: $l = T - w$.

Step WS3. Repeat
Define $\mathscr{I}_l^{\mathrm{D}}$ and $\mathscr{I}_l^{\mathrm{A}}$.
For the assignment window starting at l, use the cost estimation (CE) procedure (described in Section 4) to update the values of v_i^t and $\hat{x}_{s_i}$.
Set $l = l - 1$.
Until $l = 0$

Step WS4. The probabilities p_i^t and p_i are updated using (17)–(18).

Step WS5. The Steps WS2 to WS4 are repeated for a predefined number of passes.

4. Assignment window subproblem

To estimate v_i^t, $i \in \mathscr{I}_l^{\mathrm{D}}$ for a given set of v_i^t, $i \in \mathscr{I}_l^{\mathrm{A}}$ and p_i^t, $i \in \mathscr{I}_l^{\mathrm{D}}$ in step WS3, random samples on the finishing times of tasks in $\mathscr{I}_l^{\mathrm{D}}$ are drawn using the set of probabilities q_i^t. For each sample, a network flow problem is generated and then it is solved. Next, the optimal dual prices obtained are used to estimate the costs u_i^t, which in turn are used to estimate v_i^t. Section 4.1 shows how the network flow problem is created and Section 4.2 shows the steps of the cost estimation.

4.1. *Assignment network*

For each task in $\mathscr{I}_l^{\mathrm{D}}$, a set of *finishing nodes* are created to represent the possible finishing times and, for each task in $\mathscr{I}_l^{\mathrm{A}}$, a set of *starting nodes* are created to represent the possible starting times. These times are defined if they are within the assignment window. For each task in $\mathscr{I}_l^{\mathrm{A}}$, a capacity node is also created. Finally, for the whole network, a super-sink node is created for representing the end of the planning horizon. Denote

n_{s_j} = the starting node for starting task j at time s_j;
n_{f_i} = the finishing node for finishing task i at time f_i;
n_j = the capacity node of task j;
n_{ss} = the super-sink node.

The cost of leaving from the destination of task i at the specific time f and arriving at the origin of task j at the specific time s is written as:

$$c(f,s) = c_{ij} + (s - f - t_{ij})c_{ij}^{\mathrm{w}}.$$

There are four types of arcs created in the networks. First, for each task j in $\mathscr{I}_l^{\mathrm{A}}$, there is a *summation arc* from each of its n_{s_j} to n_j with a cost of zero and no capacity restriction. Second, for each n_j, there is a *capacity arc* to n_{ss} with zero cost and capacity one. This arc and the summation arcs of a task together ensure that constraint (4) for this task is satisfied. Third, for each n_{f_i}, there is an *assignment arc* from it to a starting node $n_{s'}$ of each task j where

$$s' = \arg\min\left\{c(f_i,s) - r_j + v_j^s \mid s \in \mathscr{S}_{j,l}, \ s - f_i \geq t_{ij}\right\}. \quad (19)$$

Equation (19) says that if a driver available at time f_i will cover task j (which is feasible only if $s - f_i \geq t_{ij}$), then this driver will choose the start time with the lowest cost. The arc cost is set to $c(f_i,s') - r_j + v_j^{s'}$ and the arc capacity is one. Finally, there is an arc from each n_{f_i} to n_{ss}, reflecting that a driver always has the option of returning to the depot. The arc cost is the transportation cost from the task's finishing location to the depot and the arc capacity is one.

The nodes and arcs for tasks 1, 2, 3 and 4 in our example are shown in Fig. 4. The supplies for the nodes n_{s_j} and n_j are zeros. The supply of n_{f_i} is either one or zero depending on whether task i is being covered and when the task finishes. Finally, we define π_{f_i} as the dual price at node n_{f_i} after the network flow problem is solved optimally.

4.2. *Cost estimation*

Assume that the network flow problem described in Section 4.1 is available and K samples are used. For each sample, a network is generated. The networks differ in the pattern of node supplies. The cost estimation involves three major steps for each network: generate the supplies to the nodes; solve the network flow problem to obtain the optimal duals, and use the duals to estimate v_i^t. The supplies depend on the tasks that will finish within the decision buffers. Let m be the number of such tasks and m should not exceed both the number of drivers initially available and the number of tasks that can be finished in the decision buffer. That is, we have

$$m = \min(|\mathscr{I}_l^{\mathrm{D}}|, N). \quad (20)$$

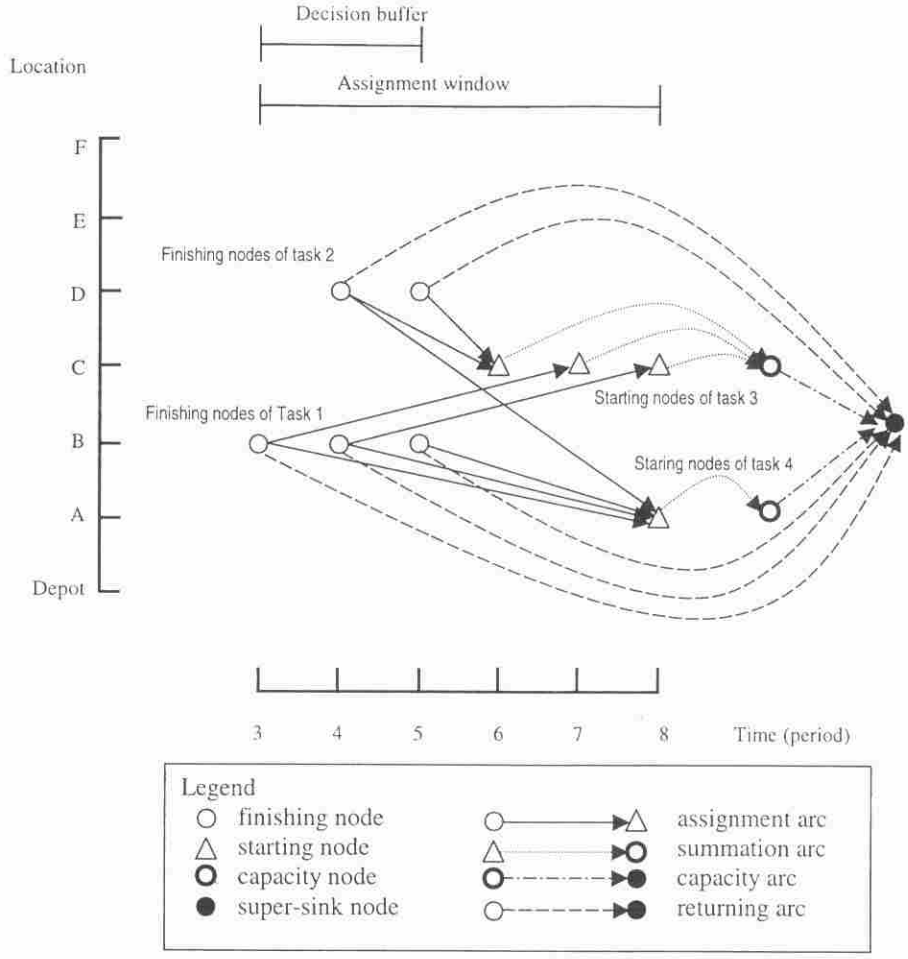

Fig. 4. Nodes and arcs in an assignment network.

If $|\mathscr{I}_l^{\mathrm{D}}| > m$, then m tasks are selected randomly according to the probabilities p_i. The finishing time of one of these tasks, say i, is selected according to q_i^t. Then, the supply of the corresponding finishing node is set to one. Since the total supply to the network is m, the demand of the super-sink node is set to m.

After the node supplies are set, the network flow problem is solved (by any standard algorithm such as the network simplex algorithm). The dual price, $\pi_{f_i}^t$, obtained from the optimal solution for the network problem measures the downstream impact of having an additional flow going into the finishing node $\pi_{f_i}^t$. Thus, the average dual prices over the K samples are used to update u_i^t where u_i^t measures the impact of finishing task i at time t. At the same time, we update the number of times that a particular task is started at a particular time (that is, $\hat{x}_{s_i}$). This number is used in Step WS4 of the (WS) procedure.

Given u_i^t, we measure the impact of starting this task i at a given time (that is, v_i^t) using the relationship:

$$v_i^t = \sum_{\tau \in \mathscr{T}_i} p_{i,\tau} u_i^{t+\tau}. \tag{21}$$

When it is the first time that v_i^t is considered in the entire (WS) procedure, Equation (21) is used. This happens only when it is in the first pass and task i has not been in the decision buffers of the previous assignment windows. The condition can be mathematically expressed as $n = 1$; $i \in \mathscr{I}_l^{\mathrm{D}}$ and $i \notin \mathscr{I}_{l+1}^{\mathrm{D}}$ hold. If v_i^t has been calculated before, then the value of v_i^t is set as the weighted average of the immediate past estimate and the new estimate. We use a parameter, α, that is between zero and one as a smoothing parameter to update v_i^t:

$$v_i^t = \alpha v_i^t + (1 - \alpha) \sum_{\tau \in \mathscr{T}_i} p_{i,\tau}\, u_i^{t+\tau}. \tag{22}$$

The steps of the procedure for cost estimation are summarized as follows.

Procedure: Cost estimation (CE)

Step CE1. Create the network as described in Section 4.1.

Step CE2. Set $k = 0$, $m = \min(|\mathscr{I}_l^{\mathrm{D}}|, N)$ and set the supply of n_{ss} to $-m$.
Set $u_i^{f_i} = 0$, $\forall f_i \in \mathscr{F}_i$, $i \in \mathscr{I}_l^{\mathrm{D}}$.

Step CE3. For tasks in $|\mathscr{I}_l^{\mathrm{D}}|$, select m tasks randomly according to p_i.

Step CE4. For a selected task, generate a finishing time f according to q_i^t.
If $f_i \in \mathscr{F}_{i,1}$, set the supply of n_{f_i} to one.

Step CE5. Solve the network flow problem.
For each finishing node n_{f_i}, obtain the dual price, π_{f_i}, and set

$$u_i^{f_i} = u_i^{f_i} + \frac{1}{K}\pi_{f_i}.$$

For each n_{s_j}, if the inbound flow is one, then

$$\hat{x}_{s_j} = \hat{x}_{s_j} + 1.$$

Step CE6. Set $k = k + 1$ and repeat Steps CE3 to CE5 until $k = K$.

Step CE7. Given the values of $u_i^{f_i}$, the expected future cost v_i^t can be determined as follows:
If $n = 1$, $i \in \mathscr{I}_l^{\mathrm{D}}$ and $i \notin \mathscr{I}_{l+1}^{\mathrm{D}}$, then apply (21) to obtain v_i^t.
Else if $i \in \mathscr{I}_l^{\mathrm{D}}$ and $i \in \mathscr{I}_{l+1}^{\mathrm{D}}$, then apply (22) to obtain v_i^t.

5. Numerical experiments

Numerical experiments were conducted to evaluate the benefit of modeling the uncertainty of the service times explicitly and the efficiency of the proposed procedure on some randomly created problems. We describe the characteristics of the test problems, consider the implementation of the algorithm, and offer a discussion of the computational results.

The algorithm is implemented in Java1.2 and all tests are performed on a personal computer with PIII 600 MHz CPU and 128 MB RAM. For simplicity, let $U(l, u)$ be the distribution of a discrete, uniform random variable that takes any integer between l and u inclusively unless specified. The time unit is 10 minutes. There are several parameters to be set. The problem sets differ in size and in the percentage of tasks that have starting time windows. A task without time windows can only be started at a particular time. The planning horizon has 72 periods and the driver-task assignments take place within a 12-hour period.

5.1. *Problem sets*

Test problems are generated for different combinations of the number of tasks, the number of drivers, and the percentage of drivers that have time-window requirements. The number of tasks can be 15, 20, 30 and 50 tasks. The number of drivers for each problem is set to one-fifth of the number of tasks. The percentage of tasks with starting time windows can be 25, 50 or 75%. For each time-window, the middle point is randomly generated according to $U(0, 72)$ and the width follows $U(12, 36)$. However, if any part of a start time window is beyond the planning horizon, the time-window is truncated and only the part that is inside the planning horizon is considered.

To compute the travel distances and times between tasks, the origin and destination of each task are randomly generated in a 100 by 100 units square. The travel distance between the two locations with coordinates (x_1, y_1) and (x_2, y_2) respectively is calculated by $|x_2 - x_1| + |y_2 - y_1|$, whereas the travel speed is set to 100 distance units per hour (or 16.7 distance unit per time unit). For each task, there are two possible service durations in which the probability of the first duration used is uniformly between zero and one. Each of the two service durations is the sum of the travel time and the time required for intermediate activities (such as loading and unloading), which is generated by $U(2, 12)$.

The travel cost is set to 150 per unit time and the waiting cost to 75 per unit time. The reward for finishing each task depends on its expected service time and is set to 180 per unit time.

5.2. *Implementation issues*

To evaluate the impact of random service times in the model on the solution quality, three versions of the algorithm are considered:

1. Myopic version (denoted as (WS-M)): All expected future costs are set to zero. This version aims at minimizing the current assignment cost but ignores the downstream impact.
2. Deterministic version (denoted as (WS-D)): The service time of each task is set to its rounded expected value.
3. Stochastic version (denoted as (WS-S)): This implementation considers both future costs and uncertain task service times.

For a given problem size and a given percentage of tasks that have time windows, we randomly generate a problem and then generate randomly 1000 possible scenarios (on the combinations of service times) for the problem. For each scenario and for each version of the algorithm, the assignment cost, denoted by C, is obtained using a rolling horizon implementation as follows:

Step 1. Apply (WS-M), (WS-D) or (WS-S) to obtain v_i^t (in the case of (WS-M), $v_i^t = 0$).

Step 2. Set $t = 0$, $C = 0$.

Step 3. Assign the drivers in the decision buffer starting at time t by solving an assignment problem:

$$\min \sum_{i \in \mathscr{I}_l^{\mathrm{D}}} \sum_{j \in \mathscr{I}_l^{\mathrm{A}}} \sum_{f \in \mathscr{F}_i} \sum_{s \in \mathscr{S}_j} \left(\bar{c}_{ij} - r_j + v_j^s\right) x_{ij}^{fs},$$

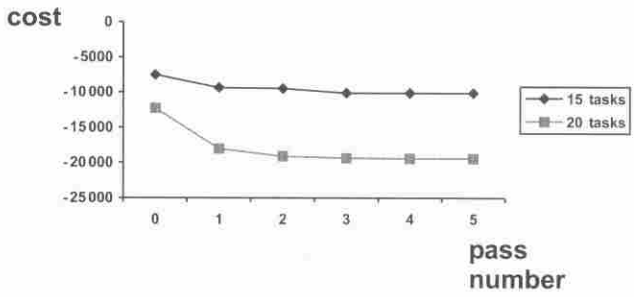

Fig. 5. The impact of the number of passes on average costs.

subject to constraints (1)–(4), (6), (8), (9) and (13). The durations $\tau_i(\omega_l)$ in (13) are given. Let $\bar{x}_{ij}^{fs}$ be the optimal solution.

Step 4. Set $C = C + \sum_{i \in \mathcal{I}_l^D} \sum_{j \in \mathcal{I}_l^A} \sum_{f \in \mathcal{F}_i} \sum_{s \in \mathcal{S}_j} (\bar{c}_{ij} - r_j)\bar{x}_{ij}^{fs}$.

Step 5. Execute the decision $\bar{x}_{ij}^{fs}$ for the current stage. Advance t to the first task finishing time after the current decision buffer.

Step 6. Repeat Steps 3–6 until $t > T$.

In Step 3, the estimated future cost v_i^t is considered when making the driver-task assignment; in Step 4, v_i^t is not used in computing the actual assignment cost.

The parameters used in the (WS) procedure and the (CE) procedure are set as follows: the width of the decision buffer is set to 2 hours and the assignment limit is set to 4 hours. The updating scalars, β_p, $\beta_{p'}$ and α, are all set to 0.5. To select the number of passes required in the (WS) procedure, the total assignment costs in 1000 randomly sampled scenarios of the 15-task and the 20-task problems (50% tasks with time windows) at different passes are compared. The results shown in Fig. 5 indicate that when the number of passes is larger than three, the marginal cost reduction is very small. Therefore, the predefined number of passes in Step (WS5) is set to three. In the (CE) procedure, the sample size, K, is the minimum of 500 and the number of possible combinations of the available driver times in the decision buffer. For example, if we have four tasks in the decision buffer and each task has four possible starting times, then there are 256 combinations and K is set to 256. If each task has five possible starting times, then there are 1024 combinations and K is set to 500.

5.3. *Computational results*

Table 1 shows the computational results on the randomly generated problems of different sizes and different percentages of tasks that have time windows. Columns 1 and 2 describe the number of drivers and tasks for the test problems. Column 3 shows the percentage of tasks that have time windows. Columns 4–6 report the assignment costs for (WS-M), (WS-D), and (WS-S) respectively. Columns 7–9 show the number of scenarios, out of 1000, for which a particular implementation gives the best assignment cost. For example, the last number in the first row means that in 781 out of the 1000 cases, (WS-S) produces the best results. Notice that for some problems, the total of the last three numbers exceeds 1000. This happens because more than one implementation may produce the best cost in a scenario.

The computation results show that the costs obtained by (WS-S) are, on average, 6.4% lower than those by (WS-D). The costs obtained by (WS-D) are in turn 55.7% lower than those by (WS-M). With respect to the number of scenarios for which a particular implementation gives the best costs, (WS-M) is the best only in a very small number of cases while (WS-S) is better than (WS-D) at a

Table 1. Summary of average driver-task assignment costs

Problem size		*Time window percentage*	*Average cost*			*Number of scenarios that performs the best*		
Number of drivers	*Number of tasks*	*(%)*	*(WS-M)*	*(WS-D)*	*(WS-S)*	*(WS-M)*	*(WS-D)*	*(WS-S)*
3	15	25	−6640	−7470	−7577	33	390	781
		50	−7560	−9504	−10111	26	310	688
		75	−8105	−9913	−10225	61	546	708
4	20	25	−10745	−16869	−18057	0	212	788
		50	−12289	−19026	−19373	0	501	583
		75	−13344	−23288	−24917	0	292	714
6	30	25	−18346	−27982	−28848	0	472	561
		50	−19101	−30562	−39822	0	225	777
		75	−23982	−39439	−40023	0	432	671
10	50	25	−32974	−52074	−54484	0	400	604
		50	−38841	−52253	−56051	0	356	650
		75	−45525	−61590	−63525	0	303	697

Table 2. Summary of computation times for cost estimation

Problem size		*Time for cost estimation (seconds)*	
Number of drivers	*Number of tasks*	*(WS-D)*	*(WS-S)*
3	15	13	14
4	20	14	17
6	30	21	24
10	50	31	33

ratio of 2:1. It is clear that considering the downstream impact when making the current assignment decisions is essential. The additional consideration of uncertainty explicitly can further improve the solution quality.

Finally, it is observed that by increasing the number of tasks that have time windows, the assignment cost is quite significantly reduced. The more flexibility that we allow for when a task can be started, the lower the assignment cost.

The computation time consists of two parts: the time for cost estimation (that is, computing v_i^t) and the time for making driver-task assignments. For the first part, Table 2 gives the average times required by (WS-D) and (WS-S), which range from 13 to 33 seconds. Given the complexity and the size of the problems, these times are considered short. For the second part, the computation times for making the actual driver-task assignments (that is, Step 2 to Step 6 in the rolling horizon implementation) are less than 0.1 seconds in all scenarios for any problem size.

6. Conclusions

The goal of this paper is to develop a procedure for dynamic driver-task assignments in short-haul land container transportation, where stochastic task service times and task starting time windows are considered. The problem is formulated in a stochastic optimization framework with the objective of minimizing the costs of the current stage driver-task assignment and the expected future costs. An assignment window sliding procedure is proposed to estimate the future cost of starting a task at a particular time. The procedure involves solving minimum cost flow problems for different samples of available driver times. Numerical experiments are conducted to evaluate the benefit of modeling the uncertainty explicitly and the efficiency of the solution procedure. The experiment results indicate that the model and the procedure take benefits from considering both the future assignment costs and the stochastic service times when making assignment decisions. In terms of CPU time, the procedure can produce solutions efficiently.

Acknowledgements

We thank the referees for their helpful suggestions on improving the paper. The research is supported by CERG Grant HKUST6205/99E of the Hong Kong Research Grants Council.

References

Bertsimas, D.J. (1992) A vehicle routing problem with stochastic demand. *Operations Research*, **40**, 574–585.

Birge, J. and Louveaux, F. (1997) *Introduction to Stochastic Programming*, Springer, New York, NY.

Cheung, R.K. and 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.

Desrosiers, J., Dumas, Y., Solomon, M. and Soumis F. (1995) Time constrained routing and scheduling, in *Network Routing*, Ball, M.O., Magnanti, T.L., Monna, C.L. and Nemhauser, G.L. (eds.), North-Holland, Amsterdam, pp. 35–139.

Desrosiers, J., Soumis, F. and Desrochers, M. (1984) Routing with time windows by column generation. *Networks*, **14**, 545–565.

Ermoliev, Y. (1988) Stochastic quasigradient methods, in *Numerical Methods for Stochastic Optimization*, Ermoliev, Y. and Wets, R. (eds.), Springer-Verlag, Berlin, pp. 141–185.

Fischetti, M., Martello, S. and Toth, P. (1987) The fixed job schedule program with spread-time constraints. *Operations Research*, **35**, 849–858.

Fischetti, M., Martello, S. and Toth, P. (1989) The fixed job schedule program with working-time constraints. *Operations Research*, **37**, 395–403.

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

Gendreau, M., Laporte, G. and Séguin, R. (1995) An exact algorithm for the vehicle routing problem with stochastic demands and customers. *Transportation Science*, **29**, 143–155.

Gendreau, M., Laporte, G. and Séguin, R. (1996a) Stochastic vehicle routing. *European Journal of Operational Research*, **88**, 3–12.

Gendreau, M., Laporte, G. and Séguin, R. (1996b) A tabu search heuristic for the vehicle routing problem with stochastic demands and customers. *Operations Research*, **44**, 469–477.

Jagannathan, R. (1985) Use of sample information in stochastic recourse and chance-constrained programming models. *Management Science*, **31**, 96–108.

Laporte, G. and Louveaux, F. (1993) The integer L-shaped method for stochastic integer programs with recourse. *Operations Research Letters*, **13**, 133–142.

Laporte, G., Louveaux, F. and Mercure, H. (1992) The vehicle routing problem with stochastic travel times. *Transportation Science*, **26**, 161–170.

Powell, W.B., Snow, W. and Cheung, R.K. (2000) Adaptive labeling algorithms for the dynamic assignment problem. *Transportation Science*, **34**, 50–66.

Rockafellar, T.R. and Wets, R. (1991) Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, **16**, 119–147.

Secomandi, N. (1998) Exact and heuristic dynamic programming algorithms for the vehicle routing problem with stochastic demands. Ph.D. dissertation, Department of Decision and Information Sciences, University of Houston, Houston, TX.

Biographies

Raymond K. Cheung is an Associate Professor in the Department of Industrial Engineering and Engineering Management of the Hong Kong University of Science and Technology. His teaching and research interests include transportation systems management, logistics and distribution planning, and network modeling and optimization. He earned a Ph.D. degree in Civil Engineering and Operations Research from Princeton University. He received the George B. Dantzig dissertation prize from INFORMS and a Career Award from the National Science Foundation of America. In 1996, he went to Hong Kong to help initiate the logistics research and teaching programs. He has published papers in journals such as *IIE Transactions*, *Operations Research*, *Naval Research Logistics*, *Transportation Science*, and *Networks*.

Darren D. Hang is an optimization expert at Carmen Systems in Gothenburg, Sweden. He earned a Ph.D. in Industrial Engineering and Engineering Management from the Hong Kong University of Science and Technology, an M.S. and a B.S. from Nanjing University of Science and Technology, Nanjing, China. His recent work with Carmen Systems has been on research and development in resource optimization for airlines and railway transportation.

Contributed by the Location and Transportation Modeling Department