Transportation Research Part E 45 (2009) 96–106

Contents lists available at ScienceDirect

Transportation Research Part E

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

Anticipatory algorithms for same-day courier dispatching

Gianpaolo Ghiani [a,*], Emanuele Manni [a], Antonella Quaranta [a], Chefi Triki [b]

[a] Dipartimento di Ingegneria dell'Innovazione, Università del Salento, via per Monteroni, 73100 Lecce, Italy
[b] Dipartimento di Matematica, Università del Salento, Via Per Arnesano, 73100 Lecce, Italy

ARTICLE INFO

Article history:
Received 12 February 2008
Received in revised form 26 June 2008
Accepted 7 August 2008

Keywords:
Real-time vehicle routing and dispatching
Courier industry
Waiting strategies
Idle vehicle relocation

ABSTRACT

This paper describes anticipatory algorithms for the dynamic *vehicle dispatching problem with pickups and deliveries*, a problem faced by local area courier companies. These algorithms evaluate alternative solutions through a short-term demand sampling and a fully sequential procedure for indifference zone selection. They also exploit an unified and integrated approach in order to address all the issues involved in real-time fleet management, namely assigning requests to vehicles, routing the vehicles, scheduling the routes and relocating idle vehicles. Computational results show that the anticipatory algorithms provide consistently better solutions than their reactive counterparts.

1. Introduction

Same-day couriers are utilized by clients who require maximum speed and security for deliveries of letters and small parcels in urban areas. Customers usually request couriers with little or no notice, all but eliminating the ability to construct routes and schedules in advance. The traditional model of same-day courier service utilizes dispatchers who communicate with bicycle, motorcycle, car and van couriers via radio or mobile phone. Controllers ask couriers to relay their location information and then assign jobs to the most appropriate vehicle. This model is not only inefficient, but also suffers from errors inherent with the involvement of a human element. Indeed, the recent advances in communication and information technologies, that now allow data on courier locations and customers requests to be obtained and processed in real-time, have stimulated research on algorithms for dynamic vehicle routing and dispatching problems. Unlike their static counterparts, these problems are characterized by data which are disclosed in a dynamic fashion over a planning horizon. In this context, decisions made at an early stage of the planning horizon might affect the ability to make good decisions at a later stage. Nonetheless, large part of the current literature is focused on algorithms reacting to new requests only once they have occurred, while neglecting available stochastic information. As a result, these algorithms are not able to take advantage of recurrent patterns in customer demands and vehicle travel times, which usually constitutes an easy task for dispatchers.

The purpose of this article is to describe and assess anticipatory heuristics for the dynamic and stochastic *vehicle dispatching problem with pickups and deliveries* that anticipate future demands through a Monte Carlo sampling procedure. The insight gained by simulating near-future demand is used in order to manage in an unified way several kinds of decisions, including vehicle dispatching, route scheduling and idle vehicle relocation.

* Corresponding author. Tel.: +39 (0) 832297791; fax: +39 (0) 832297792.
E-mail address: gianpaolo.ghiani@unile.it (G. Ghiani).

1366-5545/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.tre.2008.08.003

2. Problem statement

The dynamic and stochastic *vehicle dispatching problem with pickups and deliveries* (VDPPD) is defined on a graph $G = (V, A)$, where V is a vertex set and A is an arc set. A fleet of m vehicles, located at a depot $i_0 \in V$ at time $t = 0$, has to service a number of pickup and delivery requests $\{(i_k^+, i_k^-, T_k) : k = 1, 2, \ldots\}$, where $i_k^+ \in V$, $i_k^- \in V$, $T_k \geqslant 0$ are the pickup point, the delivery point and the occurrence time of the kth request, respectively. Vertices may represent individual customer locations or the zones in which the service territory is divided. Let t_{ij} be the shortest travel time from vertex $i \in V$ to vertex $j \in V$. The aim is to maximize the overall customer service level. Let τ_k be the delivery time of the kth request. With each customer is associated a non-decreasing and convex penalty function $f_k(\tau_k)$ expressing the inconvenience associated with customer i_k. This definition includes the case in which $f_k(\tau_k)$ represents the customer *system time* (i.e., $f_k(\tau_k) = \tau_k - T_k$, $\tau_k \geqslant T_k$) or a more involved penalty function (e.g., $f_k(\tau_k) = 0$, $T_k \leqslant \tau_k \leqslant D_k$ and $f_k(\tau_k) = \tau_k - D_k$, $\tau_k \geqslant D_k$, where D_k is a *soft deadline* associated with the kth request).

The static version of the VDPPD amounts to determining an ordered sequence of locations on each vehicle route such that:

1. each route starts at the depot;
2. a pickup and its associated delivery are satisfied by the same vehicle;
3. a pickup is always made before its associated delivery;
4. the total penalty incurred by the vehicles $z = \sum_k f_k(\tau_k)$ is minimized.

In the dynamic variant, we also have to adequately distribute waiting time along the routes, since this may affect the overall solution quality, as well as reposition idle vehicles to anticipate future demand. In this paper we assume that the requests arrive according to a known stochastic process. The objective function to be minimized is the expected customer inconvenience over the planning horizon:

$$z' = \sum_k E[f_k(\tau_k)]$$

where $E[f_k(\tau_k)]$ is the expected penalty associated with the delivery of the kth request. Moreover, we assume that a vehicle cannot be diverted away from its current destination to service a new request in the vicinity of its current position. Indeed, especially in the case of bike and motorbike couriers (which are very popular in Europe), drivers cannot pay attention continuously to their palmtop computers. As a result, courier companies often assume that once a driver is "en route" to his/her next location, he/she must necessarily service that location (i.e., the vehicle cannot be diverted away from its current destination). For an extensive treatment of diversion issues in real-time vehicle dispatching the reader is referred to Regan et al. (1994), Regan et al. (1995) and Ichoua et al. (2000).

In this paper we develop an algorithm in which, any time a new request arrives, the short-term arrival process is sampled and alternative solutions are compared. The key issue of the comparison is to determine the number of samples required for each alternative solution in order to select the best insertion with a given level of confidence. The remainder of the paper is organized as follows. In Section 3 the current literature on the dynamic VDPPD is surveyed and the literature on the anticipatory algorithms is reviewed. In Section 4 the algorithms are described while computational results and conclusions follow in Sections 5 and 6, respectively.

3. Literature review

The dynamic VDPPD falls under the broad category of real-time fleet management in which vehicle routes are built in an on-going fashion as customer requests, vehicle locations and travel times are revealed over the planning horizon. Overviews of these problems can be found in Psaraftis (1995), Gendreau and Potvin (1998), Ghiani et al. (2003) and Larsen et al. (2008). A large part of the current literature is characterized by algorithms reacting to new requests only once they have occurred, while neglecting available stochastic information. Reactive procedures for the dynamic VDPPD are described by Gendreau et al. (2006). This paper proposes adaptive descent and tabu search heuristics that utilize a neighborhood structure based on ejection chains. Numerical results show that when enough computing power is available, they produce improved results over simpler heuristics (even if the optimization takes place over known requests only, with no consideration for the future).

Assuming that some probabilistic knowledge is available, two different ways of exploiting this information are reported into the literature: analytical studies and anticipatory algorithms. The former approach examines dispatching and routing policies whose performance can be determined analytically if specific assumptions are satisfied. Along this line of research, Bertsimas and Van Ryzin (1991), and Bertsimas and Van Ryzin (1993) identify optimal policies both in light and heavy traffic, whereas demands are distributed in a bounded area in the plane and arrivals are modelled as a Poisson process. Papastavrou (1996) describes a routing policy that performs well both in light and heavy traffic, while Swihart and Papastavrou (1999) examine a dynamic pickup and delivery extension. Thomas and White (2004) introduce a problem in which the objective is to maximize rewards being received for serving customers minus costs incurred for travelling along arcs. They model the problem as a finite-horizon Markov decision process and perform numerical experiments in order to

compare the optimal policy with a reactive strategy that ignores potential customer requests. Other relevant contributions in this area are Branke et al. (2005) and Thomas (2007). We refer the reader to Larsen et al. (2008) for an in-depth description of these articles.

The latter approach aims at developing *anticipatory algorithms* that incorporate explicitly currently available information about future events. A seminal contribution in this research area is Powell et al. (1988), whose work was motivated by long-haul truckload trucking applications. These results have been subsequently extended in Godfrey and Powell (2002), Powell and Topaloglu (2003) and Spivey and Powell (2004). Other related contributions are Larsen (2000) and Larsen et al. (2002), Bent and Van Hentenryck (2004), van Hemert and La Poutré (2004), Ichoua et al. (2006) and Bent and Van Hentenryck (2007). Larsen (2000) and Larsen et al. (2002) describe some policies for relocating idle vehicles in anticipation of future demands. Bent and Van Hentenryck (2004) consider a vehicle routing problem with hard time windows where customer locations and service times are random variables which are realized dynamically during plan execution. They develop a multiple scenario approach which continuously generates plans consistent with past decisions and anticipating future requests. The current solution (or distinguished plan) is selected by a consensus function (Stefik, 1981) that chooses the solution most similar to a continuously updated pool of routings. No details are provided on the number of samples to be taken for each alternative solution. By exploiting probabilistic information about future service requests, van Hemert and La Poutré (2004) introduce the concept of fruitful regions in a dynamic routing context, i.e., regions that have a high potential of generating loads to be transported. These authors define potential schedules by sampling fruitful regions and provide an evolutionary algorithm for deciding whether to move or not towards one of such regions in anticipation of future demand. Waiting strategies are not addressed explicitly. Ichoua et al. (2006) develop a parallel tabu search heuristic that allows a vehicle to wait in its current zone if the probability of a future request reaches a particular threshold. Idle vehicle relocation is not dealt with. Bent and Van Hentenryck (2007) consider online stochastic multiple vehicle routing with time windows in which the goal is to maximize the number of serviced customers. Contrary to earlier algorithms which only move vehicles to known customers, they investigate waiting and relocation strategies in which vehicles may wait at their current location or relocate to arbitrary sites.

Particularly relevant to our paper are the articles by Mitrović-Minić et al. (2004) and Mitrović-Minić and Laporte (2004), albeit these contributions do not exploit probabilistic information about future requests. Mitrović-Minić et al. (2004) have recently proposed double-horizon based heuristics. These procedures make use of two different planning horizons (short-term and long term) to which different goals are applied. The idea is that in the short-term, it is important to concentrate on minimizing total route length, whereas in the long term, it is more crucial to preserve a certain amount of flexibility in order to better accommodate future requests. Mitrović-Minić and Laporte (2004) examine four waiting strategies for the dynamic *pickup and delivery problem with time windows* (PDPTW). In the dynamic PDPTW, the presence of time windows allows the vehicles to wait at various locations along their routes. The authors show that an adequate distribution of the waiting time may affect the planner's ability to make good decisions at a later stage.

This paper makes the following contributions to the literature. First, we describe a quite general anticipatory algorithm in which, every time a new request arrives, the arrival process is sampled and alternative solutions are evaluated with respect to the short-term expected inconvenience under perfect information. Moreover, we address in an unified and integrated way all the main issues involved in real-time fleet management (i.e., assigning requests to vehicles, routing the vehicles, scheduling the routes and relocating idle vehicles). With respect to Bent and Van Hentenryck (2004), Bent and Van Hentenryck (2007), and standard rollout policies (Bertsekas and Tsitsiklis, 1996; Secomandi, 2001), our approach has some peculiar features that make it novel: (a) we make use of a fully sequential procedure for indifference zone selection that allows to eliminate, at an early stage of experimentation, those solutions that are clearly inferior, thus reducing the overall effort to select the best; (b) we limit the computational effort by restricting the sampling procedure to a short-term horizon whose duration is empirically correlated to the problem data. As a result, the computational effort can be kept low, while still achieving dramatic improvements over a pure reactive procedure.

4. The anticipatory algorithms

We devise the following anticipatory mechanism which we then embed in both an insertion and a local search procedure. Let $P_k \subseteq \{1, \ldots, k\}$ be the set of pending requests (i.e., the requests have occurred but have not been serviced) at time T_k, when request (i_k^+, i_k^-, T_k) arrives. A *reactive* algorithm generates a new solution incorporating i_k^+ and i_k^- with the aim to minimize the total inconvenience associated with requests in P_k:

$$z_k = \sum_{r \in P_k} f_r(\tau_r).$$

On the contrary, our *anticipatory* algorithms aim at minimizing z_k plus the expected value (under perfect information) of the total penalty $\xi^{[T_k, T_k + \Delta t_k]}$ associated with the requests arriving in the short-term future $[T_k, T_k + \Delta t_k]$:

$$z'_k = \sum_{r \in P_k} f_r(\tau_r) + E[\xi^{[T_k, T_k + \Delta t_k]}], \tag{1}$$

where Δt_k is the short-term duration. Of course, our procedures become reactive if $\Delta t_k = 0$ $(k = 1, 2, \ldots)$.

We now provide an in-depth description of the main ingredients of the two anticipatory algorithms.

4.1. Approximation of the near-future inconvenience

In order to estimate $E[\xi^{[T_k,T_k+\Delta t_k]}](s)$ for any given solution s, we generate n_s samples of the near-future demand and compute the corresponding total penalties under perfect information: $\xi_j^{[T_k,T_k+\Delta t_k]}(s)$ $(j=1,\ldots,n_s)$. Then, we approximate $E[\xi^{[Tk,T_k+\Delta t_k]}](s)$ through its sample mean:

$$\overline{\xi^{[T_k,T_k+\Delta t_k]}}(s)=\frac{\sum_{j=1}^{n_s}\xi_j^{[T_k,T_k+\Delta t_k]}(s)}{n_s}$$

as well as compute confidence intervals on $E[\xi^{[T_k,T_k+\Delta t_k]}](s)$. For every solution s and for every sample j $(j=1,\ldots,n_s)$, we approximate the penalties under perfect information $\xi_j^{[T_k,T_k+\Delta t_k]}(s)$ by means of a cheapest insertion heuristic.

4.2. Determination of the number of samples

Simulation-based optimization is used to tackle optimization problems in which either the objective function or some functions appearing in the constraints are not known explicitly, and can be estimated through simulation models (Fu, 2002). In such a setting, the evaluation of the objective function z' via discrete event simulation sets up a dichotomy not present in deterministic optimization: that of the search process versus the evaluation process. Indeed, most of the computation is spent estimating the objective function for given values of the decision variables, which is different from the deterministic setting, where the search is the primary computational burden. Our approach is based on the observation that there is no reason a priori to assume that the number of samples n_s should be the same for all solutions s. Indeed, our procedure compares the alternative solutions by using a *fully sequential indifference zone selection* (IZS) procedure (Kim and Nelson, 2001). Such a procedure requires that the user specifies two parameters:

- an indifference zone width δ;
- a confidence level $1-\alpha$.

The goal of the IZS procedure is to select a solution with expected penalty that is within δ units of the optimal performance with a given level of confidence $1-\alpha$. In practice, parameter δ is set equal to the smallest relevant absolute difference in the expected penalty. An objective function gap less than δ units is considered negligible. The procedure takes only a single basic output from each alternative still in contention at each stage. Also, if there exists clear evidence that a solution is inferior, then it will immediately be eliminated from consideration. It can be shown Kim and Nelson (2001) that the set of candidate solutions reduces to a singleton in a finite number of iterations. We now illustrate this concept through an example (Fig. 1). For each of the five alternative solutions, 10 samples are initially taken and 99% confidence intervals are computed. On the basis of these results, we can discard solutions 1 and 5. We then take one more sample for each of the three remaining solutions and compare the updated confidence intervals. Such a procedure is iterated until all solutions but one are eliminated.

As customary in simulation-based optimization, we make use of common random numbers (CRNs) in order to reduce variance (Kim and Nelson, 2001). In the following, we report a more formal description of the screening and selection procedure, which is an adaption of the method presented in Kim and Nelson (2001).

Step 1. (*Initialization*). Select a confidence level $1-\alpha$, an indifference zone parameter δ and a first stage sample size $n_0 \geqslant 2$. Let $S=\{1,\ldots,p\}$ be the set of solutions still in contention. Obtain n_0 independent outputs $\xi_j^{[T_k,T_k+\Delta t_k]}(s)$ $(j=1,\ldots,n_0)$ from each solution $s \in S$. Set $r=n_0$.

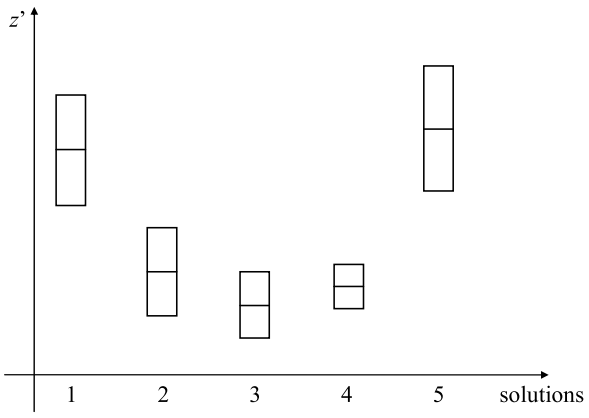

Fig. 1. Boxplot representation of five alternative solutions.

Step 2. (*Screening*). Let $\overline{\xi_s(r)} = \frac{\sum_{l=1}^{r} \xi_l^{[T_k, T_k+\Delta t_k]}(s)}{r}$ denote the sample mean of the first r outputs from solution s. For all $s' \neq s$ compute $\widehat{\sigma}^2_{ss'}(r)$, the sample variance of the difference between the values of solutions s and s':

$$\widehat{\sigma}^2_{ss'}(r) = \frac{\sum_{l=1}^{r} \left(\xi_l^{[T_k, T_k+\Delta t_k]}(s) - \xi_l^{[T_k, T_k+\Delta t_k]}(s') - \left[\overline{\xi}_s(r) - \overline{\xi}_{s'}(r) \right] \right)^2}{r-1}.$$

Set $S^{old} = S$. Let

$$S = \{ s : s \in S^{old} \quad \text{and} \quad \overline{\xi}_s(r) \leqslant \overline{\xi}_{s'}(r) + W_{ss'}(r), \forall s' \in S^{old}, s' \neq s \}$$

where

$$W_{ss'}(r) = \max \left\{ 0, \frac{\delta}{2r} \left(\frac{h^2 \widehat{\sigma}^2_{ss'}(r)}{\delta^2} - r \right) \right\},$$

and $h^2 = \left[\left(\frac{2\alpha}{p-1} \right)^{-2/(n_0-1)} - 1 \right] (r-1)$. $W_{ss'}(r)$ denotes how far the sample mean from solution s can stand above the sample mean of solution s' without being eliminated.

Step 3. (*Stopping Rule*). If $|S| = 1$, then stop and select the solution whose index is in S as the best. Otherwise, take one additional output $\xi_{r+1}^{[T_k, T_k+\Delta t_k]}(s)$ from each solution $s \in S$, set $r = r+1$ and go to Step 2.

4.3. Generation of alternative solutions

In the anticipatory insertion procedure, we generate alternative solutions as follows. At any time, the current solution can be seen as a set of routes $R_j = \{(i_{0,j}, i_{1,j}, \ldots, i_{n_j,j}, i_{n_j+1,j}), w_{0,j}, w_{1,j}, \ldots, w_{n_j,j}\}$ $(j = 1, \ldots, m)$, where $i_{0,j}$ is the current position of vehicle j, $i_{1,j}, \ldots, i_{n_j,j}$ are pickup and delivery points to be visited by vehicle j, $i_{n_j+1,j}$ is a parking position, and $w_{0,j}, w_{1,j}, \ldots, w_{n_j,j}$ are the waiting times of vehicle j at $i_{0,j}, i_{1,j}, \ldots, i_{n_j,j}$, respectively. In order to keep the computational effort within reasonable limits, it is worth selecting a reduced number κ of promising solutions to be assessed through near-future simulation. When a request (i_k^+, i_k^-, T_k) occurs, we generate every feasible insertion. Then, for each of the best κ insertions with respect to z_k, we enumerate all the feasible solutions associated with those insertions, by inserting waiting times along the individual route receiving request k as well as repositioning the corresponding vehicle when it becomes idle. In order to make the solutions feasible in the real world, we impose that the total waiting time along a route does not exceed a maximum amount MWT. This reflects the fact that large waiting times would seem a waste of time to both couriers and fleet supervisors, and would be judged as unacceptable. In addition, in order to keep directions to couriers simple, we require the waiting time at any node to be a multiple of a given quantity WQ (common WQ values are 5 or 10 min). Moreover, we assume that idle vehicles may be relocated only at a restricted number of home positions $H \subseteq V$ (which is what happens in real world). These operational restrictions are beneficial from a computational point of view, since they reduce the number of alternative solutions to be evaluated. For instance, if MWT = 20 min, WQ = 10 min and $H = \{i_q, i_{q'}\}$, a customer sequencing $(i_1^-, i_3^+, i_2^-, i_3^-)$ gives rise to the route schedules reported in Table 1. It is worth noting that two distinct route schedules, which coincide during the short-term future $[T_k, T_k + \Delta t_k]$, incur the same expected penalty (1) and require a single simulation experiment. Of course, the number of candidate solutions depends on the number of pending requests at time T_k.

Table 1
Feasible route schedules associated with customer sequencing $(i_1^-, i_3^+, i_2^-, i_3^-)$

Total waiting time along the route	Waiting time at i_1^-	Waiting time at i_3^-	Waiting time at i_2^-	Waiting time at i_3^-	Home positions
0	–	–	–	–	$i_q, i_{q'}$
10	10	–	–	–	$i_q, i_{q'}$
	–	10	–	–	
	–	–	10	–	
	–	–	–	10	
20	20	–	–	–	$i_q, i_{q'}$
	–	20	–	–	
	–	–	20	–	
	–	–	–	20	
	10	10	–	–	
	10	–	10	–	
	10	–	–	10	
	–	10	10	–	
	–	10	–	10	
	–	–	10	10	

4.4. Anticipatory local search

In our local search scheme, the neighbors of a given solution are obtained by removing in turn each pending request from its current route and reinserting it feasibly through the anticipatory insertion procedure described before. In order to keep the computational effort into reasonable limits, at each iteration we remove only the pending requests currently allocated to the routes modified in the previous iteration. Of course, if the pickup of a request has been already visited or a courier is currently moving towards it, then this request cannot be reinserted into another route.

5. Experimental results

Our computational experiments aim at assessing the effectiveness of our anticipatory algorithms versus a purely reactive procedure. Both heuristics have been coded in C++ and run on a PC with a Pentium IV processor clocked at 2.8 GHz. We utilize two sets of randomly generated instances, resembling standard and premium service, respectively. In the former instances ("ST" instances, in the following), a courier may consolidate an unlimited number of deliveries. In the latter instances ("PR" instances, in the following), consolidation is not allowed, so that vehicles may be assumed to have capacity equal to one. In both cases, the service territory is modelled as a grid with 25 zones, giving rise to 600 origin-destination pairs. Travel times between adjacent zones are set equal to 15 min. Demands are spatially independent and uniformly distributed over $Q = \{(o,d) \in \{1,\ldots,25\}^2 : o \neq d\}$. For each origin-destination pair, their arrival times constitute a Poisson process with rate $\lambda_{o,d}$ over a planning horizon [0,480] minutes. The arrival rates $\lambda_{o,d}$ are set equal to a constant c for 20 randomly chosen origin-destination pairs $(o,d) \in Q_1 \subseteq Q$ and to c/a for the remaining 580 pairs, where c and a are determined in such a way that the expected overall number of requests is a given value μ, and the expected number of requests between pairs in Q_1 is a given value μ_1. Moreover, we assume that the inconvenience associated with each request k is 0 for $\tau_k \leqslant D_k$, and $\tau_k - D_k$ otherwise, where $D_k = T_k + 45$ min for the "ST" instances, and $D_k = T_k + 20$ min for the "PR" instances.

Let $z_{P,i}$ be the objective function value provided by procedure P on instance i ($P = R, A, L$ stand for the reactive, anticipatory insertion, and anticipatory local search procedures, respectively). When comparing two algorithms (say, algorithms 1 and 2) on a class of n instances, let OBJDIFF and OBJSSD be the average deviation of the objective function values for the two procedures (i.e., $\frac{\sum_{i=1}^{n}(z_{1,i}-z_{2,i})}{n}$), and the sample standard deviation of the objective function values for the two procedures (i.e., $\sqrt{\frac{\sum_{i=1}^{n}(z_{1,i}-z_{2,i})^2}{n-1}}$), respectively. We generate as many instances n as are required to keep the variation coefficient of the sample mean (VC) of the objective function deviations $(\text{OBJSSD}/\sqrt{n})/|\text{OBJDIFF}|$ less than 0.1.

Moreover, algorithm parameters are set as follows: $\kappa = 5$, $\delta = 20$ min, $\alpha = 0.1$, $n_0 = 10$, MWT = 20 min, $WQ = 10$ min. Finally, we allow idle vehicles to be relocated to any of the three stochastic medians of the graph ($|H| = 3$).

5.1. Short-term duration tuning

The first aim of our computational experiments is to correlate empirically the duration Δt_k of the short-term horizon to the problem data. Intuitively, as Δt_k increases, we expect that the procedure becomes less and less myopic while the computational effort turns out to be heavier. On the other hand, very large Δt_k values do not provide any relevant additional objective function improvement. Tables 2–5, as well as Fig. 2, provide an account of the results whereas $\mu = \mu_1 = 200$ and $m = 36$. In particular, Tables 2 and 3, as well as Figs. 2a and 2b, refer to the case in which the expected service time (i.e., the expected time lag between the arrival of a request and its delivery instant) is around 40 min. Tables 4 and 5, as well as Figs. 2c and 2d, refer to the case in which the expected service time is around 50 min. The column headings which have not been explained yet are as follows:

- z_P: average objective function value (overall customer inconvenience) for procedure $P\left(\frac{\sum_{i=1}^{n} z_{P,i}}{n}\right)$;
- TIME_P: average CPU time (in seconds) for procedure P;
- SERVTIME_P: average CPU time per request (in seconds) for procedure P;
- ρ_P: average utilization rate of a vehicle (the fraction of time a vehicle is moving with at least one parcel on board, or is moving towards a pickup location) for procedure P;

Table 2
Preliminary computational experiments: "ST" instances, expected service time around 40 min

Δt_k	n	z_R	z_A	TIME_R	TIME_A	SERV TIME_R	SERV TIME_A	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
10	4	1692	1309	1.2	5454	0.01	26.1	0.63	0.35	−0.22	0.03	0.07	0.06
20	5	1729	1318	1.8	6891	0.01	33.5	0.63	0.35	−0.23	0.04	0.08	0.07
30	5	1729	1265	1.8	8120	0.01	39.4	0.63	0.36	−0.26	0.04	0.06	0.07
40	5	1729	1234	1.8	9378	0.01	45.5	0.63	0.36	−0.28	0.03	0.04	0.04
50	5	1729	1237	1.8	10725	0.01	52.1	0.63	0.36	−0.28	0.02	0.04	0.04
60	5	1729	1244	1.8	11957	0.01	58.1	0.63	0.37	−0.28 −0.26	0.02	0.04	0.05

Table 3
Preliminary computational experiments: "PR" instances, expected service time around 40 min

Δt_k	n	z_R	z_A	TIME$_R$	TIME$_A$	SERV TIME$_R$	SERV TIME$_A$	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
10	4	1736	1271	1.0	806	0.001	3.7	0.63	0.49	−0.26	0.02	0.05	0.05
20	5	1802	1078	1.2	2527	0.1	12.1	0.63	0.46	−0.39	0.07	0.09	0.13
30	5	1802	967	1.2	1942	0.1	9.3	0.63	0.45	−0.45	0.08	0.09	0.15
40	5	1802	947	1.2	2597	0.1	12.5	0.63	0.46	−0.47	0.07	0.07	0.14
50	4	1736	912	1.0	3259	0.001	15.7	0.63	0.46	−0.47	0.04	0.05	0.09
60	5	1802	925	1.2	3973	0.1	19.1	0.63	0.46	−0.48 −0.42	0.06	0.06	0.11

Table 4
Preliminary computational experiments: "ST" instances, expected service time around 50 min

Δt_k	n	z_R	z_A	TIME$_R$	TIME$_A$	SERV TIME$_R$	SERV TIME$_A$	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
10	4	1764	1309	1.2	5454	0.01	26.1	0.63	0.17	−0.25	0.03	0.06	0.03
20	5	1764	1316	1.8	14542	0.01	70.6	0.63	0.35	−0.23	0.04	0.08	0.07
30	5	1764	1247	1.8	17956	0.01	87.2	0.63	0.17	−0.29	0.03	0.05	0.02
40	5	1764	1230	2.2	21509	0.01	104.5	0.63	0.18	−0.30	0.02	0.03	0.04
50	5	1764	1217	1.8	24936	0.01	121.2	0.63	0.18	−0.31	0.03	0.05	0.06
60	5	1764	1224	2.2	28638	0.01	139.94	0.63	0.19	−0.30 −0.28	0.02	0.04	0.04

Table 5
Preliminary computational experiments: "PR" instances, expected service time around 50 min

Δt_k	n	z_R	z_A	TIME$_R$	TIME$_A$	SERV TIME$_R$	SERV TIME$_A$	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
10	4	1666	1061	1.2	2981	0.01	14.3	0.65	0.24	−0.37	0.04	0.07	0.1
20	5	1666	978	1.5	1666	0.01	21.2	0.63	0.23	−0.41	0.03	0.04	0.03
30	5	1666	842	1.8	7742	0.01	37.3	0.63	0.22	−0.49	0.02	0.02	0.02
40	5	1666	803	1.4	10197	0.01	49.1	0.63	0.22	−0.51	0.04	0.04	0.04
50	4	1666	810	1.8	12464	0.01	60.1	0.63	0.22	−0.53	0.008	0.009	0.02
60	5	1666	755	1.8	15003	0.01	72.2	0.63	0.22	−0.53 −0.47	0.06	0.05	0.05

- OBJDEV: average deviation of the objective function values of the two compared procedures (say, procedures 1 and 2), i.e., $\frac{z_1 - z_2}{z_2}$;
- CIW: width of the 0.9 confidence interval for the expected difference of the objective function values provided by the two compared procedures.

These results show that OBJDEV becomes almost constant (within $\pm 1\%$) when Δt_k approaches the expected service time at T_k. Hence, larger Δt_k values would not result in any further objective function improvement, while causing an increase in the average CPU time per request.

5.2. Reactive versus anticipatory insertion procedures

The second part of our experimental study aims at comparing the anticipatory insertion algorithm with its reactive counterpart. Tables 6–11 show the results for the "ST" and "PR" instances. In particular, Tables 6 and 7 refer to the case $\mu = 100$, $\mu_1 = 0, 50, 100$; Tables 8 and 9 refer to the case $\mu = 200$, $\mu_1 = 0, 50, 100, 150, 200$; Tables 10 and 11 refer to the case $\mu = 300$, $\mu_1 = 0, 100, 150, 200, 300$. The number of vehicles is set equal to 18, 36, and 54, respectively. This choice gives rise to similar vehicle utilization rates in the three experiments. The column heading which has not been explained yet is as follows:

- SAMPLES: average number of samples generated for each instance.

The results show that the anticipatory insertion procedure provides consistently better solutions than a purely reactive procedure on all generated instances. More precisely, the anticipatory algorithm allows achieving an average improvement of the objective function value ranging between 11% and 27% for the "ST" instances, whereas for the "PR" instances the improvements vary between 29% and 69%.

As the expected number of requests increases, the average inconvenience per request provided by both the reactive and anticipatory algorithms remains fairly constant (e.g., if $\mu = \mu_1$ the reactive procedure achieves an average inconvenience

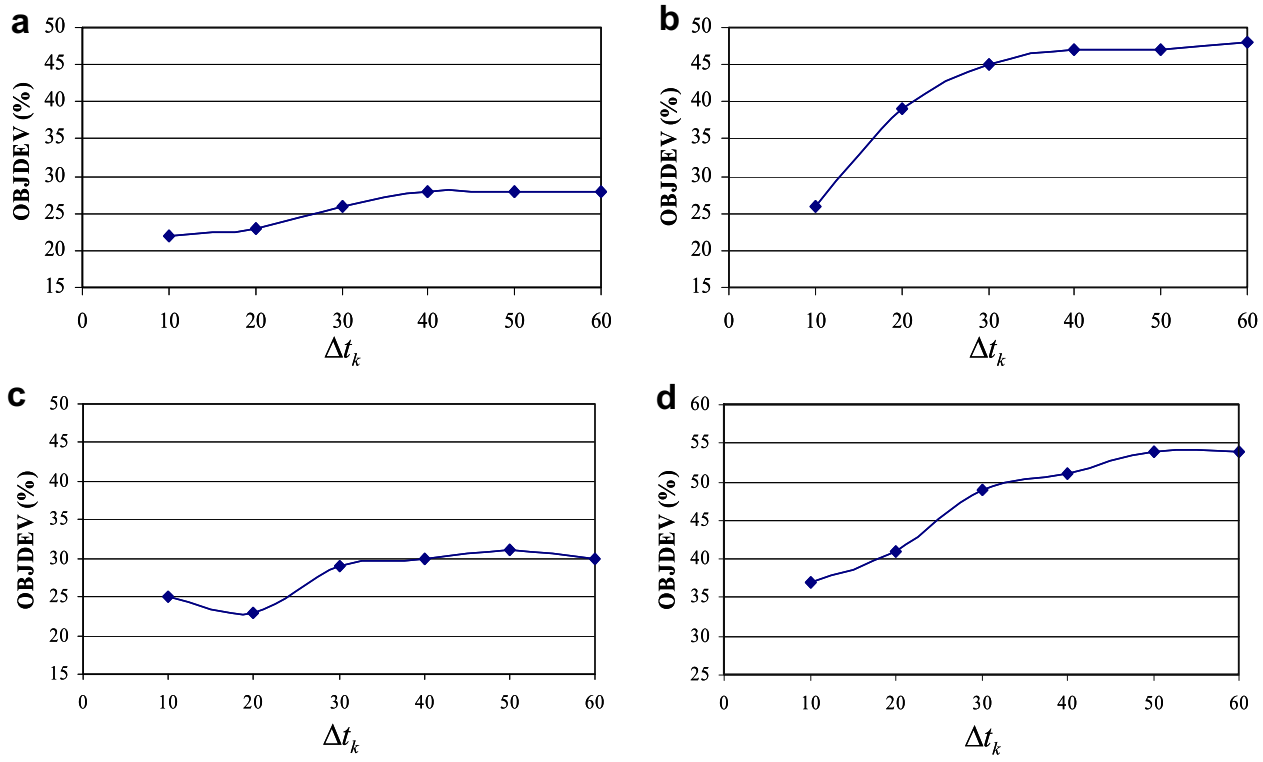

Fig. 2. Average deviation of the objective function values for the reactive and anticipatory procedures as a function of Δt_k: (a) and (c) refer to "ST" instances; (b) and (d) refer to "PR" instances. The expected service time is around 40 min in (a) and (b), around 50 min in (c) and (d).

Table 6
Comparison between the anticipatory and reactive insertion algorithms: "ST" instances, 100 expected requests

μ_1	n	SAMPLES	z_R	z_A	TIME$_R$	TIME$_A$	SERV TIME$_R$	SERV TIME$_A$	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
100	5	13.6	858	621	1.0	2380	0.01	23.7	0.53	0.39	−0.27	0.02	0.05	0.05
50	3	13.2	866	645	1.4	2490	0.01	25.5	0.52	0.42	−0.25	0.01	0.03	0.03
0	6	11.4	1087	906	1.0	2473	0.01	24.5	0.53	0.49	−0.16	0.03	0.10	0.03
											−0.22			

Table 7
Comparison between the anticipatory and reactive insertion algorithms: "PR" instances, 100 expected requests

μ_1	n	SAMPLES	z_R	z_A	TIME$_R$	TIME$_A$	SERV TIME$_R$	SERV TIME$_A$	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
100	5	12.1	660	449	1.3	288	0.01	2.6	0.55	0.45	−0.31	0.05	0.08	0.09
50	5	13.6	613	398	1.4	269	0.01	2.6	0.55	0.46	−0.35	0.07	0.10	0.12
0	4	12.4	745	526	1.3	266	0.01	2.4	0.54	0.47	−0.29	0.03	0.08	0.08
											−0.31			

Table 8
Comparison between the anticipatory and reactive insertion algorithms: "ST" instances, 200 expected requests

μ_1	n	SAMPLES	z_R	z_A	TIME$_R$	TIME$_A$	SERV TIME$_R$	SERV TIME$_A$	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
200	5	12.4	1729	1302	2.1	4743	0.01	23.04	0.63	0.37	−0.25	0.03	0.06	0.026
150	5	11.9	1751	1454	1.8	4738	0.01	22.75	0.62	0.39	−0.17	0.04	0.10	0.035
100	5	13.1	1770	1511	1.8	4683	0.01	23.26	0.62	0.45	−0.15	0.05	0.10	0.047
50	5	14.2	1795	1599	1.4	4913	0.01	24.79	0.60	0.46	−0.11	0.06	0.10	0.051
0	4	13.4	1814	1601	1.3	5109	0.01	26.04	0.58	0.44	−0.18	0.02	0.10	0.021
											−0.17			

Table 9
Comparison between the anticipatory and reactive insertion algorithms: "PR" instances, 200 expected requests

μ_1	n	SAMPLES	z_R	z_A	TIME$_R$	TIME$_A$	SERV TIME$_R$	SERV TIME$_A$	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
200	5	13.6	1667	936	1.7	2563	0.01	12.35	0.64	0.45	−0.44	0.05	0.08	0.056
150	5	12.8	1691	830	2.0	2350	0.01	11.34	0.63	0.46	−0.50	0.11	0.10	0.104
100	4	11.7	1366	707	1.8	2214	0.01	11.11	0.63	0.43	−0.59	0.09	0.10	0.085
50	5	13.5	1274	787	1.8	2090	0.01	10.69	0.63	0.45	−0.38	0.03	0.09	0.029
0	4	14.1	1391	726	2.2	2156	0.01	11.11	0.65	0.46	−0.48	0.08	0.10	0.083
											−0.48			

Table 10
Comparison between the anticipatory and reactive insertion algorithms: "ST" instances, 300 expected requests

μ_1	n	SAMPLES	z_R	z_A	TIME$_R$	TIME$_A$	SERV TIME$_R$	SERV TIME$_A$	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
300	5	13.4	2420	1817	2.1	34077	0.01	111.43	0.68	0.31	−0.25	0.02	0.06	0.05
200	5	11.9	2512	1936	1.8	36007	0.01	117.39	0.67	0.38	−0.23	0.01	0.04	0.04
150	5	11.1	2751	2152	1.8	38883	0.01	129.94	0.60	0.41	−0.22	0.04	0.10	0.08
100	5	14.2	2544	1947	2.4	52796	0.01	175.13	0.64	0.41	−0.24	0.04	0.10	0.09
0	4	12.4	2792	2229	1.8	37628	0.01	124.40	0.61	0.42	−0.20	0.03	0.09	0.06
											−0.22			

Table 11
Comparison between the anticipatory and reactive insertion algorithms: "PR" instances, 300 expected requests

μ_1	n	SAMPLES	z_R	z_A	TIME$_R$	TIME$_A$	SERV TIME$_R$	SERV TIME$_A$	ρ_R	ρ_A	OBJ DEV	OBJ SSD	VC	CIW
300	5	13.2	4428	1322	1.8	16217	0.01	51.79	0.69	0.37	−0.69	0.07	0.04	0.14
200	5	11.8	2742	1094	2.4	14233	0.01	46.96	0.66	0.42	−0.59	0.06	0.06	0.12
150	4	12.7	1808	669	1.6	15745	0.01	52.72	0.67	0.45	−0.63	0.04	0.04	0.09
100	5	12.2	2037	917	1.8	18721	0.01	62.94	0.68	0.41	−0.55	0.06	0.06	0.11
0	4	14.1	2188	944	2.6	13793	0.01	46.5	0.65	0.43	−0.57	0.05	0.05	0.11
											−0.60			

equal to 8.6, 8.6 and 8.1 min for the "ST" instances, with 100, 200 and 300 expected requests, respectively, whereas, under the same conditions, the anticipatory algorithm gives rise to 6.2, 6.5 and 6.1 min). The unique exception is represented by the reactive procedure, which turns out to perform poorer and poorer on "PR" instances of increasing size. Indeed, while the anticipatory procedure provides a fairly constant average inconvenience per request (4.5, 4.7 and 4.4 min with $\mu = \mu_1$ and 100, 200 and 300 expected requests, respectively), the reactive procedure produces an increasing average inconvenience per request (6.6, 8.3 and 14.8 min with $\mu = \mu_1$ and 100, 200 and 300 expected requests, respectively). These data explain the dramatic increase in the OBJDEV values for the "PR" instances, as the demand rate goes up.

Moreover, it is worth noting that the use of the anticipatory algorithm leads to an average vehicle utilization rate which is around 19% lower than its reactive counterpart.

In conclusion, the experiments show that the anticipatory procedure is very effective, leading to a consistent reduction in the customer inconvenience and to lower vehicle utilization rates.

5.3. Anticipatory insertion vs anticipatory local search procedures

In the third part of our experimental study, we compare the anticipatory insertion and local search algorithms. Tables 12 and 13 report the results for the "ST" and "PR", respectively. In both cases, $\mu = 200$, $\mu_1 = 0, 50, 100, 150, 200$, and $m = 36$.

Table 12
Comparison between the anticipatory insertion and local search algorithms ("ST" instances)

μ_1	n	SAMPLES	z_A	z_L	TIME$_A$	TIME$_L$	SERV TIME$_A$	SERV TIME$_L$	ρ_A	ρ_L	OBJ DEV	OBJ SSD	VC	CIW
200	5	12.8	1302	1258	4743	5167.6	23.04	25.11	0.37	0.33	−0.03	0.02	0.10	0.017
150	5	12.2	1454	1445	4738	5054.2	22.75	24.23	0.39	0.38	−0.01	0.01	0.08	0.001
100	5	13.4	1511	1502	4683	5296.6	23.26	26.34	0.45	0.43	−0.01	0.03	0.10	0.030
50	5	14.1	1599	1519	4913	5063.8	24.79	25.56	0.46	0.41	−0.05	0.06	0.09	0.057
0	4	13.7	1601	1592	5109	5124.0	26.04	26.09	0.44	0.43	−0.01	0.03	0.09	0.029
											−0.02			

Table 13
Comparison between the anticipatory insertion and local search algorithms ("PR" instances)

μ_1	n	SAMPLES	z_A	z_L	TIME$_A$	TIME$_L$	SERV TIME$_A$	SERV TIME$_L$	ρ_A	ρ_L	OBJ DEV	OBJ SSD	VC	CIW
200	5	11.8	936	930	2069	16272.6	12.34	45.30	0.64	0.62	−0.01	0.06	0.10	0.13
150	5	13.2	830	838	2350	14779.0	11.34	71.96	0.63	0.63	0.01	0.01	0.06	0.14
100	4	13.4	707	702	2214	15294.2	10.40	72.14	0.62	0.64	−0.01	0.05	0.07	0.10
50	5	14.1	787	764	2090	13203.6	10.68	67.36	0.62	0.64	−0.03	0.07	0.08	0.11
0	4	12.7	726	736	2156	9328.6	10.72	46.19	0.66	0.65	0.01	0.05	0.08	0.10
											−0.01			

The experiments show that in a typical same-day courier setting, the anticipatory local search allows to yield only very slight improvements (if any) on the simpler anticipatory insertion policy. The explanation of this behavior is twofold. First, our experiments reproduce an operational setting in which the fleet size is large enough to provide customers a high quality of service; as a result, the average number of pending requests along any route is usually low (1.8 in our experiments). Second, once a courier has picked-up a parcel, the associated delivery cannot be transferred to another route. Combined together, these two effects restrict the number of feasible "moves" in the local search procedure.

6. Conclusion

In this paper we have proposed an anticipatory mechanism for the dynamic *vehicle dispatching problem with pickups and deliveries*, which is faced by local area courier companies serving same-day pickup and delivery requests for the transport of letters and small parcels. The contribution of the paper to the literature is twofold. First, the number of demand samples has been determined through an IZS procedure that allows to eliminate, at an early stage of experimentation, those solutions that are clearly inferior. Second, we have limited the computational effort by applying the sampling procedure to a short-term horizon whose duration has been empirically correlated to the problem data. As a result, the computational effort has been kept low, while still achieving dramatic improvements (up to 69%) over a pure reactive procedure.

In this paper we have assumed that the requests arrive according to a known stochastic process. Future work should be aimed at verifying how robust our approach is whereas demand sampling is based on an approximation of the arrival process.

Acknowledgement

This research was partially supported by the Ministero dell'Istruzione, dell'Università e della Ricerca Scientifica (MIUR) of Italy and by Regione Puglia (Progetto Esplorativo e-FLEET). This support is gratefully acknowledged. Thanks are also due to Dr. Rosita Guido for her help with programming and to two anonymous referees for their useful comments, which helped us to improve the paper.

References

Bent, R., Van Hentenryck, P., 2004. Scenario-based planning for partially dynamic vehicle routing with stochastic customers. Operations Research 52, 977–987.
Bent, R., Van Hentenryck, P., 2007. Waiting and relocation strategies in online stochastic vehicle routing. In: Proceedings of the Twentieth International Joint Conference on Artifical Intelligence, pp. 1816–1821.
Bertsekas, D.P., Tsitsiklis, J.N., 1996. Neuro-dynamic programming. Athena Scientific.
Bertsimas, D.J., Van Ryzin, G., 1991. A stochastic and dynamic vehicle routing problem in the euclidean plane. Operations Research 39, 601–615.
Bertsimas, D.J., Van Ryzin, G., 1993. Stochastic and dynamic vehicle routing problem in the euclidean plane with multiple capacitated vehicles. Operations Research 41, 60–76.
Branke, J., Middendorf, M., Noeth, G., Dessouky, M., 2005. Waiting strategies for dynamic vehicle routing. Transportation Science 39, 298–312.
Fu, M., 2002. Optimization for simulation: theory vs. practice. Informs Journal on Computing 14, 192–215.
Gendreau, M., Potvin, J.-Y., 1998. Dynamic routing and dispatching. In: Crainic, T.G., Laporte, G. (Eds.), Fleet Management and Logistics. Kluwer, pp. 115–126.
Gendreau, M., Guerten, F., Potvin, J.-Y., Séguin, R., 2006. Neighborhood search heuristics for a dynamic vehicle dispatching problem with pick-ups and deliveries. Transportation Research Part C 14, 157–174.
Ghiani, G., Guerriero, F., Laporte, G., Musmanno, R., 2003. Real-time vehicle routing: solution concepts, algorithms and parallel strategies. European Journal of Operational Research 151, 1–11.
Godfrey, G., Powell, W.B., 2002. An adaptive dynamic programming algorithm for stochastic resource allocation problems I: single period travel times. Transportation Science 36, 21–39.
Ichoua, S., Gendreau, M., Potvin, J.-Y., 2000. Diversion issues in real-time vehicle dispatching. Transportation Science 34, 426–438.
Ichoua, S., Gendreau, M., Potvin, J.-Y., 2006. Exploiting knowledge about future demands for real-time vehicle dispatching. Transportation Science 40, 211–225.
Kim, S.-H., Nelson, B.L., 2001. A fully sequential procedure for indifference-zone selection in simulation. ACM Transactions on Modeling and Computer Simulation 11, 251–273.
Larsen, A., 2000. The Dynamic Vehicle Routing Problem. PhD thesis, Technical University of Denmark.
Larsen, A., Madsen, O.B.G., Solomon, M.M., 2002. Partially dynamic vehicle routing – models and algorithms. Journal of the Operational Research Society 53, 637–646.
Larsen, A., Madsen, O.B.G., Solomon, M.M., 2008. Recent developments in dynamic vehicle routing systems. In: Golden, B., Raghavan, R., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges. North-Holland, pp. 199–218.

Mitrović-Minić, S., Laporte, G., 2004. Waiting strategies for the dynamic pickup and delivery problem with time windows. Transportation Research Part B 38, 635–655.
Mitrović-Minić, S., Krishnamurti, R., Laporte, G., 2004. Double-horizon based heurisitcs for the dynamic pickup and delivery problem with time windows. Transportation Research Part B 38, 669–685.
Papastvrou, J.D., 1996. A stochastic and dynamic routing policy using branching processes with state dependent migration. European Journal of Operational Research 95, 167–177.
Powell, W.B., Topaloglu, H., 2003. Stochastic programming in transportation and logistics. In: Ruszczynski, A., Shapiro, A. (Eds.), Handbooks in Operations Research and Management Sciences: Stochastic Programming. Elsevier, pp. 555–635.
Powell, W.B., Sheffi, Y., Nickerson, K.S., 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.
Psaraftis, H.N., 1995. Dynamic vehicle routing: status and prospects. Annals of Operations Research 61, 143–164.
Regan, A.C., Mahmassani, H.S., Jaillet, P., 1994. Improving efficiency of commercial vehicle operations using real-time information: potential uses and assignment strategies. Transportation Research Record 1493, 188–198.
Regan, A.C., Mahmassani, H.S., Jaillet, P., 1995. Dynamic decision making for commercial fleet operations using real-time information. Transportation Research Record 1537, 91–97.
Secomandi, N., 2001. A rollout policy for the vehicle routing problem with stochastic demands. Operations Research 49, 796–802.
Spivey, M., Powell, W.B., 2004. The dynamic assignment problem. Transportation Science 38, 399–419.
Stefik, M., 1981. Planning with constraints (molgen: Part 1). Artificial Intelligence 16, 111–139.
Swihart, M.R., Papastavrou, J.D., 1999. A stochastic and dynamic model for the single-vehicle pick-up and delivery problem. European Journal of Operational Research 114, 447–464.
Thomas, B.W., 2007. Waiting strategies for anticipating service requests from known customer locations. Transportation Science 41, 319–331.
Thomas, B.W., White III, C.C., 2004. Anticipatory route selection. Transportation Science 38, 473–487.
van Hemert, J.I., La Poutré, J.A., 2004. Dynamic routing with fruitful regions: models and evolutionary computation. In: Yao, X., Burke, E., Lozano, J.A., Smith, J., Merelo-Guervós, J.J., Bullinaria, J.A., Rowe, J., Tino, P., Kabáan, A., Schwefel, H.-P. (Eds.), Parallel Problem Solving from Nature, vol. 8. Springer-Verlag, pp. 690–699.