Short Term Strategies for a Dynamic Multi-Period Routing Problem

E. Angelelli[*], *N. Bianchessi*[*], *R. Mansini*[§], *M.G. Speranza*[*]

[*] *Dipartimento Metodi Quantitativi, University of Brescia, Italy*

[§] *Dipartimento di Elettronica per l'Automazione, University of Brescia, Italy*

Abstract

We consider a Dynamic Multi-Period Routing Problem (DMPRP) faced by a company which deals with on-line pick-up requests and has to serve them by a fleet of uncapacitated vehicles over a finite time horizon. When a request is issued, a deadline of a given number of days $d \leq 2$ is associated to it: if $d = 1$ the request has to be satisfied on the same day (unpostponable request) while if $d = 2$ the request may be served either on the same day or on the day after (postponable request). At the beginning of each day some requests are already known, while others may arrive as time goes on. Every day the company faces on-line requests by possibly making new plans for the service and decides whether or not to serve postponable requests without knowing the set of new requests that will be issued the day after. The company objective is to satisfy all the received requests while minimizing the average operational costs per day. The daily cost includes a very high cost paid for each request forwarded to a back-up service. We propose different short term routing strategies and analyze their impact on the long term objective. Extensive computational results are provided on randomly generated instances simulating different real case scenarios and conclusions are drawn on the effectiveness of the strategies.

Keywords: Dynamic Multi-Period Routing Problems, Postponable requests, Short term strategies.

Introduction

We focus on a dynamic transportation problem typically faced by a company which deals with on-line pick-up requests. There is a central depot which provides the fleet of vehicles for serving the requests. Requests are satisfied by carrying parcels from a specific location to the depot. When a request is issued, a deadline is associated to it. On the basis of the urgency of a request and the service tariffs the customer chooses its deadline.

Every morning the vehicles leave the depot and have to return to the depot at the end of the day. At the beginning of each day some requests are already known (*off-line requests*), while others may arrive during the day (*on-line requests*). Moreover, some of them may be postponed to a later day (*postponable requests*) while others are not (*unpostponable requests*). Thus a request is on-line on the day it is issued by the customer, while if it is not served on the

same day it becomes off-line. Thanks to modern communication technology, the company knows the exact position of the vehicles at any time instant and is able to forecast their position in the near future. The company can react to on-line requests and possibly modify the previous traveling plans. Actually the assignment of a pick-up request to a vehicle may be changed until the very last moment before the service takes place.

The objective of the company is to guarantee the accomplishment of all the received requests while minimizing the average operational cost per day. Since issued requests are never rejected, in the case the company is not able to guarantee the service to all received requests, it has the opportunity to forward some of them, at a high cost, to a backup service.

In this paper we assume that the parcel size is not relevant and, for this reason, the fleet of vehicles is assumed to be uncapacitated. Moreover, each request has to be served within $d \leq 2$ days. If $d = 1$ the request is unpostponable and has to be served on the same day, whereas if $d = 2$ the request is postponable and may be served on the same day or on the day after. We call this problem the Dynamic Multi-Period Routing Problem (DMPRP). If all the requests to be served in the future would be known in advance, the DMPRP would reduce to a special case of the Periodic Vehicle Routing Problem (PVRP) (see Cordeau et al.[5]). We believe that many of the ideas presented in this paper can be extended to more general cases.

Dynamic routing problems have been attracting the interest of many researchers in the last years (see the surveys by Psaraftis [14], [15] and Ghiani et al. [8]). According to the existing literature, there exist several important problems that have to be solved in real-time. They are usually classified according to the application that motivates the work. There are problems dealing with dynamic fleet management (see [16] and [18]), problems analyzing the long-distance courier service (see, for instance, [7], [9], [11] and [12]), problems facing dial-a-ride systems (see [10] and [6], where an interesting and effective objective function is used for the case of a service to handicapped and elderly people).

With respect to the above cited dynamic problems, the DMPRP we study in this paper is characterized by some distinctive features. First of all, the issues typically addressed in the literature concern dynamic problems where one day only is considered and requests have all to be served within the end of the day. The requests are divided into off-line requests when they are known in advance (as the loads coming from remote terminal and to be distributed locally (see, for example, [7] and [9]) or as the trip booked one day in advance in dial-a-ride problems (see, for example, [6] and [10])) and on-line requests when they come over in real-time, while the vehicles are traveling. In such dynamic contexts the main concern of the decision-maker is to decide whether on-line requests should be accepted or rejected. An on-line request is accepted only if it can be inserted in the existing service plan, otherwise it is rejected. In our case requests can be served within two days. Thus, the requests can be either off-line or on-line but are also classified as postponable or unpostponable. If a request is unpostponable, it has to be inserted in the currently traveled routes, while if it is postponable it may be, if beneficial, postponed to the following day. The possibility to postpone the service opens new research issues. The only known dynamic multi-period routing problem characterized by both customers to be served today and customers that can be served either today or tomorrow (postponable customers) is that analyzed in Angelelli et al. [2] and [3], where

the authors provide the competitive analysis of some algorithms in a simple case.

Another distinctive feature with respect to the literature is the approach used to tackle the dynamic problem. The most common approach is based on a repeated re-optimization of various instances of the off-line problem. We propose, instead, *short term routing strategies* characterized by a look-ahead period and a short term objective. For each strategy, we define a corresponding optimization problem and provide a computational analysis of its impact on the long term objective of the problem. We propose a Variable Neighborhood Search (VNS) heuristic for the solution of the optimization problems. The running time is limited by the need to have answers within a time compatible with the real-time environment.

In order to understand the value of the proposed strategies, we compare their efficiency and effectiveness with a naive method that mimics the behavior of a decision-maker with no optimization tool at hand. The method immediately inserts new requests as soon as they become available by using a cheapest insertion procedure.

The paper is organized as follows. In Section 1 the proposed Dynamic Multi-Period Routing Problem is described and all its relevant features are put in evidence. In this section we also provide the general definition of *Short Term Strategy* (STS) and the description of the solution framework used to tackle the dynamic problem. In Section 2 we propose several STSs and describe the re-optimization problem which is a relevant part of the strategies. Section 3 is devoted to the description of the proposed Variable Neighborhood Search. Finally, in Section 4, the simple method used as a comparative approach is described and extensive computational results are provided on randomly generated instances simulating different real case scenarios and some conclusions are drawn on the effectiveness of each short term strategy over all the tested scenarios. Finally, some future developments are sketched.

1 The Dynamic Multi-Period Problem

The proposed Dynamic Multi-Period Routing Problem (DMPRP) is defined as follows.

A set of pick-up requests need to be served over a time horizon of T days. Each day a fleet of uncapacitated vehicles $V = \{v_1, \ldots, v_m\}$ is available for the service. Every day the vehicles leave the depot in the morning and have to return to the depot at the end of the day. The duration of a day is denoted by τ which is also the maximum length in time of each vehicle route. We will refer to the length of a route by understanding that the length is a time length.

At the beginning of each day some of such requests are already known, while others may arrive over time. The traveling plans can be modified during the day to take into account the new requests. This can happen also while the vehicle is traveling between two destinations, i.e. we allow *diversion*.

Each request has a deadline of $d \leq 2$ days which makes it either *unpostponable* ($d = 1$) or *postponable* ($d = 2$). With respect to a fixed day in the time horizon we define as *off-line* all the requests that are known in the morning before the vehicles leave the depot, while we define as *on-line* all the requests that arrive during the day while the vehicles are traveling. For instance, a new

request which arrives today with deadline $d = 2$ is on-line and postponable. Off-line requests available at the beginning of each day consist of pick-up requests issued the day before. All such off-line requests are unpostponable and have to be served within the day.

Each day the company has to decide whether or not to serve postponable requests. The decision has to be taken without knowing what the set of new requests will be issued in the future. The company never rejects a new request. According to a common practice, we assume that no unpostponable pick-up request is accepted after a fixed time L (e.g. noon or 1:00 pm). This rule has the consequence that at that time the company knows the set of unpostponable requests and can decide which of them it is able to serve with its fleet and which it is forced to forward, at a high cost, to a back-up service. The goal is the minimization of the total service cost over the whole horizon. We have formalized such major target through two hierarchical objectives. The first one is the maximization of the number of requests directly served by the company, which is equivalent to the minimization of the number of requests forwarded to the backup service. The second one is the minimization of the length of the routes traveled.

1.1 The short term framework

In this section we provide the general description of a solution strategy. The common approach used to solve a dynamic problem is based on the repeated solution of its off-line version when new information becomes available and on the application of such solution until a new re-optimization is run. While widely accepted, it is not guaranteed that such an approach will lead to good solutions in the long term. For this reason, the crucial questions which need to be addressed when solving a dynamic problem are related to what off-line problem should be dynamically solved, if any, in order to achieve the long term objectives. Should we optimize by taking as objective the long term objective? If not, what is the right objective to optimize? Should we forget about optimization and seek for other goals like feasibility? How often is it beneficial to re-optimize?

All such questions find an answer through the definition of a solution strategy. To tackle the long term problem we have analyzed some *Short Term Strategies* (STSs), each of which consists in the following main components:

1. *A look-ahead period*: The period of time over which the re-optimization problem will be defined.

2. *A short term objective*: The criterion used to evaluate the quality of a solution in the re-optimization problem.

3. *A re-optimization problem*: The off-line problem which is formulated and solved, after a look-ahead period and a short term objective have been defined.

4. *A re-optimization interval*. The length of the time interval between the solution of two consecutive re-optimization problems.

When we choose the look-ahead period, we can, for example, consider the next hours or the remaining part of the current day or a period which may also include the day after. A too long period may cause problems because of the

lack of information on the future, while a too short period may not allow us to fully exploit the availability of the vehicles.

The second component is the criterion to measure the quality of a solution, that is the objective function of the re-optimization problem. On one hand the number of served requests is important, on the other hand the length of the routes cannot be forgotten. The objective function selected in the short term is strictly related to the length of the look-ahead period and may strongly affect the quality of the solution obtainable in the long term. If the selected look-ahead period is the current day only and we decide to maximize the number of requests served we may end up with postponing all farthest requests and making the satisfaction of all requests in the future a hard task. On the other hand, if we minimize the length of the routes traveled today we may end up with postponing as many requests as possible. Similarly, bad solutions can be figured out by considering a two days look-ahead horizon along with the two described objectives.

Once the first two components of the strategy have been chosen, we can define the re-optimization problem, i.e. the off-line problem which will be solved over time. The solution of the re-optimization problem is obtained by means of a heuristic algorithm and aims at providing new routes for the vehicles possibly including some of the new requests that have arrived since the last re-optimization.

Finally, the last component is the frequency of the re-optimization, i.e. the time interval which elapses between the solution of two consecutive re-optimization problems. While it is clear that updating the routes on the basis of the new information may be locally beneficial, a large number of re-optimizations might imply an instability effect and have negative conseguences on the long term objective.

1.2 The long term framework

A Short Term Strategy is implemented through the solution of the corresponding re-optimization problem. Each day the re-optimization problem is solved once just before the beginning of the day and then continuously over time at constant time intervals (re-optimization intervals). The problem solved before the beginning of the day only considers unpostponable requests and provides for each vehicle a route that starts and ends at the depot. The subsequent re-optimization problems take into account all known requests (postponable and unpostponable) and provide for each vehicle a route that starts at the forecasted position of the vehicle at the end of the re-optimization according to the previously planned routes and ends at the depot.

Let us provide some notation. For sake of simplicity, we assume that each day the time evolution of the dynamic system is indexed by a continuous variable $t \in [0, \tau]$, where τ is the length of the working day. Initially, at time $t = 0$, all vehicles are idle at the central depot. Request locations, as well as vehicle positions at any given time $t \geq 0$, are assumed to be points in a bounded region of a metric space. We indicate by $R^P(t)$ and $R^U(t)$ the set of postponable and unpostponable requests at a given time t, respectively. We also denote by $R(t) = R^P(t) \cup R^U(t)$ the total set of the known requests at time t. Let Δt be the length of the re-optimization interval and let $t' = t + \Delta t$. The set $R(t')$ differs from $R(t)$ for the inclusion of all the new requests which have become available during the last re-optimization interval Δt and for the elimination of

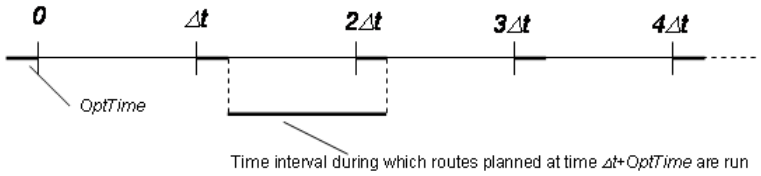

Figure 1: The long term framework: Temporal distribution of the re-optimization problem instances.

all the requests served in the meantime.

A maximum time $OptTime$ (for Optimization Time) is made available to the algorithm that solves each re-optimization problem. The solution found by a re-optimization is implemented until the end of the next re-optimization. More precisely, if a re-optimization has taken place at time t, the generated routes are followed by the vehicles from time $t + OptTime$ to time $t' + OptTime$ where $t' = t + \Delta t$, that is until the routes obtained with the subsequent re-optimization have become available. Figure 1 provides the temporal description of the proposed framework. Since we have assumed that, at the beginning of each day, time is measured starting with time 0, the problem that has to be solved in the morning before all the vehicles start their tours is assumed to be solved at time $-OptTime$.

2 The Short Term Strategies

We introduce two classes of Short Term Strategies. The first one, called *1-day look-ahead(f)*, is characterized by a look-ahead period of only one day, while the second one, named *2-day look-ahead(f)*, by a look-ahead period of two days. A strategy belonging to the former class "works" on the routes of the current day while a strategy of the latter class on the routes of the current day and of the day after. Moreover, a strategy differs from the others of the same class for the function f selected as objective function. Different re-optimization intervals will be tested.

2.1 The short term objectives

Let r_1^P and r_1^U represent the number of postponable and unpostponable requests served today, respectively. Moreover, let r_1 and l_1 denote the total number of served requests and the total length of the routes traveled in the current day, respectively. The objective functions f we consider for the *1-day look-ahead(f)* class are the following:

$$min \quad f_1^1 = \quad l_1 + r_1^U K \tag{1}$$

$$min \quad f_1^2 = \quad l_1 + r_1^P M + r_1^U K \tag{2}$$

$$min \quad f_1^3 = \quad \frac{l_1}{r_1} + r_1^U K \tag{3}$$

where M and K are negative constant values such that $K \ll M \ll 0$. According to the notation used to indicate the objective function, the lower index refers to the look-ahead period while the upper index to the objective function.

We recall that the hierarchical objectives of the DMPRP are the maximization of the number of served requests and the minimization of the length of the traveled routes. To achieve such hierarchical objectives, the objective functions of the class *1-day look-ahead(f)* first maximize the number of unpostponable served requests, through the term $r_1^U K$ shared by all functions. Then, each function differs from the others for the second hierarchical objective achieved: The minimization of the length l_1 of the routes traveled in the current day (function f_1^1), the maximization of the number of postponable requests served $r_1^P M$ and the minimization of the length l_1 of the routes (function f_1^2), the minimization of the average distance traveled by the vehicles to serve a request l_1/r_1 (function f_1^3). Function f_1^1 is consistent with the objectives of the long term problem. However, due to the horizon of the strategy, the minimization of the length of the routes of the current day implies that the service of all postponable requests will be postponed to the day after. In order to overcome this limit and to avoid problems for the day after, function f_1^2 maximizes the number of postponable requests served in the current day and then minimizes the length of the routes. This creates the opportunity to serve more requests the day after. However, function f_1^2 anticipates to the current day whatever request can be anticipated, independently of its location. This is the reason for the introduction of function f_1^3 that minimizes the average distance per request traveled by a vehicle.

Given two solutions and an objective function, the best of the two is identified avoiding the impact of specific values of the parameters. Actually, in the case of function f_1^1 for example, the best of the two solutions is the one that has the minimum value of r_1^U and, in case of tightness, the one that has the minimum value of l_1. Similarly for the other functions.

When the routes for a certain day are planned, let r_2 and l_2 denote the number of served requests and the total length of the routes traveled the day after, respectively. The objective functions f we consider for the *2-day look-ahead(f)* strategies are the following:

$$min \quad f_2^1 = \quad \alpha l_1 + (1-\alpha) l_2 + (r_1^P + r_2) K_2 + r_1^U K_1 \tag{4}$$

$$min \quad f_2^2 = \quad \alpha l_1 + (1-\alpha) l_2 + r_1^P M + (r_1^P + r_2) K_2 + r_1^U K_1 \tag{5}$$

$$min \quad f_2^3 = \quad \alpha \frac{l_1}{r_1} + (1-\alpha) \frac{l_2}{r_2} + (r_1^P + r_2) K_2 + r_1^U K_1 \tag{6}$$

where α is a real number such that $0 \leq \alpha \leq 1$ and M, K_1, K_2 are negative constant values such that $K_1 \ll K_2 \ll M \ll 0$.

In the *2-day look-ahead(f)* strategies the three proposed objective functions share the objective to maximize the number of unpostponable served requests (term $r_1^U K_1$) and, as second hierarchical objective, to maximize the total number of postponable requests to be served within the day after (term $(r_1^P + r_2)K_2$). Actually, the requests that are postponable today will become unpostponable tomorrow and have to be served within tomorrow. The three objective functions differ for the remaining objective to be achieved in a way similar to the functions of the *1-day look-ahead(f)* strategies.

It is reasonable to assume $\alpha > 1/2$, i.e. that the length of the today routes

is more relevant to the today decisions than the length of the routes of the day after. If α is close to 1, function f_2^1 tends to postpone to the day after the postponable requests. Function f_2^2 anticipates to the current day the postponable requests and then optimizes the routes. Finally, function f_2^3 finds a compromise between postponing and anticipating the postponable requests.

2.2 The re-optimization problem

The definition of a re-optimization problem is crucial to the success of a strategy. We recall that τ is the maximum length in time of a route. At any re-optimization time $t \in [-OptTime, \tau - OptTime]$, a directed graph $G_t = (N_t, A_t)$ is defined. The set of vertices N_t represent positions. The vertex 0 represents the position of the depot. Vertices $1, \ldots, m$ represent the forecasted positions of vehicles $v_1, \ldots, v_m$ at time $t + OptTime$. Finally, vertices $m+1, \ldots, m+n$, represent the positions of the requests that are known at time t and we forecast will not yet be served at time $t + OptTime$. A nonnegative value t_{ij} is associated to each arc $(i, j) \in A_t$ and indicates the travel time from vertex i to vertex j.

The re-optimization problem for a 2-day look-ahead(f) strategy defines, for each vehicle v_k, $k \in \{1, \ldots, m\}$, the route for the current day s_1^k and the route for the day after s_2^k which optimize the objective function f. The re-optimization problem for a 1-day look-ahead(f) strategy defines the routes of the vehicles for the current day only. Let $S_1 = \{s_1^k \mid k \in \{1, \ldots, m\}\}$ and $S_2 = \{s_d^k \mid k \in \{1, \ldots, m\}, d \in \{1, 2\}\}$ be the set of routes representing the feasible solution of a re-optimization problem of a 1-day look-ahead(f) and of a 2-day look-ahead(f) strategy, respectively.

At a re-optimization time t, each route s_1^k or s_2^k is defined as a sequence of vertices over the graph $G_t = (N_t, A_t)$. A solution is feasible if it satisfies the following operational constraints:

1. a route s_1^k, assigned to the vehicle v_k, $k \in \{1, \ldots, m\}$, must originate in vertex k and terminate in vertex 0, while a route s_2^k must originate and terminate in vertex 0;

2. a route s_2^k, assigned to the vehicle v_k, $k \in \{1, \ldots, m\}$, has to serve postponable requests only;

3. the daily traveling time of each vehicle cannot be greater than τ. Thus, if $t > 0$, the length of the routes s_1^k and s_2^k assigned to the vehicle v_k, $k \in \{1, \ldots, m\}$, must not exceed the time limits $\tau - (t + OptTime)$ and τ, respectively, while, if $t = 0$, the length of both routes must not exceed τ.

3 A Variable Neighborhood Search heuristic

A method for the solution of a re-optimization problem can update a current solution (set of routes) with respect to the previous one in different ways. A postponable request that was planned to be served today may be postponed to tomorrow and viceversa, a just arrived on-line request may be included in a route, a request assigned to a vehicle may be assigned to a different one or a set of requests not visited may be inserted in new planned routes by substituting existing requests.

The computational time $OptTime$ made available for the solution of the re-optimization problem depends on the re-optimization frequency and on the desired time set by the decision-maker to react to new information. According to practice, this time is usually quite short. For this reason, we present a Variable Neighborhood Search (VNS) heuristic that can solve any re-optimization problem within a short computational time. At each re-optimization time, the heuristic starts from an initial solution S (i.e. the part not yet implemented of the solution obtained at the previous re-optimization) and tries to improve it. The quality of a solution is measured through the short-term objective of the problem.

Initial solution

Let $t \neq -OptTime$ be the current re-optimization time and let S^- be the solution found at time $t - \Delta t$. Depending on the re-optimization problem to be solved, such solution will include routes for the current day only or also routes for the day after. At time $t + OptTime$ there may be some routes s_1^k, $k \in \{1, \ldots, m\}$, belonging to S^-, which have been completed and some others that still have to be completed (residual routes).

The initial solution S includes all the residual routes and the routes $\{s_2^k | k = 1, ..., m\}$ built at time $t - \Delta t$ in the case of a *2-day look-ahead(f)* re-optimization problem. It may happen that $S = \emptyset$, like for example at the re-optimization time $t = -OptTime$. In this case the VNS heuristic builds a solution from scratch.

Variable Neighborhood Search heuristic (*DynaRoute*)

Let us assume a local search (LS) heuristic is available. A LS heuristic may get stuck in a local optimum. In order to escape from it, a solution can be selected in a neighborhood of the current local optimum and the local search be started again. If the solution is too close to the current local optimum, the LS heuristic is likely to get back to the same local optimum. On the other hand, if the solution is too far, the search could move away from a promising area. The goal of a VNS heuristic is to provide a sequence of neighborhoods $N_h(\cdot)$, each one characterized by a different radius h. If intensification of the search is required, the initial radius will be set to a minimum value so that the closest neighborhood is selected first. The radius will be increased only if no solution improvement can be obtained. If diversification of the search is required, the initial radius will be set to a maximum value and then decreased. The variable neighborhood search scheme was first proposed by Mladenovic and Hansen in [13] and has been effectively applied to many routing problems.

The VNS heuristic we propose is called *DynaRoute* and is initialized with the minimum value 0 of the radius. According to the scheme presented in [13], a solution S' is randomly selected in the neighborhood $N_h(S)$ of the current solution S. Then, a LS heuristic, which we call *DynaSearch* and will be explained later, is applied to S'. If the local search *DynaSearch* ends without a solution better than S, a further attempt is done by selecting at random a new neighbor S'. After p_{max} unsuccessful attempts, the radius h is increased and the process repeated in the new neighborhood. If the *DynaSearch* succeeds, the current solution S is updated with S', the radius h is set to 1 and the algorithm is restarted. The *DynaRoute* heuristic stops when no better solution can be found in the neighborhood of maximum radius h_{max} or when the elapsed execution time exceeds the time available for the re-optimization process $OptTime$.

Let $\widetilde{R} = \widetilde{R}^P \cup \widetilde{R}^U$ be the set of all requests not yet scheduled for the service, where $\widetilde{R}^P$ and $\widetilde{R}^U$ are the postponable and the unpostponable requests, respectively. Moreover, let R^S be the set of requests scheduled for the service in the current solution S. The solution S' in the neighborhood $N_h(S)$ is obtained by randomly removing h requests from the routes $s_1^k \in S$, $k \in \{1, \ldots, m\}$, and, in the case of a *2-day look-ahead(f)* strategy, also $2h$ requests from the routes $s_2^k \in S$, $k \in \{1, \ldots, m\}$. The extracted requests are then labeled as tabu and inserted in $\widetilde{R}$. Then, the solution S' is enhanced by means of *DynaSearch*.

The heuristic *DynaRoute* can be described as follows.

DynaRoute$(S, \widetilde{R})$
p_{max}:= maximum number of trials;
h_{max}:= maximum value of the radius;

$S^* := S$; $\widetilde{R}^* := \widetilde{R}$;
$h := 0$; $p := 1$;

$InitTime := clock()$; /* $clock()$ returns the current time in seconds */

while $((h \leq h_{max})$ **and** $((clock() - InitTime) \leq OptTime))$
/* Neighbor selection */
Randomly select a solution S' in the neighborhood $N_h(S^*)$;
Label the extracted requests as tabu and insert them in $\widetilde{R}$;

/* Neighbor enhancement */
$S'' :=$*DynaSearch*$(S', \widetilde{R})$;
Label all the requests in $\widetilde{R}$ as non-tabu;

if $(f(S'') < f(S^*))$ **then**
$S^* := S''$; $\widetilde{R}^* := \widetilde{R}$;
$h := 1$; $p := 1$;
else
$\widetilde{R} := \widetilde{R}^*$;
if $(p < p_{max}$ and $h > 0)$ **then** $p := p + 1$;
else $p := 1$; $h := h + 1$;
end-while

return S^*

The local search heuristic *DynaSearch* explores a sequence of three disjoint neighborhoods (see [4]). At each step of the local search one of the three disjoint neighborhoods is completely explored. If an improving solution is found, then the current solution is updated with the best solution found and the search starts over again. Otherwise, the next neighborhood is explored. In the case no improving solution can be found in any neighborhood the local search terminates. The order in which the neighborhoods are considered is fixed and at each iteration the neighborhoods list is scanned from the beginning. The neighborhoods employed are:

- INSERT: This neighborhood includes all feasible solutions that can be obtained by inserting a not served unpostponable request in a route $s_1^k \in S'$ or a not served postponable request in the routes $s_1^k, s_2^k \in S'$. A new solution is generated for each not served request and for each feasible

insertion position. A not served request can be inserted into a route serving n requests in, at most, $n+1$ feasible positions. If the n requests are spread over m routes, the possible insertion positions become $n+m$. The number of considered solutions is, thus, $O(|\widetilde{R}|(r_1+m))$ and $O(|\widetilde{R}|(r_1+m)+|\widetilde{R}^P|(r_2+m))$ in the case of a *1-day look-ahead(f)* and *2-day look-ahead(f)* problem, respectively.

- RELOCATE: This neighborhood includes all feasible solutions which can be obtained by deleting an unpostponable request from a route $s_1^k \in S'$ and re-inserting it into a route $s_1^p \in S'$, or by deleting a postponable request from a route s_1^k or $s_2^k \in S'$, and re-inserting it into a route s_1^p or s_2^p of S'. A request may be relocated in the route from which it has been deleted if inserted in a different position. The request to be relocated can also be chosen in a route which only contains such request. Moreover, if the number of existing routes is less than m, a request that has to be relocated can also be assigned to a new route that will be inserted in S'. A new solution is considered for each request belonging to a route and for each feasible insertion position. The number of considered solutions in the case of a *1-day look-ahead(f)* strategy is $O(r_1((r_1-1)+m))$, while for the case of a *2-day look-ahead(f)* strategy the number of evaluated solutions is $O(r_1((r_1-1)+m)+r_1^P(r_2+m)+r_2((r_2-1)+m)+r_2(r_1+m))$.

- EXCHANGE: This neighborhood includes all feasible solutions that can be obtained by exchanging an unpostponable request of a route $s_1^k \in S'$ with a not served unpostponable request, or a postponable request of a route $s_1^k \in S'$ with a not served request, or a request of a route $s_2^k \in S'$ with a not served postponable request. Finally, the neighborhood includes also all the solutions that can be obtained by exchanging a postponable request of a route $s_1^k \in S'$ with a postponable request of a route $s_2^k \in S'$. During the exchange, each of the requests has to be inserted into the same route and in the same position from which the other has been removed. The number of considered solutions is, thus, $O(|\widetilde{R}^U|r_1+|\widetilde{R}^P|r_1^P)$ and $O(|\widetilde{R}^U|r_1+|\widetilde{R}^P|(r_1^P+r_2)+r_1^P r_2)$ in the case of a *1-day look-ahead(f)* and *2-day look-ahead(f)* problem, respectively.

The neighborhoods RELOCATE and EXCHANGE are inspired by neighborhoods RELOCATE and XSTRONG presented in [4]. According to some preliminary results, we have decided to sort the three neighborhoods by considering first the neighborhood INSERT and then the neighborhoods RELOCATE and EXCHANGE.

The *DynaSearch* heuristic is executed twice. The first time, in order to avoid cyclic moves, the heuristic is executed by only considering the non-tabu postponable and unpostponable requests included in $\widetilde{R}$. The second time, the heuristic is executed over all the remaining requests included in $\widetilde{R}$, searching for improving solutions by possibly including some of the previously excluded tabu requests. After the execution of the *DynaSearch* heuristic, the tabu status of all the requests included in $\widetilde{R}$ is reset.

Notice that, by setting $h_{max}=0$, the *DynaRoute* heuristic reduces to the *DynaSearch* heuristic.

4 Computational results

The computational analysis has been carried out on two classes of scenarios that differ for the geographical dispersion of the requests over a service area of $100 \times 100\ km^2$ in the Euclidean plane. The distance between any two points is the Euclidean distance. In the first class (*random scenarios*) the requests are uniformly distributed, while in the second one (*cluster scenarios*) the requests are clustered. Each scenario is characterized by a planning horizon of $T = 10$ days and a daily working service period of length $\tau = 10$ hours (from 8:00 AM to 6:00 PM). Requests are generated dynamically according to a Poisson distribution with parameter λ. With a probability equal to $\frac{1}{3}$, requests arriving before 1:00 PM are considered as unpostponable. We consider 5 different values for the Poisson distribution parameter $\lambda \in \{100, 200, 300, 400, 500\}$. This means 5 scenarios for each class (one for each value of λ) for a total number of 10 scenarios. Then, we generate 3 different instances for each scenario for a total number of 30 instances. Each instance differs from the others for the number of daily requests and for the coordinates of such requests. In a random scenario instance, whenever a request arrives its coordinates are randomly selected among all those of the requests in problem sets r1 and r2 of the Solomon's instances for the VRPTW (see [17]). In all the instances of the random scenarios, the coordinates of the depot are those of the Solomon's instances. Instances of cluster scenarios are similarly generated using sets c1 and c2 of the Solomon's instances. The service is supposed to be provided by means of a fleet of 3 vehicles, each of them traveling at a constant speed of 40 km/h.

We have tested 18 different strategies obtained by combining three different values for the re-optimization interval Δt equal to 5, 2.5 and 1 hours along with 3 re-optimization problems with a look-ahead period of 1 day and 3 re-optimization problems with a look-ahead period of 2 days. In order to make all possible strategies comparable, an additional day is considered to complete the work of the not yet served requests. In fact, a strategy may take improper advantage by postponing as many requests as possible from day T to day $T+1$. The parameters of $DynaRoute(S, \widetilde{R})$ are set as follows: $OptTime$ has been set equal to $\frac{1}{12}\Delta t$, while h_{max} and p_{max} have been set equal to $\lfloor\sqrt{|R^S|}\rfloor$ and $\lfloor\frac{h_{max}}{10}\rfloor$, respectively.

All computational experiments have been carried out on a 1.5GHz Intel Pentium IV machine with 512MB of RAM. The results presented in the following are the average values of the results obtained out of the 3 instances associated with each scenario.

4.1 1-day look-ahead(f) strategies

We have analyzed 9 different *1-day look-ahead(f)* strategies obtained by combining the three short term objectives with three different values of the re-optimization interval Δt.

Tables 1 and 2 report the results obtained by applying such strategies in the case of random and cluster scenarios, respectively.

Each table consists of a row for each strategy and as many columns as the tested values of the parameter λ plus the first two columns which provide the value of the re-optimization interval Δt and the short term objective function which defines the re-optimization problem. For each column associated to a

Δt	f	$\lambda = 100$		$\lambda = 200$		$\lambda = 300$		$\lambda = 400$		$\lambda = 500$	
		time	#	time	#	time	#	time	#	time	#
	f_1^1	$204^h15'15''$	0.0	$267^h49'57''$	5.7	$292^h49'20''$	34.7	$300^h44'42''$	97.3	$303^h26'18''$	147.0
5.0^h	f_1^2	$229^h53'4''$	0.0	$258^h57'20''$	15.7	$291^h9'35''$	32.7	$302^h58'50''$	82.7	$305^h14'30''$	141.7
	f_1^3	$210^h7'5''$	0.3	$262^h32'3''$	10.3	$291^h55'27''$	30.3	$299^h5'32''$	80.7	$303^h21'35''$	136.3
	f_1^1	$195^h55'38''$	0.0	$258^h37'17''$	0.3	$289^h47'12''$	8.3	$302^h16'3''$	27.7	$307^h25'28''$	62.3
2.5^h	f_1^2	$238^h56'21''$	0.0	$277^h52'57''$	1.7	$299^h28'11''$	16.0	$310^h36'60''$	38.0	$312^h40'60''$	72.0
	f_1^3	$193^h58'15''$	0.0	$254^h32'40''$	0.3	$287^h42'56''$	6.7	$304^h31'45''$	18.0	$310^h19'27''$	43.3
	f_1^1	$186^h50'26''$	0.0	$248^h25'5''$	0.0	$275^h50'34''$	0.7	$296^h55'34''$	4.7	$305^h1'15''$	21.7
1.0^h	f_1^2	$244^h5'33''$	0.0	$276^h17'56''$	0.0	$298^h28'24''$	3.3	$312^h30'10''$	6.3	$316^h13'21''$	20.3
	f_1^3	$185^h27'7''$	0.0	$237^h27'17''$	0.0	$277^h23'13''$	0.0	$292^h56'24''$	4.3	$307^h34'59''$	11.0

Table 1: *1-day look-ahead(f)* strategies: Random scenarios.

Δt	f	$\lambda = 100$		$\lambda = 200$		$\lambda = 300$		$\lambda = 400$		$\lambda = 500$	
		time	#	time	#	time	#	time	#	time	#
	f_1^1	$188^h12'55''$	0.0	$219^h59'44''$	0.3	$237^h36'39''$	4.3	$241^h41'58''$	25.3	$245^h1'28''$	49.0
5.0^h	f_1^2	$208^h49'43''$	0.0	$242^h20'31''$	1.0	$246^h29'3''$	13.3	$248^h45'56''$	30.3	$249^h6'55''$	67.0
	f_1^3	$190^h52'46''$	0.0	$229^h6'6''$	0.7	$240^h45'28''$	17.0	$245^h16'18''$	33.0	$246^h59'38''$	60.7
	f_1^1	$173^h9'2''$	0.0	$204^h31'18''$	0.0	$232^h39'9''$	0.0	$249^h6'40''$	2.0	$252^h29'3''$	6.0
2.5^h	f_1^2	$221^h2'47''$	0.0	$254^h17'50''$	0.0	$265^h51'49''$	2.0	$268^h54'19''$	4.0	$277^h8'34''$	9.7
	f_1^3	$176^h51'51''$	0.0	$219^h51'24''$	0.0	$237^h26'5''$	0.7	$246^h56'6''$	5.3	$258^h58'10''$	10.0
	f_1^1	$161^h21'59''$	0.0	$190^h15'52''$	0.0	$218^h0'23''$	0.0	$226^h57'35''$	0.0	$238^h11'6''$	0.0
1.0^h	f_1^2	$225^h26'46''$	0.0	$261^h36'46''$	0.0	$269^h35'12''$	0.0	$275^h40'19''$	0.0	$281^h10'7''$	0.0
	f_1^3	$162^h27'55''$	0.0	$195^h15'10''$	0.0	$211^h46'56''$	0.0	$227^h17'51''$	0.0	$240^h13'41''$	0.0

Table 2: *1-day look-ahead(f) strategies:* Cluster scenarios.

value of the parameter λ the total length of the routes measured as traveling time and the total number of not served requests are reported, where the total is taken over the planning horizon.

In case of random scenarios, for each value of λ, the strategies with higher frequency of the re-optimization (smaller re-optimization interval) provide better results in terms of not served requests. In the most difficult scenarios ($\lambda = 300, 400, 500$), moving from strategies with $\Delta t = 5h$ to strategies with $\Delta t = 2.5h$ improves on average the results by about 65%, whereas the average improvement is of about 80% if we consider strategies with $\Delta t = 1h$ instead of $\Delta t = 2.5h$. Smaller values of the re-optimization interval Δt, from half an hour to few minutes, have also been tested confirming this trend. Clearly, from a practical point of view, Δt cannot take values that are too small. Selecting strategies with a higher re-optimization frequency allows also, in most of the cases, to obtain better results with respect to the second hierarchical objective. In particular, this is true if we move from strategies with $\Delta t = 2.5h$ to strategies with $\Delta t = 1h$ and we consider functions f_1^1 and f_1^3. On the other hand, when function f_1^2 is employed, the total length of the routes traveled increases on average with the increase of the re-optimization frequency. This is partially due to the increase of the number of served requests, but it also depends on the fact that the strategies that use this function maximize the number of requests served. This objective generates solutions where vehicles travel in already visited areas in order to serve new requests as soon as they appear. The strategies that evaluate the solutions by means of the function f_1^3 are the best. This is especially true when $\Delta t = 1h$. In this case, for $\lambda = 500$, function f_1^3 reduces the number of not served requests by about 49% and 46% with respect to functions f_1^1 and f_1^2, respectively. This suggests to minimize first the distance traveled to serve unpostponable requests, and then to visit postponable requests that can be efficiently served along the routes already planned to serve the unpostponable ones.

The behavior of the 9 strategies in the case of cluster scenarios is similar. The results reported in Table 2 show that cluster scenarios are easier to handle than the random ones. Actually, the strategies based on a re-optimization interval of one hour guarantee the service of all the requests in every scenario, while in the case of random scenarios the best strategy is able to serve all the requests only for values of $\lambda \leq 300$. In cluster scenarios the best strategy is the one that evaluates solutions by means of the function f_1^1.

4.2 2-day look-ahead(f) strategies

In order to test the *2-day look-ahead(f)* strategies, we have considered three different values of the parameter α (set equal to 0.5, 0.75 and 1^-, where 1^- is a value slightly smaller than 1). For each value of α a different short term objective, and thus a different strategy, is generated. A value $\alpha = 0.5$ means that the same weight is associated to the distances traveled today and tomorrow, while with $\alpha = 0.75$ and $\alpha = 1^-$ a greater weight is associated to the distance traveled today. With $\alpha = 1^-$ a decrease in the distance traveled today is to be preferred to any decrease in the distance traveled tomorrow. Preliminary tests show that the re-optimization interval $\Delta t = 1h$ gives better results than those obtained with larger values of Δt. Smaller values of Δt, from half an hour to few minutes, have also been tested confirming this trend (see [1]). We report

f	α	$\lambda = 100$		$\lambda = 200$		$\lambda = 300$		$\lambda = 400$		$\lambda = 500$	
		time	#	time	#	time	#	time	#	time	#
f_2^1	0.50	$204^h27'3''$	0.0	$256^h5'18''$	0.0	$294^h25'43''$	2.3	$310^h38'28''$	8.0	$315^h8'27''$	17.0
	0.75	$178^h1'3''$	0.0	$236^h45'13''$	0.0	$272^h1'28''$	0.0	$295^h13'38''$	2.3	$306^h8'31''$	12.7
	1^-	$181^h41'13''$	0.0	$235^h1'60''$	0.0	$267^h14'3''$	0.0	$287^h0'29''$	2.0	$298^h36'8''$	6.7
f_2^2	0.50	$241^h18'39''$	0.0	$277^h22'47''$	0.0	$298^h45'1''$	2.0	$312^h34'15''$	7.7	$316^h32'30''$	23.7
	0.75	$243^h1'36''$	0.0	$275^h55'43''$	0.0	$299^h38'26''$	1.0	$313^h20'45''$	10.3	$315^h41'34''$	19.0
	1^-	$244^h7'11''$	0.0	$276^h17'56''$	0.0	$298^h28'24''$	3.3	$312^h30'10''$	6.3	$316^h13'21''$	20.3
f_2^3	0.50	$246^h52'13''$	0.0	$278^h52'60''$	0.0	$306^h51'52''$	1.1	$314^h21'12''$	5.3	$317^h0'56''$	17.7
	0.75	$237^h39'34''$	0.0	$273^h58'7''$	0.0	$300^h8'55''$	1.7	$311^h53'52''$	4.0	$314^h23'26''$	12.0
	1^-	$183^h19'38''$	0.0	$235^h14'30''$	0.0	$276^h51'14''$	1.7	$292^h18'28''$	0.7	$305^h28'42''$	10.3

Table 3: *2-day look-ahead(f)* strategies and $\Delta t = 1h$: Random scenarios.

f	α	$\lambda = 100$		$\lambda = 200$		$\lambda = 300$		$\lambda = 400$		$\lambda = 500$	
		time	#	time	#	time	#	time	#	time	#
f_2^1	0.50	$184^h29'12''$	0.0	$218^h1'4''$	0.0	$246^h54'31''$	0.0	$265^h35'52''$	0.0	$276^h4'28''$	0.0
	0.75	$161^h20'15''$	0.0	$196^h27'56''$	0.0	$220^h22'27''$	0.0	$236^h29'51''$	0.0	$249^h53'23''$	0.0
	1^-	$160^h31'58''$	0.0	$185^h40'20''$	0.0	$206^h5'6''$	0.0	$217^h30'51''$	0.0	$231^h16'8''$	0.0
f_2^2	0.50	$225^h31'42''$	0.0	$262^h24'3''$	0.0	$269^h48'8''$	0.0	$278^h58'18''$	0.0	$281^h0'59''$	0.0
	0.75	$225^h59'25''$	0.0	$260^h10'29''$	0.0	$269^h55'46''$	0.0	$275^h26'55''$	0.0	$279^h6'46''$	0.0
	1^-	$226^h6'24''$	0.0	$261^h43'2''$	0.0	$269^h35'12''$	0.0	$276^h5'30''$	0.0	$280^h2'30''$	0.0
f_2^3	0.50	$228^h26'7''$	0.0	$258^h30'55''$	0.0	$268^h29'19''$	0.0	$276^h47'31''$	0.0	$282^h2'44''$	0.0
	0.75	$223^h29'40''$	0.0	$255^h13'4''$	0.0	$264^h33'51''$	0.0	$272^h30'25''$	0.0	$276^h54'59''$	0.0
	1^-	$161^h4'12''$	0.0	$190^h11'29''$	0.0	$213^h58'17''$	0.0	$225^h39'58''$	0.0	$237^h9'16''$	0.0

Table 4: *2-day look-ahead(f)* strategies and $\Delta t = 1h$: Cluster scenarios.

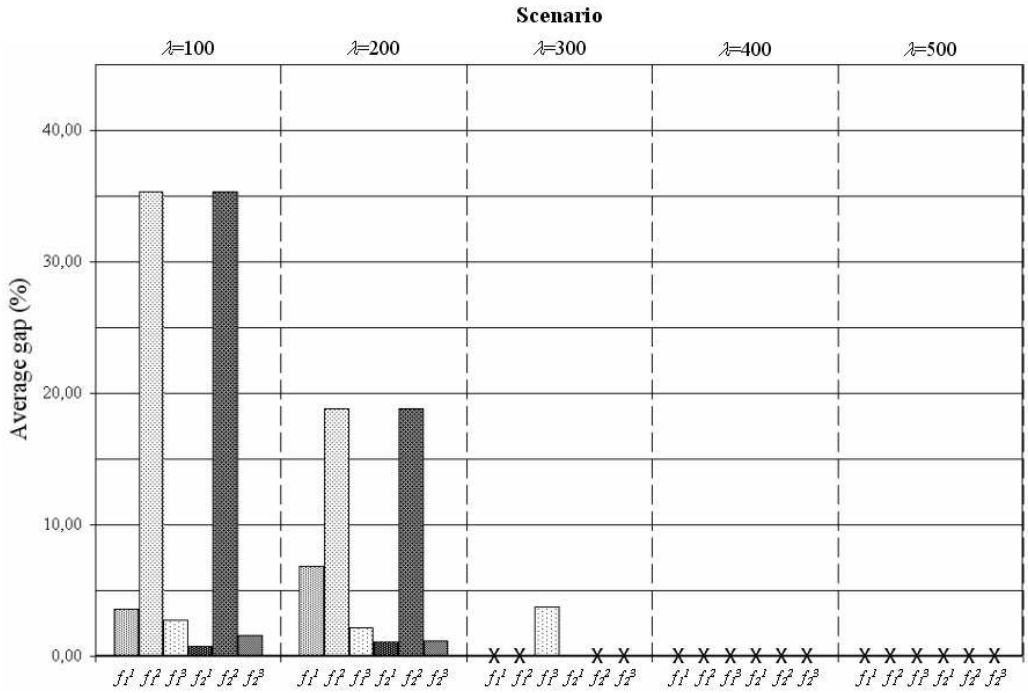

Figure 2: Average gap: Random scenarios.

the results for $\Delta t = 1h$ only. Results for random and cluster scenarios are presented in Tables 3 and 4, respectively. In both cases, the results obtained by the *2-day look-ahead(f_2^2)* strategies are only slightly affected by the values of α. In general, the quality of the results obtained by applying the other strategies improves when α increases.

If the short term objective is f_2^1 or f_2^3, the number of not served requests decreases when α increases or, if all requests are served, the length of the routes traveled decreases when α increases. In the case of cluster scenarios, all the requests are served independently from the chosen strategy. For $\alpha = 1^-$, the *2-day look-ahead(f_2^1)* strategy allows us to obtain in all but one scenarios results that are better than those obtained by means of the function f_2^3. The strategies based on *2-day look-ahead(f_2^2)* problem are confirmed to perform worse than the others.

The results obtained by a 2-day look-ahead strategy are definitely better than those obtained by 1-day look-ahead strategies. Comparing the best 1-day strategies, that is the *1-day look-ahead(f_1^3)* strategy in case of random scenarios and the *1-day look-ahead(f_1^1)* strategy in case of cluster scenarios, with the *2-day look-ahead(f_2^1)* strategy with $\alpha = 1^-$, we can see that in random scenarios, the most difficult ones, the number of not served requests is halved and the distance traveled is improved on average by about 2.33% as well. On the other hand, in case of cluster scenarios, where all the strategies allow us to serve all the requests, the best 2-day look-ahead strategy allows to decrease on average the traveled distance by about 3.09%.

In Figure 2 the average increase of the distance with respect to the minimum

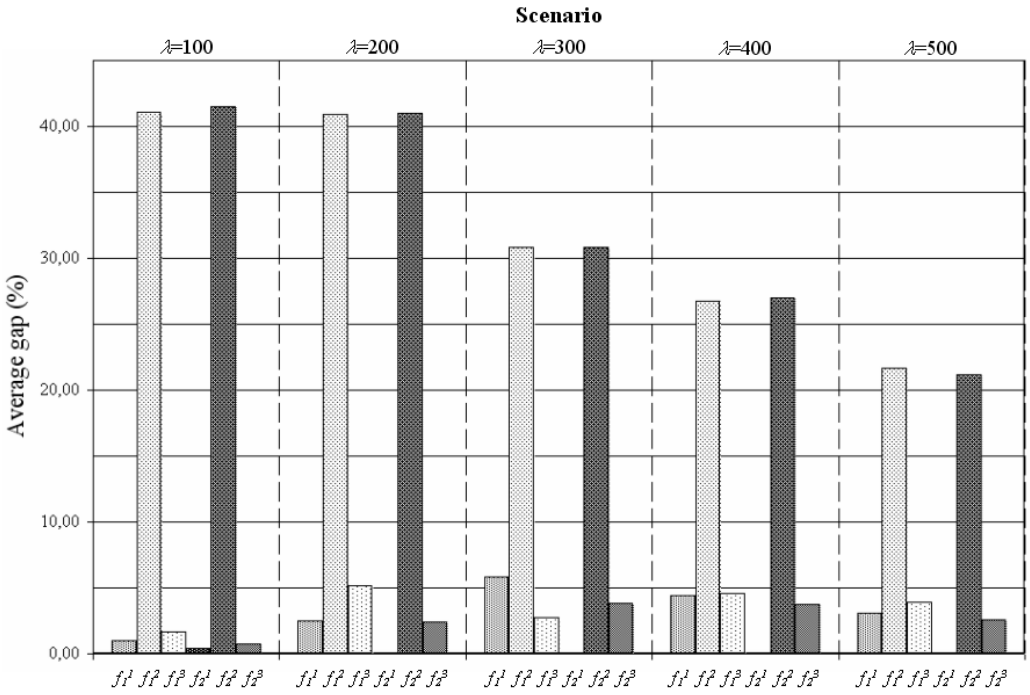

Figure 3: Average gap: Cluster scenarios.

Strategy	$\lambda = 100$		$\lambda = 200$		$\lambda = 300$		$\lambda = 400$		$\lambda = 500$	
	time	#	time	#	time	#	time	#	time	#
myopic	$256^h11'39''$	0.7	$312^h57'41''$	123.7	$321^h51'56''$	422.7	$325^h6'12''$	728.7	$325^h30'39''$	1074.7
2-day look-ahead	$181^h41'13''$	0.0	$235^h1'60''$	0.0	$267^h14'3''$	0.0	$287^h0'29''$	2.0	$298^h36'8''$	6.7

Table 5: Random scenarios, 3 vehicles.

value (over all the tested strategies) is shown. The values are reported only for the functions that have succeeded to serve all requests. The same type of representation is given in Figure 3 for the case of cluster scenarios. Here all values are reported because with any function all requests are served.

4.3 The comparison with a myopic strategy

In order to estimate the value of the dynamic strategies proposed in this paper, we compare the solutions obtained by the best of the tested strategy with the solutions obtained with a simple strategy that we call *myopic*. The *myopic* strategy models the behavior of a decision maker that takes decisions without the support of any tool. The comparison should provide evidence of the value of implementing sophisticated methods with respect to simple ones.

With the *myopic* strategy, a new request is inserted in the current routes and

Strategy	$\lambda = 100$		$\lambda = 200$		$\lambda = 300$		$\lambda = 400$		$\lambda = 500$	
	time	#	time	#	time	#	time	#	time	#
myopic	$205^h 8' 3''$	0.0	$284^h 14' 4''$	2.3	$307^h 36' 52''$	13.7	$315^h 31' 39''$	42.3	$317^h 28' 57''$	54.3
2-day look-ahead	$160^h 31' 58''$	0.0	$185^h 40' 20''$	0.0	$206^h 5' 6''$	0.0	$217^h 30' 51''$	0.0	$231^h 16' 8''$	0.0

Table 6: Cluster scenarios, 3 vehicles.

is never removed. The routes are modified only to accommodate new requests. As soon as an unpostponable request becomes known, an attempt is made to insert it in the current routes. The request is inserted by means of the cheapest insertion rule. If the request cannot be inserted without violating the time constraint on the routes length, the request is classified as not served. In fact, it will be served by the back-up service. In the case of a new postponable request, an attempt is made to insert it in the routes of the day after. If such attempt fails, an attempt is made to insert it in the current routes.

The *myopic* strategy has the advantage of being extremely simple. We compare it with the best 2-day strategy, that is the *2-day look-ahead*(f_2^1) strategy with $\alpha = 1^-$. The results are reported in Table 5 for the case of random scenarios and in Table 6 for the case of cluster scenarios. In the case of random scenarios, the *myopic* strategy is unable to serve all requests, even in the case of $\lambda = 100$. When $\lambda = 500$, the number of not served requests is very high and is about 20.51% of the total number of requests. With the *2-day look-ahead*(f_2^1) strategy with $\alpha = 1^-$ the percentage of not served requests is equal to 0.13%. The *myopic* strategy not only is extremely inefficient in terms of number of requests served, but also makes the vehicles travel much more to serve a much lower number of requests. In the case of cluster scenarios, the *myopic* strategy is able to serve all requests only when $\lambda = 100$, while the *2-day look-ahead* strategy serves all for any value of λ. Moreover, the *myopic* strategy makes the vehicles travel for an average time 42.5% longer than that required by the *2-day look-ahead* strategy.

Future developments

As future development, it is of practical interest to analyze the case where a request has to be served within d days, with $d > 2$. We believe that many of the ideas presented in this paper for $d \leq 2$ can be extended to the case of larger value of d. Moreover, according to the operational context where the problem is applied additional constraints can be taken into account. The most common include constraints on vehicle capacity and time windows. We believe that our solution approach can be extended to such cases as well.

Acknowledgements

We are grateful to three anonymous referees who helped us improve an earlier version of this paper.

References

[1] E. Angelelli, N. Bianchessi, R. Mansini, and M.G. Speranza. Management policies in a dynamic multi-period routing problem. In L.Bertazzi, J. van Nunen, and M.G. Speranza, editors, *Innovation in distribution logistics*, Lecture Notes in Economics and Mathematical System. Springer-Verlag (in press).

[2] E. Angelelli, M.W.P. Savelsbergh, and M.G. Speranza. Competitive analysis for dynamic multi-period uncapacitated routing problems. *Networks*, 49:308–317, 2007.

[3] E. Angelelli, M.W.P. Savelsbergh, and M.G. Speranza. Competitive analysis of a dispatch policy for a dynamic multi-period routing problem. *Operations Research Letters*, to appear, 2007.

[4] N. Bianchessi and G. Righini. Heuristic algorithms for the vehicle routing problem with simultaneous pick-up and delivery. *Computers & Operations Research*, 34:578–594, 2007.

[5] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30:105–119, 1997.

[6] Y. Dumas, J. Desrosiers, and F. Soumis. The pick-up and delivery problem with time windows. Technical Report G-89-17, Les Cahiers du GERAD, HEC-Montréal, Canada, 1989.

[7] M. Gendreau, F. Guertin, J-Y. Potvin, and E. Taillard. Parallel tabu search for real-time vehicle routing and dispatching. *Transportation Science*, 33:381–390, 1999.

[8] G. Ghiani, F. Guerriero, G. Laporte, and R. Musmanno. Real-time vehicle routing: Solution concepts, algorithms and parallel computing strategies. *European Journal of Operational Research*, 151:1–11, 2003.

[9] S. Ichoua, M.Gendreau, and J-Y Potvin. Diversion issues in real-time vehicle dispatching. *Transportation Science*, 34:426–438, 2000.

[10] O.B.G. Madsen, H.F. Ravn, and J.M. Rygaard. A heuristic algorithm for a dial-ride problem with time windows, multiple capacities and multiple objectives. *Annals of Operations Research*, 60:193–208, 1995.

[11] S. Mitrović-Minić, R. Krishnamurti, and G. Laporte. The double-horizon heuristic for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38:669–685, 2004.

[12] S. Mitrović-Minić and G. Laporte. Waiting strategies for the dynamic pickup and delivery problem with time windows. *Transportation Research Part B*, 38:635–655, 2004.

[13] N. Mladenovic and P. Hansen. Variable neighbourhood search. *Computers & Operations Research*, 24:1097–1100, 1997.

[14] H.N. Psaraftis. Dynamic vehicle routing: Status and prospects. *Annals of Operations Research*, 61:143–164, 1995.

[15] H.N. Psaraftis. Dynamic vehicle routing problems. In B. L. Golden and A. A. Assad, editors, *Vehicle Routing: Methods and Studies*, pages 223–248. Elsevier Science, Amsterdam, 1998.

[16] M.W.P. Savelsbergh and M. Sol. Drive: Dynamic routing of independent vehicles. *Operations Research*, 46:474–490, 1998.

[17] M.M. Solomon. Algorithms for vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35:254–266, 1987.

[18] J. Yang, P. Jaillet, and H. Mahmassani. Real-time multivehicle truckload pickup and delivery problems. *Transportation Science*, 38:135–148, 2004.