Parallel Tabu Search for Real-Time Vehicle Routing and Dispatching

MICHEL GENDREAU

Centre de recherche sur les transports and Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succ. Centre-ville, Montréal, Québec H3C 3J7, Canada

FRANÇOIS GUERTIN

Centre de recherche sur les transports, Université de Montréal, C.P. 6128, succ. Centre-ville, Montréal, Québec H3C 3J7, Canada

JEAN-YVES POTVIN

Centre de recherche sur les transports and Département d'informatique et de recherche opérationnelle, Université de Montréal, C.P. 6128, succ. Centre-ville, Montréal, Québec H3C 3J7, Canada

ÉRIC TAILLARD

Istituto Dalle Molle di Studi sull'Intelligenza Artificiale (IDSIA), Corso Elvezia 36, CH-6900 Lugano, Switzerland

An abundant literature about vehicle routing and scheduling problems is available in the scientific community. However, a large fraction of this work deals with static problems where all data are known before the routes are constructed. Recent technological advances now create environments where decisions are taken quickly, using new or updated information about the current routing situation. This paper describes such a dynamic problem, motivated from courier service applications, where customer requests with soft time windows must be dispatched in real time to a fleet of vehicles in movement. A tabu search heuristic, initially designed for the static version of the problem, has been adapted to the dynamic case and implemented on a parallel platform to increase the computational effort. Numerical results are reported using different request arrival rates, and comparisons are established with other heuristic methods.

Efficient distribution of goods and services is of paramount importance in today's competitive markets because transportation costs represent a non-negligible fraction of the purchase price of most products or services. This efficiency is achieved, in particular, through the determination of good routes and schedules for the fleet of vehicles that fulfills the distribution task. An abundant literature may be found on different types of vehicle routing and scheduling problems (see, for example, BALL et al. 1995). A large fraction of this work is concerned with static problems, that is, problems where all data are known before the routes are constructed and do not change thereafter, like the set of customers requiring transportation services, the quantity of goods to be picked up or delivered at each customer location, etc. Recent advances in communication and information technologies now allow the exploitation of new or updated information that is revealed as the routes are executed: a vehicle has broken down or is unexpectedly delayed, a new customer has just called in for quick service, etc. These developments have led to the growth of a new class of problems, known as dynamic routing and scheduling problems

Transportation Science, © 1999 INFORMS
Vol. 33, No. 4, November 1999

0041-1655/99/3304-0381 $05.00
pp. 381–390, 1526-5447 electronic ISSN

(DROR and POWELL, 1993; POWELL, JAILLET and ODONI, 1995; PSARAFTIS, 1995; LUND, MADSEN, and RYGAARD, 1996). Such problems are found in many different application domains, like delivery of petroleum products or industrial gases (BELL et al., 1983; BROWN et al., 1987; BAUSCH, BROWN, and RONEN, 1995), transportation on demand for the elderly or the handicapped (WILSON and COLVIN, 1977), and emergency services (GENDREAU, LAPORTE, and SEMET, 1997).

This paper is about a dynamic problem motivated from courier service applications found in the local operations of international shipping services (e.g., Federal Express), where the courier is collected at different customer locations and brought back to a central office for further processing and shipping. Here, the uncertainty comes from the occurrence of new service requests that must be assigned in real time to an appropriate vehicle. As opposed to the static scenario, where all requests are known before the routes are constructed, each decision must be taken on the basis of the available requests, that is, without knowledge of future incoming requests.

The paper is organized along the following lines. First, the static version of the problem is presented in Section 1. A problem-solving method based on tabu search is then briefly introduced in Section 2 (TAILLARD et al., 1997), followed by a description of a parallel implementation (BADEAU et al., 1997). Section 3 introduces the dynamic problem and explains how the original algorithmic design is modified to handle a dynamic environment. Computational results are reported in Section 4, where different ways of dispatching new incoming requests are compared.

1. STATIC PROBLEM

THE STATIC VERSION of our problem is representative of a class of problems known as the vehicle routing problem with time windows (DESROSIERS et al., 1995). From a graph theoretical perspective, the problem can be stated as follows. Let $G = (V, E)$ be a complete undirected graph with vertex set $V = \{v_0, v_1, v_2, \ldots, v_n\}$ and edge set $E = \{(v_i, v_j): v_i, v_j \in V, i < j\}$. In this graph, vertex v_0 is the depot and the remaining vertices are customers to be serviced. Each vertex has a time window $[e_i, l_i]$, where e_i and l_i are the earliest and latest service time, respectively (with e_0, the earliest start time and l_0, the latest end time of each route). Finally, a symmetric distance matrix $D = (d_{ij})$ that satisfies the triangle inequality is defined on E, with travel times t_{ij} proportional to the distances.

Given a fixed size fleet of m identical vehicles, the goal is to find a set of minimum cost vehicle routes, originating from and terminating at the depot v_0, such that:

- each vehicle services one route;
- each vertex v_i, $i = 1, \ldots, n$ is visited exactly once;
- the start time of each vehicle route is greater than or equal to e_0;
- the end time of each vehicle route is less than or equal to l_0;
- the time of beginning of service b_i at each vertex v_i, $i = 1, \ldots, n$ is greater than or equal to the earliest service time e_i; if the vehicle's arrival time t_i is less than e_i, a waiting time $w_i = (e_i - t_i)$ is incurred.

The objective function f to be minimized over the set of feasible solutions S is

$$f(s) = \sum_{k=1}^{m} d_k + \sum_{i=1}^{n} \alpha_i (t_i - l_i)^+, \quad s \in S, \qquad (1)$$

where $y^+ = \max\{0, y\}$, d_k is the total distance traveled on route k, $k = 1, \ldots, m$, and α_i is a lateness penalty coefficient associated with vertex v_i, $i = 1, \ldots, n$.

This definition implies that each route must start and end within the time window associated with the depot. Furthermore, a soft time window constraint is found at each customer location. The time window is soft because the vehicle can arrive before the lower bound or after the upper bound. If the vehicle arrives too early, it must wait to start its service. If the vehicle is too late, a penalty for lateness is incurred. That is, the upper bound of the time window is relaxed into the objective function in a Lagrangean fashion. According to objective function, eq. 1, the penalty coefficients can be adjusted to each customer. For example, large coefficients can be associated with customers with rather strict time requirements and small coefficients with customers with some flexibility.

The next section will now present an algorithm based on tabu search for solving this problem.

2. ADAPTIVE MEMORY TABU SEARCH

THE TABU SEARCH heuristic for solving the static problem presented in Section 1 follows the guidelines in GLOVER (1989, 1990) and is characterized by the exploitation of an adaptive memory (ROCHAT and TAILLARD, 1995). This algorithm can be summarized as follows:

1. Construct I different initial solutions with a stochastic insertion heuristic (i.e., the rule for choos-

ing the next customer to be inserted contains stochastic elements).
2. Apply tabu search to each solution and store the resulting routes in the adaptive memory.
3. For W iterations do:
   3.1 Construct an initial solution from the routes found in the adaptive memory and set this solution as the current solution.
   3.2 For C cycles do:
      a. decompose the current solution into D disjoint subsets of routes
      b. apply tabu search to each subset of routes
      c. merge the resulting routes to form the new current solution
   3.3 Add the resulting routes to the adaptive memory (if indicated).
4. Apply a postoptimization procedure to each individual route of the best solution found.

Details about this algorithm may be found in TAILLARD et al. (1997). In the following, the main components are briefly introduced.

2.1 Adaptive Memory

As the tabu search heuristic proceeds, the routes of the best solutions visited during the search are stored in an adaptive memory. New initial solutions for the tabu search are then created by combining routes taken from different solutions in this memory. This way of combining components of different high quality solutions to create a new solution is quite similar to the crossover operator found in genetic algorithms (HOLLAND, 1975). Note that a new solution produced by the tabu search is included in the memory if the memory is not filled yet, or the new solution is better than the worst solution stored in memory, in which case the latter is removed.

2.2 Decomposition/Reconstruction

Each starting solution is partitioned into D disjoint subsets of routes, with each subset or subproblem being processed by a different tabu search (TAILLARD, 1993). When every subproblem is solved, the new routes are simply merged back to form the new current solution. The decomposition uses the polar angle associated with the center of gravity of each route. Through a sweep procedure, the domain is partitioned into sectors that approximately contain the same number of routes. C cycles of decomposition/reconstruction take place before the final solution is sent to the adaptive memory for possible inclusion. At each cycle, the decomposition (i.e., the subset of routes in each subproblem) changes by choosing a different starting angle to construct the sectors.

2.3 Tabu Search

The tabu search is applied to each subset of routes produced through the decomposition procedure. This tabu search exploits a neighborhood specifically designed for problems with time windows. The method for generating this neighborhood is called the CROSS exchange. It is basically a chain exchange procedure, where two segments of variable length are taken from two different routes and moved from one route to another. A neighborhood of manageable size is obtained by restricting the CROSS exchanges to segments with L service points at most, where L is a parameter of the algorithm.

2.4 Parallel Implementation

Fast response times are required to cope with real-time environments. To increase the computational effort in a given time interval, the tabu search heuristic was implemented on a network of workstations (BADEAU et al., 1997). A master–slave scheme is a natural way to parallelize the algorithm on this coarse-grained, loosely connected architecture. The master process manages the adaptive memory and creates new starting solutions for the slave processes which, themselves, apply tabu search to improve the solutions. This scheme induces a multithread search strategy where different search paths are followed in parallel (CRAINIC, TOULOUSE, and GENDREAU, 1997). Namely, many different tabu searches run in parallel, each starting from a different initial solution. Although the threads run independently, they communicate implicitly through the adaptive memory as they feed this memory with their resulting solutions and get new starting solutions from it.

The decomposition procedure allows for another level of parallelization, because the subproblems created by the decomposition can be allocated to different processors. The implementation thus involves the following processes:

- The *Manager* process that manages the adaptive memory and creates new starting solutions.
- *R Decomposition* processes that decompose a problem into subproblems for the tabu processes (see below); each decomposition process corresponds to a distinct search thread.
- The *Dispatcher* process that dispatches the work among processors.
- *P Tabu* processes that apply tabu search to subproblems.

The Manager, Decomposition, and Dispatcher processes are typically located on the same processor because they require modest computation times,

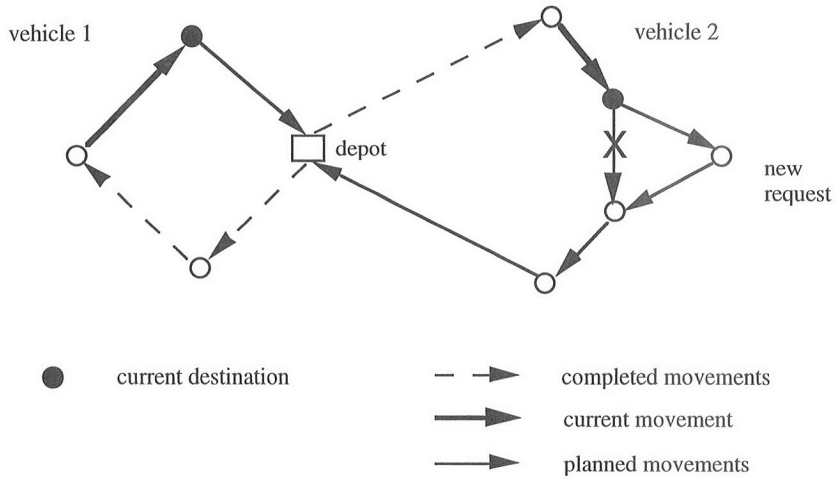

Fig. 1. Dispatching situation involving two vehicles (at the time of occurrence of a new request).

whereas a different processor is assigned to each Tabu process (for a total of $P + 1$ processors). Previous experiments (BADEAU et al., 1997) have shown that the parallel implementation is competitive with the sequential one, with respect to solution quality, for the same amount of computational work. However, this work is done much faster, in terms of wall clock times, on the parallel platform.

3. DYNAMIC PROBLEM

IN THE DYNAMIC version of the problem, a number of service requests are not known completely ahead of time, but are rather dynamically revealed as time goes by. This situation impacts the problem-solving method developed for the static problem in a number of ways, as explained in the following.

3.1 Operating Scenarios

The dynamic operating scenarios are based on the following assumptions:

- Requests must be received before a fixed time deadline to be serviced the same day. Those that are received after the deadline, however, may be kept for the next day. The operations day thus typically starts with a number of pending or static requests (for which a solution may have been constructed beforehand).
- Uncertainty comes from a single source, namely the occurrence of new service requests. In particular, there is no uncertainty associated with the service locations, like cancellation. Furthermore, the travel times are assumed to be known with certainty because no unexpected perturbation comes from the external world (like sudden congestion on the network caused by an accident, vehicle breakdown, etc.).
- Communication between the central dispatch office and the drivers take place at each service location and is aimed at identifying their next destination. Consequently, the drivers do not know the global picture represented by their current planned route (which may be modified frequently by the problem-solving procedure).
- If some waiting time is expected at the drivers' next destination, they are asked to wait at their current location. This is a form of least commitment strategy, because a movement is performed at the latest possible time to allow for last minute changes to the planned route due to the arrival of new service requests. Once a driver is en route to his next destination, however, he must necessarily service this location (i.e., no diversion is allowed).

Figure 1 illustrates a typical situation involving two vehicles at an instant associated with the occurrence of a new service request. In this figure, the little square is the depot and the circles are customer requests. Dotted arcs correspond to completed movements: customers found at the end of those arcs are already serviced and are not considered anymore. The black circles correspond to the current destination of each vehicle. These customers, as well as their successors on each route, are still unserviced. The current solution thus consists of a set of planned routes, where each route starts with the driver's current destination and ends with the last customer on the planned route (including the return trip to the depot). Note that each problem is now open, in the sense that the starting and ending points of the planned routes are not the same. As for the static problem, the objective is to minimize a

weighted sum of total distance traveled and lateness at customer locations.

Note that a natural way to quickly handle a new request, as illustrated in Fig. 1, is through its insertion between two consecutive customers on a planned route. The insertion place for the new request is chosen to minimize the additional cost over all feasible insertion places in the current solution.

3.2 Adaptation of the Static Problem-Solving Method

Broadly speaking, the dynamic environment is handled by solving a series of static problems, with a new problem being defined each time an input update occurs. The static problem contains the (yet unserviced) customer requests known at the time of the input update. Although each static problem produced by the occurrence of a new event can be solved by the algorithm presented in the previous section, the dynamic evolution of the dispatching situation over time implies a number of non-trival modifications to the algorithmic design.

The problem-solving method now interacts with its dynamic environment in the following way:

1. While "no event," optimize the planned routes using tabu search;
2. if an event occurs, then
   - 2.1 stop each tabu search thread and add the routes of their best solution to the adaptive memory (if indicated);
   - 2.2 if the event is the occurrence of a new request, then
     - a. update the adaptive memory through the insertion of the new request in each solution;
     - b. if no feasible insertion place is found, reject the request;

     otherwise (end of service at a customer location)
     - a. identify the driver's next destination, using the best solution stored in the adaptive memory;
     - b. update the other solutions accordingly;
   - 2.3 restart the tabu search processes with new solutions obtained from the adaptive memory.

Thus, the tabu search works in background between the events, trying to find a better set of planned routes. The search threads are interrupted whenever an input update occurs (because they do not work anymore on configurations that represent the current situation). Here, an input update occurs when a new request is received, or when a vehicle has completed its service at a customer location. These events are precisely handled as follows:

(a) In case of an end of service at a customer location (which is not a random event, because the travel times between the service locations are known with certainty), the driver must know his next destination, that is, the next customer to be serviced. Because many different solutions are maintained in the adaptive memory (e.g., 30 solutions in the current implementation), the best one is used for this purpose. The remaining solutions in memory are updated by removing this customer from its current location and by inserting it in the first position in the driver's planned route if it is not already there.

(b) In case of occurrence of a new request, the latter is inserted in all solutions found in the adaptive memory (see Fig. 1). The insertion place is chosen to minimize the additional cost to the current solution. If there is no feasible insertion place in any solution (due to the hard time window at the depot), the request is rejected. Thus, the customer is told almost immediately that his request cannot be handled. Alternatively, if a feasible insertion is found, the request is accepted and all solutions with no feasible insertion places are discarded from memory. Then, the best solution in memory (with the new request) is processed by a local search heuristic for further improvement. This heuristic stops at the first local minimum, using a neighborhood based on CROSS exchanges. In this way, a solution of high quality is quickly produced with the new set of requests (and is available in the adaptive memory for creating new starting solutions).

After these modifications, the search threads can be restarted with new solutions constructed from the updated adaptive memory.

Note finally that the work performed by the search threads on their current solution is not lost when they are interrupted by a new event. Rather, their best solution is returned to the adaptive memory for possible inclusion (cf., Step 2.1).

3.3 A Finer Refinement

In the decomposition procedure, a fixed number C of cycles could lead to a loss of diversity in the adaptive memory, because more solutions would be returned to and constructed from the adaptive memory (for a given set of requests) when the time interval between two consecutive events increases. The C value is thus dynamically adjusted after each event

with the formula,

$$C = C_0 + \left\lfloor \frac{H}{R} \right\rfloor,$$

where H is the number of calls to the adaptive memory between the last two events, and R is the number of decomposition processes or search threads. Hence, the number of cycles C is based on the average number of calls to the adaptive memory per search thread, with a minimum of $C_0 = 6$ cycles. When an increase in the number of calls to the adaptive memory is observed, the C value also increases: the Tabu processes thus work longer on each problem before returning their resulting routes to the adaptive memory. Conversely, when a decrease is observed, the C value also decreases.

4. COMPUTATIONAL EXPERIMENTS

TO TEST OUR algorithm, a discrete-time simulator was developed. The simulator uses data taken from Solomon's 100-customer Euclidean problems (SOLOMON, 1987) to produce new service requests. In these problems, the customer locations are distributed within a $[0, 100]^2$ square and the travel times are equivalent to the corresponding Euclidean distances. Six different sets of problems are defined, namely C1, C2, R1, R2, RC1, and RC2. The customers are uniformly distributed in the problems of type R, clustered in groups in the problems of type C and mixed in the problems of type RC. Problems of type 1 have a narrow time window at the central depot so that only a few customers can be serviced on each route; conversely, problems of type 2 have a wider time window at the depot. Finally, a fixed service time for loading or unloading the goods is found at each customer location. This service time is set at 10 time units per customer for the problems of type R and RC, and 90 time units per customer for the problems of type C.

The time horizon for the simulation is adjusted to create two different types of scenarios: scenarios of type 1, with approximately three requests per minute on average, and scenarios of type 2, with approximately one request per minute on average. Accordingly, data relating to time in Solomon's file, like the lower and upper bounds of the time windows and the travel times, are multiplied by the scaling factor

$$\frac{T}{l_0 - e_0},$$

where T is the time horizon for the simulation. Using minutes as time units for time-related data in Solomon's files, T is set to 15 minutes in scenarios of type 1 and to 60 minutes in scenarios of type 2.

TABLE I

Parameter Settings

initialization
number of initial solutions: $I = \max\{4, P\}$, where P is the number of tabu search processes
adaptive memory
size: $M = 30$ solutions
decomposition/reconstruction
minimum number of cycles: $C = 6$
tabu search
number of iterations: $A \times [1 + (DR - 1)/B]$, where DR is the current decomposition/reconstruction cycle, $DR = 1, \ldots, C$. For scenarios of type 1, $A = 30$ and $B = 3$; for scenarios of type 2, $A = 60$ and $B = 6$.
neighborhood:
maximum length of route segments: $L = 6$

The set of requests is divided into two subsets (each one with half of the requests). The first subset contains requests that are known at the start of the day, like those received at the end of the previous day for next-day service. They are randomly selected among the entire set of requests with a bias in favor of requests with early time windows. Initial routes servicing those requests are constructed in a static fashion, using the tabu search heuristic presented in Section 2. The second subset contains requests that are received in real time. In this case, a time of occurrence is associated with each request. For request i, this value is randomly generated in the interval $[0, \bar{e}_i]$, where

$$\bar{e}_i = \frac{T}{l_0 - e_0} \min\{e_i, t_{i-1}\}.$$

Note that t_{i-1} is the departure time from i's predecessor in the best solution for the static problem that we know of (see TAILLARD et al., (1997) for a list of those best solutions). Thus, the best solution can still be produced in the dynamic setting. However, this is unlikely because the solution procedure does not have any knowledge of future incoming requests when it commits to a particular assignment.

4.1 Numerical Results

The experiments reported in this section were performed on a network of 17 SUN UltraSparc workstations (143 MHz). Each process was programmed in C++ and communication between the processes was handled by the Parallel Virtual Machine software. In these experiments, Eq. 1 in Section 1 is to be minimized, with $\alpha_i = \alpha = 1$ for every customer. The algorithm's parameter settings are shown in Table I.

TABLE II

Comparison of Different Heuristics on Scenarios of Type 1

Problem Set (No. of Problems)	Insertion	Insertion +	Rebuild	Rebuild +	Adaptive Descent	Adaptive Tabu
R1 (12)	1389.0*	1212.9	1292.0	1264.1	1234.0	1212.4
	262.1	90.5	120.9	89.2	56.7	43.1
	4.42	1.17	1.08	0.75	0.58	0.17
C1 (9)	1559.6	883.9	893.0	871.4	837.2	836.3
	711.3	9.9	31.7	1.6	0.0	0.0
	0.00	0.00	0.00	0.00	0.00	0.00
RC1 (8)	1561.6	1376.7	1449.4	1418.4	1367.6	1333.7
	354.9	88.5	105.3	87.8	68.9	68.5
	5.13	1.38	0.75	1.38	0.38	0.38
R2 (11)	1203.1	1069.8	1208.4	1145.9	1020.9	1024.7
	546.4	66.8	447.7	391.7	96.7	50.7
	1.18	0.54	1.09	0.54	0.36	0.00
C2 (8)	935.0	650.3	689.6	644.9	596.4	597.3
	484.2	0.0	97.0	2.9	0.0	0.0
	1.00	0.13	0.00	0.00	0.00	0.00
RC2 (8)	1461.0	1253.8	1358.1	1356.2	1189.8	1174.9
	612.9	58.9	166.1	108.7	29.1	12.1
	0.00	0.00	0.00	0.00	0.00	0.00
Overall	1349.9	1080.8	1157.3	1124.5	1050.1	1039.2
	485.2	55.2	171.6	124.8	45.1	30.7
	2.05	0.57	0.57	0.46	0.25	0.09

*Average distance, lateness, and number of unserviced customers.

TABLE III

Comparison of Different Heuristics on Scenarios of Type 2

Problem Set (No. of Problems)	Insertion	Insertion +	Rebuild	Rebuild +	Adaptive Descent	Adaptive Tabu
R1 (12)	1389.0*	1212.9	1292.0	1278.6	1198.3	1204.4
	262.1	90.5	120.9	85.4	46.4	40.9
	4.42	1.17	1.08	0.75	0.33	0.17
C1 (9)	1559.6	883.9	891.9	862.7	833.6	830.9
	711.3	9.9	23.2	1.0	0.0	0.0
	0.00	0.00	0.00	0.00	0.00	0.00
RC1 (8)	1561.6	1371.3	1441.6	1401.9	1328.9	1337.8
	354.9	88.5	118.2	75.8	50.6	52.7
	5.13	1.38	1.38	1.38	0.38	0.13
R2 (11)	1203.1	1069.8	1171.8	1134.3	1034.8	1011.7
	546.4	66.8	249.1	275.6	12.6	10.7
	1.18	0.54	0.54	0.54	0.45	0.00
C2 (8)	935.0	650.3	673.3	641.7	597.3	595.9
	484.2	0.0	10.8	4.1	0.0	0.0
	1.00	0.13	0.00	0.00	0.00	0.00
RC2 (8)	1461.0	1253.8	1388.3	1329.8	1185.6	1154.9
	612.9	58.9	101.0	79.5	29.1	20.9
	0.00	0.00	0.00	0.00	0.00	0.00
Overall	1349.9	1080.0	1150.8	1117.4	1038.6	1031.6
	485.2	55.2	111.4	95.4	23.8	21.4
	2.05	0.57	0.54	0.46	0.21	0.05

*Average distance, lateness, and number of unserviced customers.

Tables II and III compare our algorithm with other heuristic approaches for handling new customer requests. In each case, the fleet size was set to the minimum number of vehicles reported in the literature for each problem. The first four heuristic methods maintain a single solution. The *Insertion* method simply inserts a new customer at the location that minimizes the additional cost over the current set of planned routes. The *Insertion+* method applies a local search heuristic based on CROSS exchanges to improve the planned routes after each insertion. The *Rebuild* method reconstructs the planned routes each time a new service request occurs, using an adaptation of Solomon's I1 insertion heuristic (SOLOMON, 1987). *Rebuild+* applies a local search heuristic based on CROSS exchanges to the solution obtained with *Rebuild.*

The *Adaptive descent* method is a slight modification to our parallel tabu search heuristic. In this case, the Tabu processes are stopped as soon as a local optimum is found. This method thus corresponds to a multi-start local search heuristic, based on starting solutions constructed from the adaptive memory. The last method, *Adaptive tabu,* is our parallel tabu search heuristic. The results shown in Tables II and III for *Adaptive descent* and *Adaptive tabu* were obtained with $R = 8$, $D = 2$, and $P = 16$ on the problems of type 1. That is, eight search threads were running in parallel, with each solution or set of routes being divided into two subsets of routes through the decomposition procedure, for a total of 16 tabu search processes. For the problems of type 2, $R = 16$, $D = 1$, and $P = 16$. Thus, no decomposition took place on these problems because each solution only contains a few routes (typically, two to four routes).

The three numbers in each entry of Tables II and III are the distance traveled, total lateness at customer locations (scaled back to the original value) and number of unserviced customers, respectively, for each problem set. The numbers are averages taken over all problems in a given set, except for the row, Overall, which contains averages taken over the entire set of 56 test problems.

Tables II and III show that running adaptive-memory-based heuristics to optimize the planned routes between two events is beneficial. In particular, a larger number of requests can now be serviced, thus increasing the overall throughput of the system. The methods *Adaptive descent* and *Adaptive tabu* generally perform better on scenarios of type 2, where the request arrival rate is lower (as compared with scenarios of type 1). Given that these methods run during the entire time interval between two events, they can produce better solutions when more computation time is available. There is little or no difference between scenarios of type 1 and 2 for the other heuristic approaches. These heuristics are quite fast, but cannot exploit any additional time once they have produced a solution (i.e., they stop and wait for the next event).

TABLE IV
Static Versus Dynamic Solutions

	Dynamic		
	Scenario 1	Scenario 2	Static
R1	1212.4*	1204.4	1209.4
	43.1	40.9	0.0
	0.17	0.17	0.00
C1	836.3	830.9	828.4
	0.0	0.0	0.0
	0.00	0.00	0.00
RC1	1333.7	1337.8	1389.2
	68.5	52.7	0.0
	0.38	0.13	0.00
R2	1024.7	1011.7	980.3
	50.7	10.7	0.0
	0.00	0.00	0.00
C2	597.3	595.9	589.9
	0.0	0.0	0.0
	0.00	0.00	0.00
RC2	1174.9	1154.9	1117.4
	12.1	20.9	0.0
	0.00	0.00	0.00
Overall	1039.2	1031.6	1027.2
	30.7	21.4	0.0
	0.09	0.05	0.00

*Average distance, lateness, and number of unserviced customers.

Table IV compares the solutions reported in TAILLARD et al. (1997) in the static case, using the tabu search heuristic presented in Section 2, with those produced in the dynamic settings. As in Tables II and III, the three numbers in each entry are the average distance traveled, total lateness at customer locations, and number of unserviced customers, respectively. Note that the static solutions service every customer before their time window's upper bound, as opposed to the dynamic ones, where some customers are unserviced or serviced late. As expected, the sum of distance and lateness is higher in the dynamic case. The average gap between the dynamic and static solutions is equal to 4.1% for scenarios of type 1 (higher request arrival rate) and 2.5% for scenarios of type 2.

4.2 Benefits of Parallelization

Through parallelization, the computational effort can be increased between two events. Table V shows results obtained with *Adaptive tabu* using $P = 1, 2,$

TABLE V

Adaptive Tabu *with an Increasing Number of Processors Using Scenarios of Type 1*

Problem Set	No. of Processors				
	1	2	4	8	16
R1	1210.8*	1215.2	1211.4	1197.9	1212.4
	51.7	35.8	45.5	42.3	43.1
	0.58	0.58	0.50	0.25	0.17
C1	857.2	849.2	843.2	831.0	836.3
	1.3	0.0	0.0	0.0	0.0
	0.00	0.00	0.00	0.00	0.00
RC1	1360.3	1345.9	1348.4	1337.4	1333.7
	65.9	61.5	56.1	60.6	68.5
	0.50	0.38	0.38	0.38	0.38
R2	1050.4	1063.9	1041.3	1014.9	1024.7
	40.4	30.6	25.5	51.2	50.7
	0.45	0.45	0.36	0.00	0.00
C2	628.7	612.5	616.3	609.1	597.3
	0.0	0.0	0.0	0.0	0.0
	0.00	0.00	0.00	0.00	0.00
RC2	1192.6	1206.7	1191.6	1187.8	1174.9
	43.4	23.8	36.3	19.3	12.1
	0.00	0.00	0.00	0.00	0.00
Overall	1058.0	1058.0	1050.5	1037.4	1039.2
	34.8	25.9	28.0	30.5	30.7
	0.29	0.27	0.23	0.11	0.09

*Average distance, lateness and number of unserviced customers.

4, 8, and 16 processors to run the tabu search (using scenarios of type 1 with a higher request arrival rate). The three numbers in each entry are the same as in Table IV. As expected, these numbers show that an increase in the number of processors generally lead to better solutions, by allowing more customers to be serviced and by reducing the sum of distance and lateness. These numbers also show that a request arrival rate of three requests per minute can be handled with only $P = 8$ processors, because there is not much difference between $P = 8$ and $P = 16$.

5. CONCLUSION

A TABU SEARCH heuristic, initially designed for static vehicle routing problems with time windows, has been adapted to the dynamic case. Computational results show that the tabu search heuristic allows more customers to be serviced and reduces the total distance traveled and total lateness at customer locations, when compared with other heuristic approaches. The current system can be improved or extended in a number of ways:

1. Diverting a vehicle away from its current destination could be considered when a new service request is received in the vicinity of its current position. Allowing diversion is an interesting avenue to explore, although some issues must be carefully addressed: in what situations should diversion be allowed? should we consider the possibility of diverting away many vehicles at once? how much computation time should be allowed to the exploration of diversion opportunities, given that diversion takes place in a context where the time pressure is important (vehicles are moving fast and diversion opportunities may be quickly lost)?
2. Retaining new service requests (if time permits) could also be considered. By collecting a batch of requests and by dispatching them all at once, the adverse effects of dispatching requests in a sequential fashion could be alleviated.
3. The current problem-solving approach optimizes the planned routes based on information (requests) known with certainty. It would be interesting to integrate probabilistic knowledge about the future, like the time–space distribution of future requests, to improve decision making at the current time.
4. A single source of uncertainty, namely the time–space occurrence of new service requests, is considered in this work. Other sources of uncertainty, like service delays due to congestion or

vehicle breakdown, could be considered as well. These additional events raise some interesting issues, for example, how and when recourse actions should be taken to respond to them.

5. Extensions to problems where a service request includes both a pick-up and a delivery location represent an interesting research avenue. Applications include, for example, local express mail delivery services, where both the pick-up and delivery locations are found in the same area and are serviced by the same vehicle. These problems are more difficult to handle, due to the precedence constraint between the pick-up and the delivery.

ACKNOWLEDGMENTS

THIS WORK WAS partially supported by the Canadian Natural Sciences and Engineering Research Council (NSERC) and by the Québec Fonds pour la Formation de Chercheurs et l'Aide à la Recherche. Eric Taillard was postdoctoral fellow at the Centre de recherche sur les transports of Montreal University when this work was performed. He benefitted from an NSERC International Postdoctoral fellowship and was also partially funded by Strategic Grant STR0149269. This support is gratefully acknowledged.

REFERENCES

P. BADEAU, M. GENDREAU, F. GUERTIN, et al., "A Parallel Tabu Search Heuristic for the Vehicle Routing Problem with Time Windows," *Transp. Res.* **5C,** 109–122 (1997).

M. O. BALL, T. L. MAGNANTI, C. L. MONMA, and G. L. NEMHAUSER (eds), *Network Routing, Handbooks in Operations Research and Management Science*, Vol. 8, North-Holland, Amsterdam, 1995.

D. O. BAUSCH, G. G. BROWN, and D. RONEN, "Consolidating and Dispatching Truck Shipments of Mobil Heavy Petroleum Products," *Interfaces.* **25,** 1–17 (1995).

W. BELL, L. M. DALBERTO, M. L. FISHER, et al. "Improving the Distribution of Industrial Gases with an On-Line Computerized Routing and Scheduling Optimizer," *Interfaces.* **13,** 4–23 (1983).

G. G. BROWN, C. J. ELLIS, G. W. GRAVES, and D. RONEN, "Real-Time Wide Area Dispatching of Mobil Tank Trucks," *Interfaces.* **17,** 107–120 (1987).

T. G. CRAINIC, M. TOULOUSE, and M. GENDREAU, "Toward a Taxonomy of Parallel Tabu Search Heuristics," *INFORMS J. Comput.* **9,** 61–72 (1997).

J. DESROSIERS, Y. DUMAS, M. M. SOLOMON, and F. SOUMIS, "Time Constrained Routing and Scheduling," in *Network Routing, Handbooks in Operations Research and Management Science*, Vol. 8, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser (eds), North-Holland, Amsterdam, 35–139, 1995.

M. DROR and W. B. POWELL (eds), Special issue on Stochastic and Dynamic Models in Transportation, *Opns. Res.* **41,** 1–235 (1993).

M. GENDREAU, G. LAPORTE, and F. SEMET, Solving an Ambulance Location Model by Tabu Search, Technical report CRT-97-18, Centre de recherche sur les transports, Université de Montréal, 1997.

F. GLOVER, "Tabu Search—Part I," *ORSA J. Comput.* **1,** 190–206 (1989).

F. GLOVER, "Tabu Search—Part II," *ORSA J. Comput.* **2,** 4–32 (1990).

J. H. HOLLAND, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, MI, 1975.

K. LUND, O. B. G. MADSEN, and J. M. RYGAARD, Vehicle Routing Problems with Varying Degrees of Dynamism, Technical report IMM-REP-1996-1, Institute of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark, 1996.

W. B. POWELL, P. JAILLET, and A. ODONI, "Stochastic and Dynamic Networks and Routing," in *Network Routing, Handbooks in Operations Research and Management Science*, Vol. 8, M. O. Ball, T. L. Magnanti, C. L. Monma, and G. L. Nemhauser (eds), North-Holland, Amsterdam, 141–295, 1995.

H. N. PSARAFTIS, "Dynamic Vehicle Routing: Status and Prospects," *Ann. Opns. Res.* **61,** 143–164 (1995).

Y. ROCHAT and E. D. TAILLARD, "Probabilistic Diversification and Intensification in Local Search for Vehicle Routing," *J. Heuristics,* **1,** 147–167 (1995).

M. M. SOLOMON, "Algorithms for the Vehicle Routing and Scheduling Problem with Time Window Constraints," *Opns. Res.* **35,** 254–265 (1987).

E. D. TAILLARD, "Parallel Iterative Search Methods for Vehicle Routing Problems," *Networks,* **23,** 661–673 (1993).

E. D. TAILLARD, P. BADEAU, and M. GENDREAU, et al. "A Tabu Search Heuristic for the Vehicle Routing Problem with Soft Time Windows," *Transp. Sci.* **31,** 170–186 (1997).

N. H. M. WILSON and N. H. COLVIN, Computer Control of the Rochester Dial-A-Ride System, Technical report R-77-30, Department of Civil Engineering, MIT, Cambridge, MA, 1977.

(Received: November 1996; revisions received: September 1997, June 1998; accepted: June 1998)