Dynamic vehicle routing by means of a genetic algorithm

Giselher Pankratz
Department of Business Administration and Economics, Fern Universität – University of Hagen, Hagen, Germany

Abstract

Purpose – To propose and to evaluate a new genetic algorithm (GA) for solving the dynamic pickup and delivery problem with time windows (DPDPTW).

Design/methodology/approach – First, a grouping genetic algorithm (GGA) for the (static) PDPTW is described. In order to solve the dynamic problem, the GGA then is embedded in a rolling horizon framework. Special updating mechanisms are provided which assure that reusable solution knowledge is preserved over the plan revisions. The approach is evaluated using a large number of test instances with varying degrees of dynamism.

Findings – The experimental results have demonstrated that the proposed approach is able to find high-quality solutions when compared with two comparative heuristics.

Research limitations/implications – Future research will be dedicated to the following issues: testing the proposed method using larger problem instances, using more sophisticated objective functions in order to further improve and evaluate the approach, integrating fast local search techniques into the genetic search, speeding up the algorithm by optimizing its implementation.

Practical implications – In order to meet the increasing demands on the flexibility and the promptness of transportation services, algorithms are needed for dispatching transportation requests that arrive dynamically during the planning period. The findings of this contribution justify the employment of GAs in such dynamic transportation planning environments.

Originality/value – Although the application of GAs in dynamic environments attracts growing attention, up to now no such algorithm has been published for the DPDPTW. To the best of the author's knowledge, this is the first time a GA has been applied to the DPDPTW.

Keywords Production scheduling, Transport management, Programming and algorithm theory

Paper type Research paper

1. Introduction

While in the past research mainly concentrated on static variants of the vehicle routing problem (VRP) in which all requests are known in advance and no uncertainty exists, increasing interest in dynamic VRPs can be observed in recent years. The reason for this development is twofold (Fleischmann *et al.*, 2004): On the one hand, recent advances in communication and information technologies allow real time processing of information which is subject to permanent change. On the other hand, there is increasing demand from customers for fast and flexible fulfillment of their transportation requests. To meet these requirements, planning methods are needed which are able to intelligently react to dynamically revealed information.

This contribution reports on a genetic algorithm (GA) based approach for solving the dynamic pickup and delivery problem with time windows (DPDPTW) which is a generalization of the dynamic vehicle routing problem with time windows (DVRPTW, Larsen, 2000; Lund *et al.*, 1996). Based on the definition of the (static) pickup and

International Journal of Physical Distribution & Logistics Management
Vol. 35 No. 5, 2005
pp. 362-383

0960-0035
DOI 10.1108/09600030510607346

delivery problem with time windows (PDPTW, Savelsbergh and Sol, 1995), the DPDPTW can be described as follows.

All transportation requests of a given planning period, e.g. a single day, have to be satisfied by a given fleet of vehicles. The requests are not completely known in advance, but become available during the planning period. Each request is characterized by its pickup location (origin), its delivery location (destination) and the size of the load that has to be transported from the origin to the destination. For each pickup and delivery location, a time window and loading and unloading times are specified. The load capacity, the maximum length of its operating interval, a start location and an end location are given for each vehicle. In order to fulfill the requests, a set of routes has to be planned such that each request is transported from its origin to its destination by exactly one vehicle.

The characteristic feature of the DPDPTW by which it differs from its static counterpart is the fact that a portion of the requests arrive successively during the planning period. In other words, relevant information is revealed gradually during execution of the routes. As a consequence, planning and execution are overlapping processes, and planning decisions, which may be irreversible, has to be taken before all problem data become known.

Applications of the DPDPTW range from local area courier services to less than truckload (LTL) transportation and long-distance haulage. A special case of the DPDPTW, where a vehicle can only transport a single request at a time, is the dynamic single load pickup and delivery problem with time windows (DSLPDPTW), also referred to as dynamic truckload pickup and delivery problem with time windows (DTPDPTW). In addition, the DPDPTW also models typical situations in public transit. For example, the dynamic dial-a-ride problem with time windows (DDARPTW) is a DPDPTW in which people instead of goods are to be transported.

For the purposes of this investigation, some additional assumptions are made to the general problem description given above.

First, the dynamic arrival of requests is considered to be the only source of dynamics. Other events, e.g. the breakdown of a vehicle, the occurrence of a traffic jam etc. are not taken into account here. Second, it is assumed that no stochastic information is given or gathered which could be exploited in order to predict the arrival times or any other properties of future requests.

The solution approach proposed in this paper is designed to solve DPDPTW instances with a heterogeneous fleet of vehicles, i.e. the algorithm is able to cope with situations in which the vehicles are different regarding their load capacity, their operating interval, and their home location. Nevertheless, this feature could not be exploited when testing the proposed approach. This is due to the fact that most of the approaches to the static PDPTW found in the literature are based on the assumption that all vehicles have the same capacity and the same home location where all routes start and end. Thus, in order to make it easier to evaluate the approach when referring to standard benchmark instances from the literature, the evaluation of the approach was carried out under the assumption of a homogeneous fleet of vehicles. Note that this simplification does not compromise the algorithm's ability to deal with a heterogeneous fleet of vehicles. In addition, we assume that requests cannot be refused and that subcontracting is not allowed. Thus, in order to guarantee feasibility

of the benchmark instances, we consider the fleet of vehicles to be unlimited. This assumption is also in line with most of the previous approaches to the PDPTW.

Further, in order to make it easier to systematically investigate the behavior of the proposed approach under varying dynamic conditions, we have opted for a one-dimensional objective function, taking into account the total travel distance as the only performance measure. Delays are forbidden, which requires some precautions when generating the test instances. The number of vehicles used, while relevant in principle, is disregarded for the purposes of this contribution but will be taken into consideration in future stages of our research.

Dynamic VRPs are solved in a sequential fashion by repeatedly updating the current plan over a rolling horizon. Basically, the dynamic problem is decomposed into a sequence of static subproblems, which in turn are solved by a static algorithm. Thus, any approach for solving a dynamic problem has to take a decision about the static subproblem that is to be solved when updating the current plan. Further, it must specify which algorithm is applied for solving the static subproblems and at which events a plan revision is triggered off. The approach proposed in this contribution considers the static PDPTW as a static subproblem, using the same objective as for the DPDPTW. This subproblem is solved by a GA. In order to solve the dynamic problem, the GA is (re)started every time new requests arrive. When updating the current plan at a given time, the GA takes into consideration all requests that have become known up to this time.

The rest of the paper is organized as follows. Section 2 gives a brief review of previous work on the DPDPTW and similar problems. In Section 3 the GA is outlined which is used as a basic heuristic for solving the (static) PDPTW. Section 4 supplies a detailed description of the dynamic simulation environment. Section 5 reports computational results obtained by the proposed GA based approach for 5,600 DPDPTW instances which have been generated on the basis of 56 publicly available instances for the PDPTW. Finally, the main findings of the study are summarized in Section 6.

2. Literature review

In this section, a brief overview of important previous approaches for solving dynamic vehicle routing problems is given. With respect to the main topic of this investigation, this survey focuses on approaches to the DPDPTW and its variants.

Since the early 1970s, dynamic variants of the VRP are discussed in the literature. The approaches reported in the literature strongly differ by the underlying design decisions, concerning, e.g. the definition of the static subproblems, the algorithms applied to solve the subproblems, the choice of the planning horizon, and the events that trigger the planning process. An early survey of dynamic vehicle routing is provided by Psaraftis (1988). A further survey paper by the same author was published seven years later, taking into account more recent developments (Psaraftis, 1995). An overview of dynamic vehicle routing approaches focusing on local area routing is given by Gendreau and Potvin (1998). In their comprehensive survey of Pickup and Delivery Problems, Savelsbergh and Sol (1995) also consider dynamic variants of this problem class. A recent review of methods for dynamic VRPs which puts particular emphasis on parallel computing strategies is provided by Ghiani *et al.* (2003).

Among the algorithms for dynamic VRPs, insertion heuristics are clearly the most popular, because new requests can be easily integrated into the current routes by these heuristics. Early applications of insertion algorithms to the DDARP are reported by Wilson *et al.* (1971), Wilson and Weissberg (1976) and Wilson and Colvin (1977). For solving a DDARPTW, Madsen *et al.* (1995) adapt a static insertion procedure proposed by Jaw *et al.* (1986). Savelsbergh and Sol (1998) address a DPDPTW which models a typical planning situation in LTL transportation. In this approach, plan revisions are carried out at regular time intervals using an insertion procedure. Yang *et al.* (2004) investigate several real-time planning policies for a DSLPDPTW. These policies perform complex plan revisions, partially by the help of an insertion procedure. Fleischmann *et al.* (2004) develop an insertion algorithm which is used for solving a special extension of the DSLPDPTW. In addition to the dynamic arrival of requests, they also consider fluctuating travel times as a further source of dynamics.

In some approaches, the static subproblem is defined as an assignment problem. Often the planning horizon considered by these approaches is shorter when compared to insertion algorithms. Psaraftis (1988) describes a specialized assignment heuristic for solving a military sealift emergency problem which can be interpreted as a DDARPTW. In addition to their insertion algorithm mentioned above, Fleischmann *et al.* (2004) also make use of an assignment algorithm which at each single event determines an optimal assignment, taking into account all open orders and all vehicles. Another application of an exact algorithm at each event for solving the static subproblem is reported by Psaraftis (1980) who solves a dynamic dial-a-ride problem without time windows (DDARP) by repeatedly applying a dynamic programming algorithm each time a new request arrives.

In addition to using more elaborate techniques, Fleischmann *et al.* (2004) also make use of two simple rules for the assignment of requests when solving their extension of the DSLPDPTW. These rules in each plan revision just determines the very next action of each vehicle without maintaining a comprehensive plan. Similarly, Swihart and Papastavrou (1999) investigate several policies for the single-vehicle pickup and delivery problem without time windows (DPDP). The results prove the so-called nearest neighbor policy to be successful in many cases. Under this policy, after a request has been delivered, the vehicle proceeds to the nearest pickup location.

Until now only a few metaheuristics have been reported for the DPDPTW. Gendreau *et al.* (1998) propose a tabu search heuristic for the DPDPTW featuring a neighborhood definition based on the concept of ejection chains. The authors make use of a so-called adaptive memory, which contains a number of the best solutions found thus far. After the search has terminated, a new solution is constructed from the routes in the memory, serving as a new starting solution. The algorithm is tested under different operating scenarios. In addition, the authors provide a parallel implementation of their algorithm.

Although the application of GAs and other evolutionary algorithms in dynamic environments attracts growing attention (Bierwirth, 1999; Branke, 2002, 2003), to the best of our knowledge, up to now no such algorithm has been published for the DPDPTW. However, Jih and Hsu (1999) have developed a hybrid GA for solving the static single-vehicle PDPTW, which they pronounce to be appropriate for dynamic scenarios also. The initial population contains a set of routes generated by a dynamic programming algorithm, which is aborted prior to completion. Unfortunately, the

authors do not provide a description of the dynamic planning environment nor do they present any results for the dynamic case. Thus, it is an interesting task to verify the appropriateness of GAs for solving the DPDPTW. In particular, GAs appear attractive with respect to dynamic problems like the DPDPTW because of their population-based approach, which allows the parallel exploration of different paths through the solution space of a problem. The fact that the population contains a large number of different solution fragments for the GA to choose from can significantly contribute to the required planning flexibility with respect to unforeseen events. Apart from this, GAs are considered to be robust and easy to implement (Rayward-Smith, 1994).

3. A grouping genetic algorithm for solving the PDPTW

This section gives a rough description of the GA for solving the static subproblems of the DPDPTW. Further details of this algorithm can be found in Pankratz (2005). Throughout the following description it is assumed that the reader is familiar with the terminological and conceptual basics of GAs. The concept of GAs was first introduced by Holland (1975). Introductory treatments of GAs as a class of optimization algorithms can be found in Goldberg (1989) and Davis (1991). Because the proposed GA employs an adaptation of the group-oriented genetic encoding introduced by Falkenauer (1998), it is denoted as a grouping genetic algorithm (GGA).

In the following, the group-oriented genetic encoding for PDPTW solutions is explained first. After this, some configuration details of the GGA are outlined.

3.1 Group-oriented genetic encoding

The way in which solutions to the problem at hand are encoded is crucial for the success of any GA. One difficulty of finding a good genetic encoding for solutions to the PDPTW is the fact that the PDPTW – like the VRPTW – is a combination of two interdependent subproblems: On the one hand, the subproblem of clustering requests and assigning them to a vehicle has to be solved (clustering or grouping problem). On the other hand, for each cluster of requests, a feasible route has to be specified in which the corresponding pickup and delivery nodes are visited (ordering or routing problem). It is not obvious how these two aspects of a solution can be represented simultaneously by a homogeneous encoding such as the standard binary representation of the classic GA.

Alternatively, a representation could be chosen which encodes a solution as a permutation of all requests. At the phenotype level, a chromosome of this kind can be decoded by a heuristic which constructs a solution considering the requests in the order given by the permutation. In the GA literature this rather indirect encoding is quite popular. Applications to the VRP and the VRPTW are reported for example by Kopfer *et al.* (1994) and Blanton and Wainwright (1993), respectively. Nevertheless, this order-based representation with a heuristic decoder has some important drawbacks, especially for problems with a strong grouping component (Falkenauer, 1998). In particular, identical solutions are often represented by a large number of different permutations, and the phenotypical meaning of a gene strongly depends on its genotypical context, e.g. the genes that precede it in the permutation which makes it hard for the GA to sample meaningful building blocks.

With regard to these disadvantages, a different way to encode solutions of the PDPTW was chosen here. It traces back to the question if probably one of the

subproblems of the PDPTW has more influence over the solution quality than others. If so, it appears reasonable to concentrate on this dominating aspect when seeking an appopriate representation. Obviously, the grouping part of the PDPTW is of great importance for the overall quality of a solution to the PDPTW (Savelsbergh and Sol, 1998). In particular, because both the time windows and the precedence constraints considerably restrict the number of routing alternatives for a given allocation of requests, the grouping aspect turns out to be dominant over the routing aspect in many cases. Because of this dominance of the grouping part of the PDPTW, a representation of solutions to the PDPTW is proposed which refers to the encoding principles of the so-called GGA introduced by Falkenauer (1998). In this representation, each gene represents a group of objects instead of a single object. Thus, the building blocks that are sampled and recombined by the genetic operators to produce solutions of potentially higher quality are formed by these groups. Falkenauer originally developed his GGA for pure grouping problems such as the bin packing problem (BPP). The following describes how the original encoding is adapted to the PDPTW.

Each gene in a chromosome represents a cluster of all requests that are assigned to a single vehicle. To be able to distinguish between the vehicles in the case of a heterogeneous fleet of vehicles, each vehicle must have an identity. The length of a chromosome, i.e. the number of genes is variable and depends on the number of vehicles required by a given solution. Figure 1 shows the encoding of a solution by means of a simple example with seven requests that are carried out by three vehicles.

By representing a partitioning of all transportation requests, a chromosome under this encoding only covers the grouping aspect of a PDPTW solution. As a consequence, the remaining routing aspect of a solution has to be added while decoding the chromosome. This is done by associating each chromosome with separate data structures containing the routing information for each gene. Since this information is hidden from the GGA, it cannot be manipulated directly by the genetic operators. Instead, the routes are constructed and maintained using a separate heuristic according

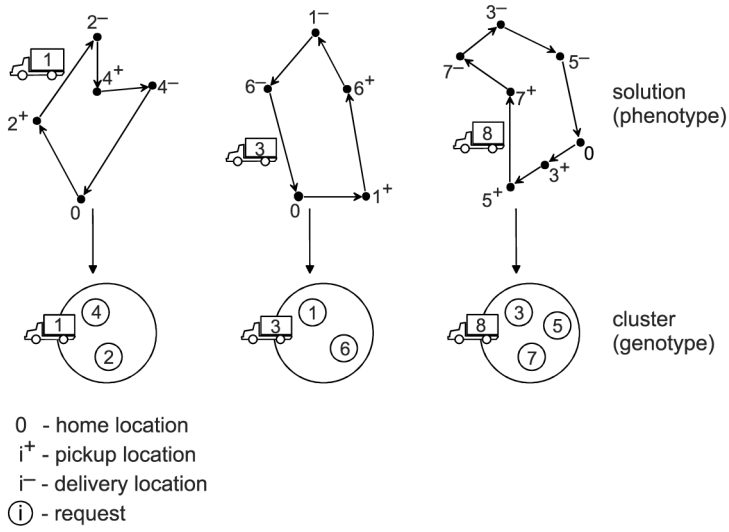

Figure 1.
Group-oriented encoding (example)

to the grouping information encoded in the chromosome. For the purposes of the proposed approach, an heuristic similar to the insertion procedure presented by Jaw *et al.* (1986) was chosen. This procedure appears attractive because of its high degree of freedom when inserting a request. The general procedure of this heuristic can be described as follows: for each request to be inserted, all feasible insertions in all existing routes of the current (partial) solution are examined. To this end, all possible insertion positions for the pickup and the delivery node in a route are tested taking into consideration precedence, capacity and time constraints. Additionally, a new vehicle is allocated and a new route for this vehicle is tentatively initialized with the current request. Among all feasible insertions, the one that causes minimal additional cost is selected and implemented.

3.2 Configuration of the GGA

The general search scheme of the proposed GGA is shown thus:

Input: parameters n^{pop}, p^{cross}, p^{mut}
begin
initialize a population P of size n^{pop};
while not (termination criterion is met)
select a pair of individuals x, y from P as parents with regard to their fitness value;
generate two children x', y' applying the crossover operator to x and y with probability p^{cross};
generate two modified children x'', y'' applying the mutation operator to x' and y', respectively, with probability p^{mut};
insert x'' and y'' into P and in turn remove the two worst individuals from P;
return best individual from P as solution.
end

An initial population is generated using the embedded insertion heuristic described above, considering all transportation requests in random order, starting with a single, empty vehicle. After all transportation requests have been inserted, a feasible initial solution is available. The procedure is repeated until an initial population of the desired size is obtained.

The population management is done following the steady-state approach without duplicates (Syswerda, 1989; Whitley, 1989). According to this approach, each newly generated pair of offspring is inserted immediately into the current population where it replaces the two worst individuals. This incremental approach ensures the survival of the best solution over the whole search and prevents the occurrence of duplicate individuals. Thus, it often reveals higher performance than generational replacement (Davis, 1991). In order to select the parent individuals, the binary tournament selection mechanism is applied (Goldberg *et al.*, 1990). In this scheme, two individuals are chosen randomly and the one, which represents the better solution, is selected as the first parent. To obtain the second parent, the procedure is repeated. The search terminates after a given total number, $\tilde{n}^{max}$, of individuals has been generated without

improvement but no later than after a given maximum number, $n^{\max}$, of generated individuals has been reached.

The crossover operator for the proposed GGA is an adaptation of the general group-oriented crossover scheme presented in Falkenauer (1998) and proceeds in five steps, as shown in Figure 2:

(1) Specify a crossing section, i.e. a coherent segment of clusters, by randomly selecting two crossing points in each of the two parent chromosomes.

(2) Insert the clusters in the crossing section of the second parent at the first crossing point in the chromosome of the first parent. The respective routes are directly adopted from the second parent without reconstruction. As a result of this operation a number of requests may occur twice (requests 1, 2, 6 and 7 in Figure 2). Moreover, two clusters may be assigned to the same vehicle (vehicle 1 in Figure 2). Note that in case of a heterogeneous fleet of vehicles, it must be made sure that each vehicle is used only once.

(3) To resolve the conflicts mentioned above, remove all clusters originally belonging to the first parent if they refer to a vehicle that is already allocated by any of the newly inserted clusters. If afterwards some requests still occur twice,

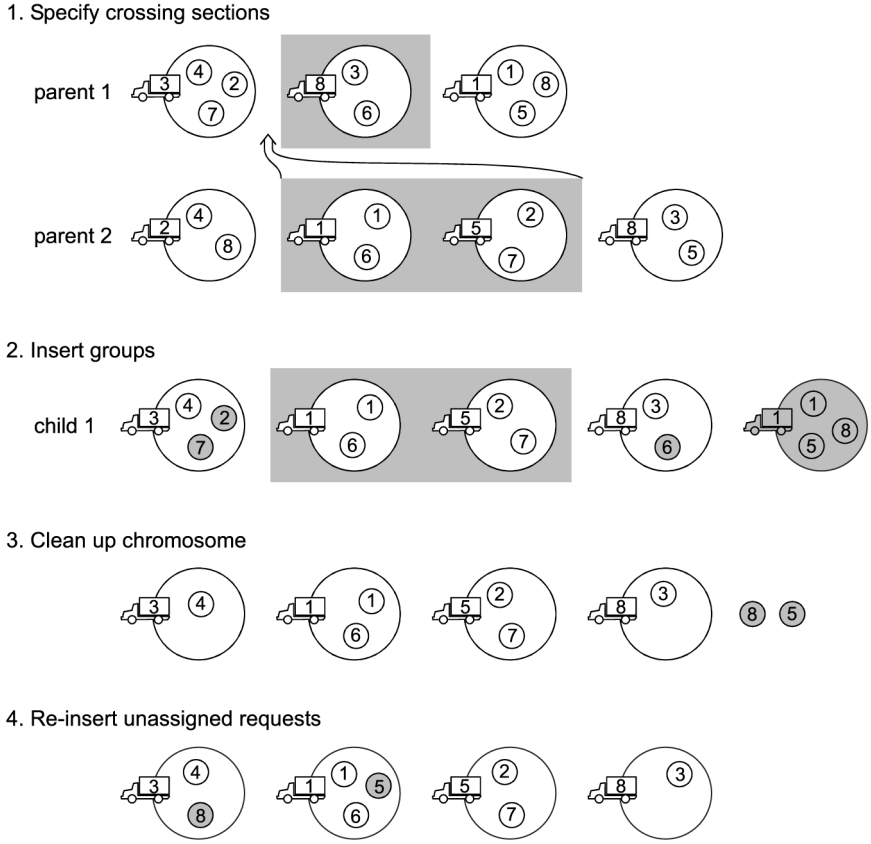

Figure 2.
The group-oriented crossover operator

eliminate them from those clusters that originally belong to the first parent. At the phenotype level, all eliminations are reproduced immediately in the routes appendant to the changed clusters. Note that the clusters and the respective routes imported from the second parent are left unchanged. After this step, some requests may remain unassigned (requests 5 and 8 in Figure 2).

(4) Reinsert the unassigned requests in random order into the individual applying the insertion heuristic. This may require the allocation of an additional vehicle in order to assure feasibility. As a result of this step, a complete offspring is obtained.

(5) Generate the second offspring by repeating steps 2 through 4 with reversed roles of the parents.

The group-oriented mutation operator for the PDPTW works as follows:

(1) select a cluster at random in the individual;

(2) eliminate this cluster from the chromosome and remove the associated route from the phenotype; and

(3) re-insert all removed requests into the individual by means of the insertion heuristic, allocating a new vehicle if necessary to maintain feasibility.

In order to re-adjust the routing part of a solution due to grouping modifications caused by the genetic operators, the insertion procedure is also used. Instead of constructing a new solution from scratch every time the grouping is modified by crossover or mutation, each request that is subject to regrouping is deleted from its original route and reinserted without disturbing the relative order in which the remaining nodes are visited. Thus, the original routing decisions in the unchanged parts of a solution are preserved.

4. Embedding the GGA in a dynamic simulation environment

In order to be able to solve the (dynamic) DPDPTW, the proposed GGA for solving the (static) PDPTW is embedded in a rolling horizon simulation framework. This section provides a description of the dynamic planning environment. First, an overview of the dynamic planning procedure is given. After that, the mechanisms for both snapshot generation and synchronization are described in detail.

4.1 Overview of the rolling horizon framework

The course of the rolling horizon simulation procedure can be outlined as follows: Compare this with the dynamic simulation environment. In the beginning, the individuals for the initial population are created taking into account all requests which are known beforehand. Subsequently, the GGA is applied to the initial population. After the search has terminated, the best solution found so far constitutes the basis for the implementation process until the occurrence of new requests. Every time a new request arrives, a snapshot of the physical transportation process is taken in order to determine the current state of execution. Note that for the simulations carried out in this study, we assume the routes to be executed exactly as planned, so the snapshot can be done referring to the best solution found so far. Then, a new population is constructed by carefully synchronizing and updating the individuals in the population in order to reflect the new situation. The main idea behind this is to preserve as much as possible of the solution fragments, which the individuals in the population have

developed so far. Finally, the GGA is restarted using the adapted population as the new initial population.

The dynamic simulation environment:

Input: set of advance requests R_0
begin
initialize time index $i := 0$;
initialize termination flag *stop* := false;
initialize start population $P(t_0)$ on the basis of R_0;
while stop := false
apply GGA to $P(t_i)$;
pick best solution $\Phi^*(t_i)$ from $P(t_i)$ as the basis for execution;
wait until next event e occurs
if e="arrival of new requests R^{new}"
$i := i + 1$;
set t_i equal to the arrival time of the new requests R^{new};
take a snapshot of the current state of execution $M(t_i)$ from $\Phi^*(t_i)$;
initialize $P(t_i)$ by synchronizing the individuals in $P(t_{i-1})$ with $M(t_i)$;
update $P(t_i)$ by inserting R^{new} into all individuals in $P(t_i)$;
else (e="end of simulation")
stop := true;
end

Following a suggestion of Bierwirth (1999), Figure 3 shows the interaction between the planning system (i.e. the GGA) and the execution system (i.e. the transportation process) by two interconnected cycles which alternately affect the population: in the upper cycle, the GGA manipulates the individuals in the population by the application of the genetic operators selection, crossover and mutation. In the lower cycle, the current best solution in the population is used as the basis for execution, and the individuals in the population are adapted to the current state of the execution system every time the situation has changed. Owing to this process of permanently adapting and re-using the individuals in the population, the population can be understood as an adaptive memory which is shared by both the planning and the execution system (Bierwirth, 1999).

Obviously, the process of snapshot generation and synchronization plays an important role in the solution methodology proposed here. The following sections will look more closely at these steps.

4.2 Snapshot generation

Let t be the time of the occurrence of new requests. Then a snapshot of the execution system is generated according to the following four-stage procedure:

(1) Create a copy of the current best individual in the population found so far.

(2) Fix the route position of all nodes which have already been served at time t. A node in a route is considered to have been served at time t if the vehicle which

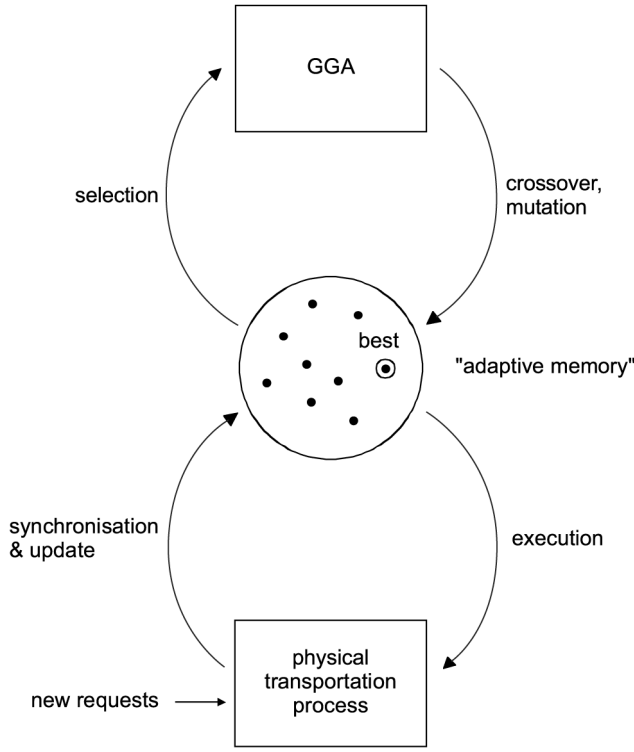

Figure 3.
The population of the GGA coupling planning with execution

serves the route has already visited this node or has already left the predecessor node for this node at time t.

(3) Fix the vehicle assignment of all active and finished requests in the solution copy. A request is considered to be finished at time t if both the position of its pickup location and the position of its delivery location are fixed at time t, whereas it is called active at t if its pickup node is fixed but its delivery node is not yet fixed at that time.

(4) Delete all requests from the copied solution which are still free for replanning at time t, i.e. which are neither finished nor active at that time. These requests are called fully disposable at time t. If a route no longer contains any requests after this step, delete the corresponding group in the copied individual and release the associated vehicle.

As a result, a truncated solution is obtained which merely contains the irreversible parts of the original best solution. This solution trunk represents the solution parts, which must be implemented by all future solutions in order to maintain feasibility. Therefore, it serves as a pattern for the subsequent synchronization of the individuals in the population. Figure 4 shows the generation of the truncated solution with a simple example. In this example, all active and finished requests are highlighted as gray circles.

4.3 Synchronization and update

When synchronizing the individuals, the goal is to adapt all individuals in the population to the current state of execution. As stated above, this should be done in a conservative manner in order to utilize the inherent solution knowledge which each individual in the population has gathered in previous GGA runs. As far as this knowledge concerns those parts of the problem that can still be revised, this knowledge is still valid after the restart of the GGA and can be of great value when creating new solutions. In the proposed GGA, the solution knowledge gathered by the individuals concerns the information about promising groupings of the requests. Therefore, we propose a straightforward synchronization mechanism, which tries to preserve as much of the grouping information originally encoded in the individuals as possible. This mechanism, which is applied to each individual in the population at time t, proceeds in six steps as follows. Note that the principle of maintaining the original grouping information is carried out by the steps 2 and 3.

(1) Create a copy T of the solution trunk generated in the snapshot procedure.

(2) Pick the first route of the solution represented by the individual I to be synchronized.

(3) Select all routes from T which have at least one fixed request in common with the current route of I.

(4) Insert as many fully disposable requests as possible from the current route of I into the routes of T which have been selected in step 3, applying the insertion heuristic described in Section 3.1. If there is no such route in T, i.e. the selection performed in step 3 is empty, allocate an additional vehicle and assign all requests under consideration to this extra vehicle.

(5) Repeat steps 3 and 4 until all routes of I have been handled. After this, a number of fully disposable requests may remain unassigned because they could not be feasibly inserted in step 4.

(6) Try to insert the unassigned requests into the routes of T regardless of the occurrence of common requests. This again may require the allocation of an additional vehicle in order to assure feasibility. Nevertheless, even this insertion may fail because after synchronization, the new schedule no longer gives enough leeway to integrate a highly urgent request. In this case, both T and I are discarded. Otherwise, synchronization was successful, and I is replaced by T in the new population.

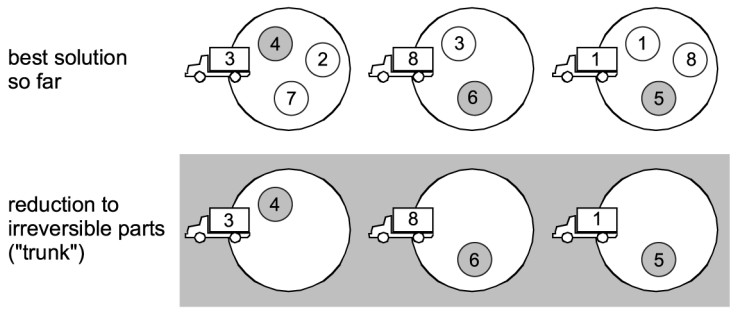

Figure 4.
Generation of the truncated solution

As a result, an individual is obtained which, on the one hand, is fully adapted to the current state of execution and, on the other hand, still exhibits similarities to the former individual. The synchronization scheme is shown in Figure 5, which continues the example of Figure 4. In this example, vehicle no. 3 has no request in common with any vehicle of the trunk, so a new vehicle is allocated incorporating all the requests of vehicle no. 5.

Note that at this stage, the population may contain fewer than n^{pop} individuals due to the possibility of synchronization failure. In order to compensate for this loss in population size, a number of additional individuals is generated by randomly inserting all disposable requests into a copy of the truncated solution.

Subsequent to the synchronization each individual in the new population is updated by inserting all requests that have newly arrived at time t.

5. Computational study

This section reports the results of the proposed approach for a number of benchmark instances when compared to two competing heuristics. First, the benchmark data sets are described, then the numerical results are discussed.

5.1 Benchmark data sets

Since there are no benchmark data sets available in the literature for the DPDPTW, a total number of 5,600 different DPDPTW instances was generated. These instances were derived from 56 static PDPTW instances provided by Li and Lim (2001), which in turn are based on Solomon's 56 100-customer VRPTW instances (Solomon, 1987). Each of the generated DPDPTW instances comprises between 100 and 106 nodes, which is equivalent to 50-53 requests. Following the classification of Solomon, Li and Lim organized their instances into six classes, denoted as LC1, LC2, LR1, LR2 LRC1 and LRC2, indicating the respective spatial distribution of the nodes and the length of the scheduling horizon. In LC1 and LC2 instances, nodes are clustered, whereas in LR1 and LR2 problems, they are randomly distributed. The nodes in LRC1 and LRC2 instances are partially clustered and partially randomly distributed. While the instances in LC1,

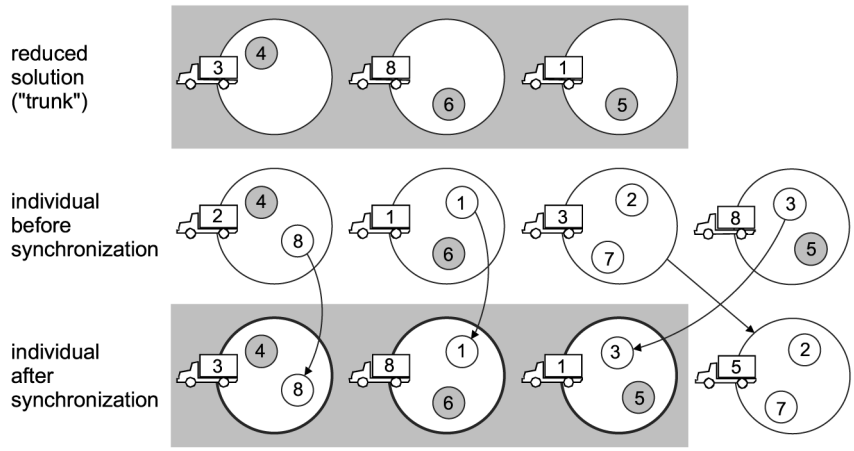

Figure 5.
Synchronization of individuals

LR1 and LRC1 have a short scheduling horizon, the horizon is longer in instances of classes LC2, LR2 and LRC2.

In order to obtain dynamic instances from the static PDPTW instances, an arrival time stamp was assigned to each request, leaving the rest of the instance data unchanged. During the simulations, each request was announced to the system according to its specific arrival time stamp. The generated DPDPTW instances are divided into two sets with different dynamic properties.

The first data set (P1) contains problem instances with varying degrees of urgency, i.e. when generating these instances, the time interval left to react to a new request after its announcement was varied. Before generating these instances, for each request r, the latest possible arrival time t_r^{latest} was determined such that feasibility is provided. Let $t^{\text{close}}(v_r^+)$ and $t^{\text{close}}(v_r^-)$ be the due times of the pickup and the delivery time window of r, respectively, $t^{\text{serve}}(v)$ the service duration at a given location v, v^{home} the home location of the vehicles, and $t(u, v)$ the time required to travel between locations u and v. Then, t_r^{latest} can be defined as:

$$t_r^{\text{latest}} := \min\{t^{\text{close}}(v_r^+), t^{\text{close}}(v_r^-) - t(v_r^+, v_r^-) - t^{\text{serv}}(v_r^+)\} - t(v^{\text{home}}, v_r^+),$$

which is the latest time at which a vehicle must depart from its home location in order to be able to reach both the pickup and the delivery location of the request just in time. Afterwards, each request was given a time stamp $t_r = a \cdot t_r^{\text{latest}}$, with a varying from 0.1 to 1.0 in steps of 0.1. According to these principles, from each of the 56 static PDPTW instances ten DPDPTW instances have been generated. Note that in all the instances of this set, no requests are known in advance.

In contrast to P1, the second data set (P2) consists of instances with varying degrees of ex-ante knowledge. They were generated by increasing the portion q of requests known in advance stepwise by 0.1, with q ranging from 0.1 to 0.9. In the remainder, these requests are referred to as static requests, whereas all other requests are called dynamic. All static requests, which were selected randomly from the total set of requests of the respective static instance, received a time stamp $t_r = 0$. The dynamic requests were given the highest possible degree of urgency by choosing a time stamp which is equal to the latest possible arrival time, $t_r = t_r^{\text{latest}}$. In order to reduce the risk of stochastic bias when picking the static requests, from each static instance, ten dynamic instances were generated for a given value of q. Under this generation scheme, a total number of 5,040 instances have been generated for instance set P2.

In order to illustrate the differences between the two data sets, Figure 6 shows the characteristics of two dynamic instances that were derived from the static instance LC201. The instance to the left is taken from data set P1, whereas the instance to the right comes from data set P2. Both instances represent a medium degree of urgency or ex-ante knowledge, respectively. For all the instances shown, the length of the time horizon is 3,390. For each point in time, the accumulated percentage of requests that have already become known to the system are plotted. Additionally, upper bounds for both the portion of known requests that are not yet completed and the portion of requests that still are fully disposable at a given time are provided. The former is calculated by subtracting all requests for which the delivery time windows have already closed at that time, whereas the latter results from deducting all requests for which the pickup time windows have already closed.

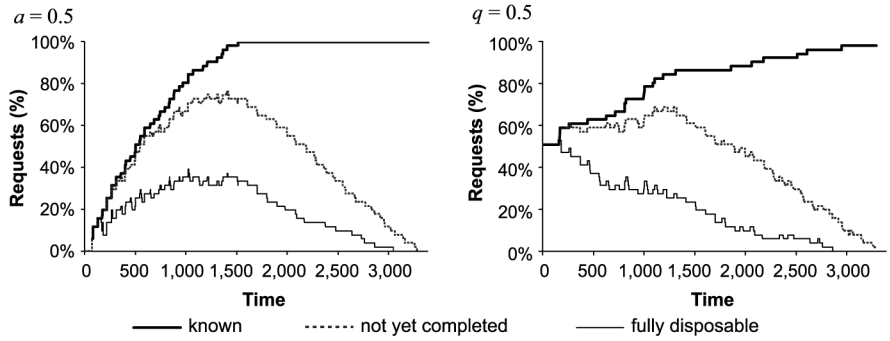

Figure 6.
Characteristics of two medium-dynamic sample instances

Figure 6 reveals that under the conditions of data set P1, the number of fully disposable requests, starting from a low level, first increases and then decreases, whereas in the case of data set P2, the number of disposable requests starts at its maximum and decreases throughout the planning period.

5.2 Numerical results

The proposed GA-based approach was evaluated using the 5,600 problem instances of data set P1 and data set P2. The parameter settings for the GGA, which were determined in a small number of preliminary experiments, are shown in Table I. The values were chosen mainly with the idea of performing a thorough search but also to keep computation times reasonable. As a guide, it should take the system less than 60 seconds, on average, to carry out a plan revision after the arrival of new requests.

The solution quality of the GA-based approach is measured in terms of cumulative total travel distance required to serve all requests of a given instance. In order to judge the solution quality of the approach, the results obtained by two variations of the insertion heuristic described in Section 3.2 are used as a standard of comparison.

The first heuristic, in the following denoted as H1, inserts each new request immediately after its arrival without fundamentally changing the routes of the previous plan, i.e. the assignment of requests to vehicles and the relative order of the locations in the routes are left unchanged. Thus, the plan revisions performed by this heuristic can be characterized as incremental. For each instance of data set P2, an initial solution is generated by inserting all advance requests in the order of ascending values of t_r^{latest}, starting with an empty plan.

By contrast, the second heuristic, referred to as H2, performs a total plan revision every time new requests arrive: all decisions of the previous plan, except for those

Parameter	Value	Description
n^{pop}	150	Maximum population size
p^{cross}	1.0	Cross-over probability
p^{mut}	0.5	Mutation probability
n^{max}	1,500	Maximum number of individuals
$\tilde{n}^{\text{max}}$	500	Maximum number of individuals without improvement

Table I.
GGA parameter values

which have been fixed due to the current state of execution, are discarded, and a new plan is generated from scratch by inserting all disposable requests in the order of ascending values of t_r^{latest}.

All algorithms were implemented in JAVA using JDK 1.4. The tests were carried out on a PC (Pentium 4 processor, 2 GHz) under the objective of minimizing total travel distance. The results of the GA-based approach were obtained by a single run on each instance.

On average, the GA-based approach requires computing times of less than 6 s for a single plan revision. Only in isolated cases have computing times of more than a minute been observed. This concerns in particular, some instances of data set P2 with a large portion of advance requests. The computing times of both H1 and H2 consistently stay below 1 seconds.

In the remainder, the results obtained by the GA-based approach will be compared to the results of the two heuristics H1 and H2, respectively.

The diagrams in Figure 7 show aggregated results for data sets P1 and P2, respectively. Note that in all curves of Figures 7-9, each point represents the average solution quality in terms of total travel distance obtained by the respective method over all instances of identical degree of dynamism. For aggregation and comparison purposes, all results have been normalized using the best objective value obtained by the (static) GGA for the corresponding (static) PDPTW instance as a reference (Pankratz, 2005).

From Figure 7 the following conclusions can be drawn:

- Although the relative solution quality of all three algorithms becomes worse when the instances become more dynamic, the GA-based approach, on average, performs significantly better than the two competing heuristics for both data sets.
- Quite naturally, this advantage of the GA-based approach diminishes with an increasing degree of dynamism. This is due to the fact that there is less and less room left to the GA-based approach to optimize because more and more requests have to be served shortly after they have arrived, so they cannot be replanned even if it turns out later that they have been scheduled awkwardly with respect to subsequent requests.

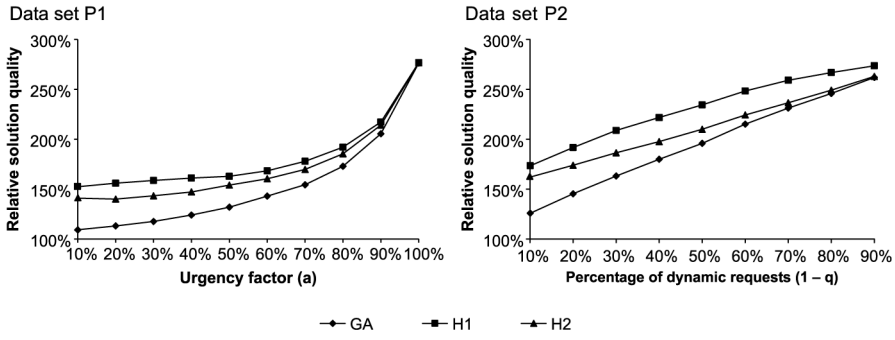

Figure 7.
Aggregated results for data sets P1 and P2, respectively

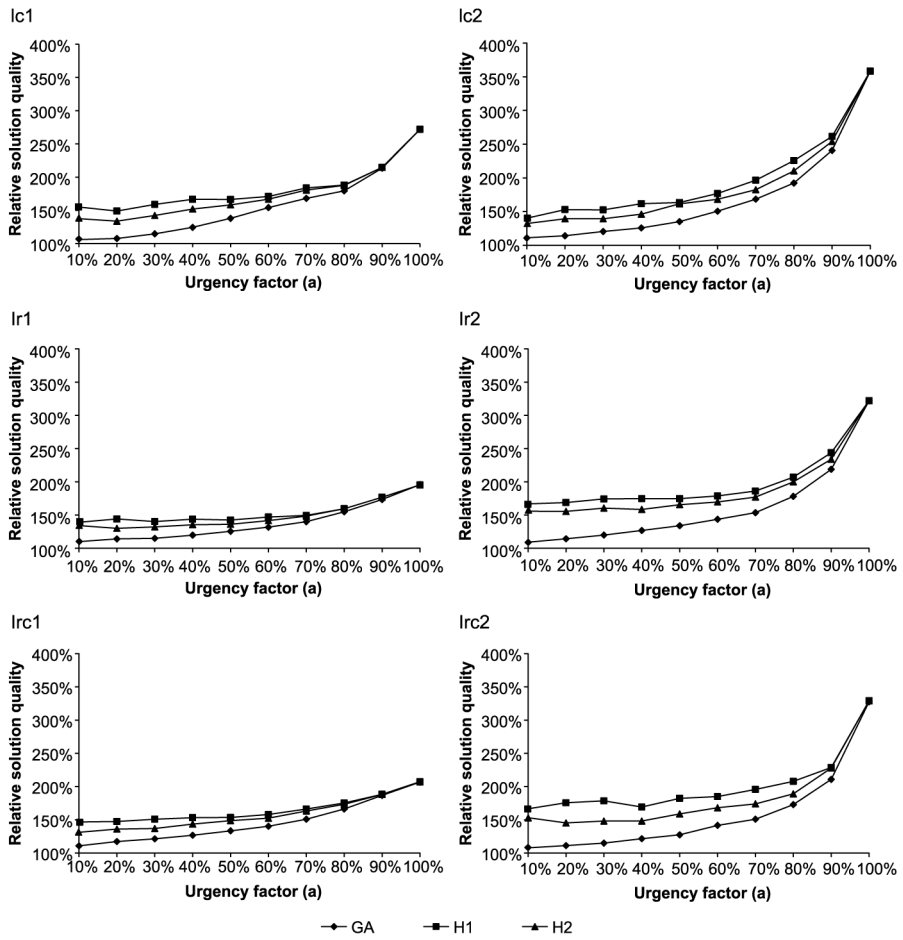

Figure 8.
Results for data P1, categorized by problem class

- Nevertheless the GA-based approach produces good results even under highly dynamic conditions. In the case of data set P1, even for very high degrees of urgency ($a = 0.9$), the GA-based approach, on average, produces results which are 5 percent better than those of H1 and 4 percent better than those of H2. Under the conditions of data set P2, the GA-based approach still performs 4 percent better than H1 when the portion of dynamic requests is 90 percent. However, when compared to H2, the advantage of the GA-based approach drops below 2 percent as soon as the portion of dynamic requests reaches 80 percent.
- When comparing H1 to H2, H2 appears predominant throughout all runs. Note that this is in line with similar observations made by other authors who state that more flexible plan revisions consistently yield better results than simple, rather incremental plan revisions (Yang *et al.*, 2004). It is particularly striking that H2 performs just slightly better than H1 in the case of data set P1, whereas

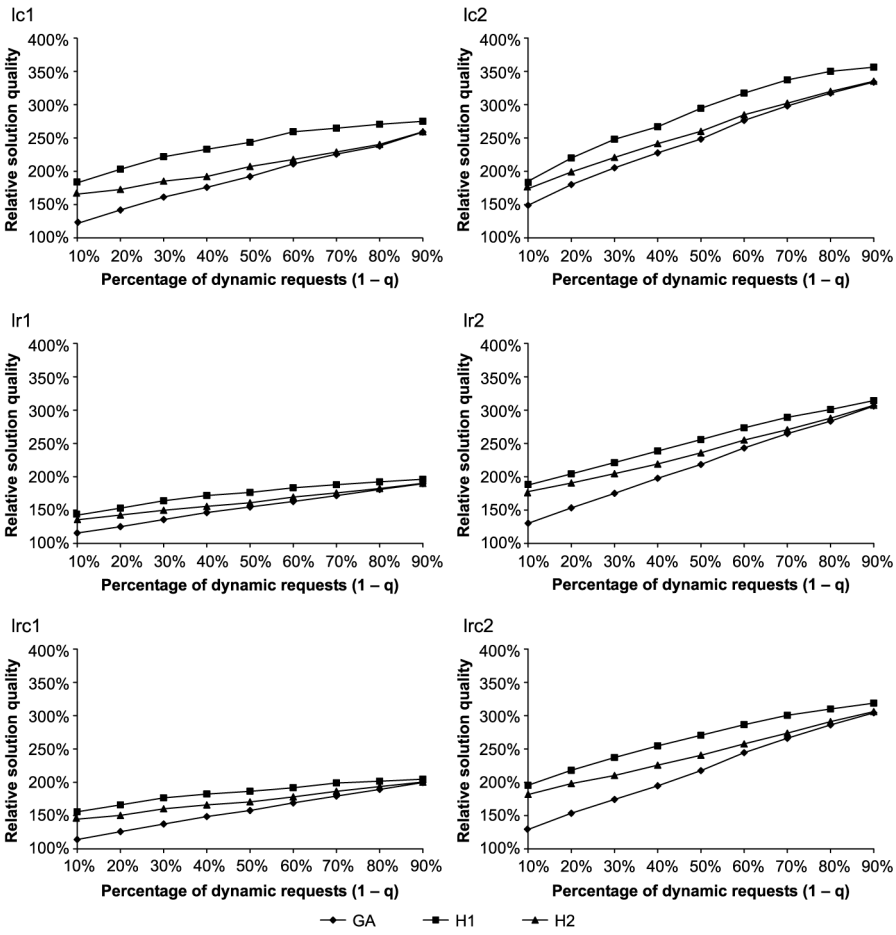

Figure 9.
Results for data P2, categorized by problem class

for data set P2, it leaves H1 far behind. Obviously, H2 can profit by its higher degree of freedom, in particular when a sediment of advance requests offers optimization opportunities.

- Interestingly, in the case of P1, the curves are comparatively flat in the beginning and ascend rapidly when the degree of urgency exceeds 70 percent, whereas in the case of data set P2, total travel distance increases almost linearly with an increasing portion of dynamic requests. This can be explained by the way the arrival times are generated for the instances in data set P1 (Larsen, 2000): because the arrival time of a request r is set relative to the value of t_r^{latest}, requests with a greater value of t_r^{latest} have a longer reaction time than those with a smaller value of t_r^{latest}. Thus, if the values of t_r^{latest} are reasonably scattered over the planning period, enough long-term requests are left, which assures considerable optimization opportunities even with values of a of about 0.5-0.7.

In order to further analyze the behavior of the GA-based approach with respect to the spatial distribution of the nodes and the length of the scheduling horizon, Figures 8 and 9 show averaged results obtained for data sets P1 and P2, respectively, categorized by problem class. The observations made by the help of these charts can be summarized as follows:

- For instances with a longer scheduling horizon (problem classes LC2, LR2, and LRC2), high degrees of dynamism lead to higher losses in solution quality than for instances with a shorter scheduling horizon (problem classes LC1, LR1, and LRC1). This holds for both P1 and P2. The reason is that in these instances, the requests are spread over a longer period of time. As a consequence, requests which arrive late encounter a situation in which a considerable portion of requests has already been fixed. Obviously, this limits the optimization opportunities.
- Interestingly, the GA-based approach behaves quite robustly under these conditions: as the curves for problem classes LR2 and LRC2 demonstrate, the GA-based approach has a considerable lead over the two comparative heuristics as long as the degree of dynamism is rather moderate.
- When analyzing the influence of the spatial distribution of the requests, it appears that if the scheduling horizon is short, instances exclusively containing clustered requests are slightly more difficult than instances where the requests are at least partially randomly distributed. This becomes particularly obvious when comparing the results of all three algorithms for problem class LC1 to those for problem class LR1 and LRC1, respectively. When the horizon is longer, one has to distinguish between P1 and P2. In the case of P2, all three algorithms perform better if the instances include randomly distributed requests, i.e. for problem classes LR2 and LRC2. In the case of P1, the GA-based approach yields better solutions for problem classes LR2 and LRC2 than for problem class LC2 if the degree of urgency is above 50 percent. Below this value the GA-based approach performs almost identically for all three problems, LC2, LR2, and LRC2. In contrast to the GA-based approach, for LR2 and LRC2, both H1 and H2 obtain results of less quality than for LC2 if the urgency factor is below 60 percent. Interestingly, up to this value, the solution quality of both H1 and H2 is nearly independent of the degree of urgency. However, above this threshold, the respective curves ascend slower than for LC2.

6. Conclusions

In this paper, a GA based approach for the DPDPTW has been presented. As a specific feature, the GA uses a group-oriented genetic encoding from Falkenauer (1998). The approach was tested using 5,600 dynamic benchmark instances which were derived from 56 publicly available benchmark instances for the PDPTW. The experimental results have demonstrated that the proposed approach is able to find high quality solutions when compared to two comparative heuristics. The overall findings seem to justify the employment of GAs for dynamic vehicle routing problems. Future research will be dedicated to the following issues:

- Generation of larger problem instances. Although the tests conducted in this contribution have provided valuable insights into the behavior of the proposed method, it would be interesting to see the behavior with problem instances comprising 700 requests. This is relevant if the algorithm is to be employed in large-scale real-world scenarios, e.g. in the dial-a-ride or truckload context.
- Development and evaluation of different objective functions. Using specialized objective functions could help make the behavior of the GA-based approach less myopic. In addition, further objective criteria, e.g. minimization of the number of vehicles used and minimization of delays are to be taken into consideration.
- Optimizing the response times of the method. Since the current prototypical JAVA implementation has not been optimized with regard to time requirements, the algorithms should be re-implemented in C or C++ using optimized data structures.
- Further improvement of the embedded GGA. In particular, the integration of fast local search techniques into the genetic search appears promising.

At the moment, the proposed GA-based approach is prepared for being integrated as a local planning component into a multi agent system for cooperative transportation planning.

References

Bierwirth, C. (1999), *Adaptive Search and the Management of Logistics Systems – Base Models for Learning Agents*, Kluwer, Boston, MA.

Blanton, J.L. and Wainwright, R.L. (1993), "Multiple vehicle routing with time and capacity constraints using Genetic Algorithms", in Forrest, S. (Ed.), *Proceedings of the 5th International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 452-9.

Branke, J. (2002), *Evolutionary Optimization in Dynamic Environments*, Kluwer, Boston, MA.

Branke, J. (Ed.) (2003), *Proceedings of the Workshop on Evolutionary Algorithms for Dynamic Optimization Problems (EvoDOP-2003)*, held in conjunction with the Genetic and Evolutionary Computation Conference (GECCO-2003), Chicago, IL, 12 July.

Davis, L. (Ed.) (1991), *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, NY.

Falkenauer, E. (1998), *Genetic Algorithms and Grouping Problems*, Wiley, Chichester.

Fleischmann, B., Gnutzmann, S. and Sandvoß, E. (2004), "Dynamic vehicle routing based on online traffic information", *Transportation Science*, Vol. 38 No. 4, pp. 420-433.

Gendreau, M. and Potvin, J.Y. (1998), "Dynamic vehicle routing and dispatching", in Crainic, T.G. and Laporte, G. (Eds), *Fleet Management and Logistics*, Kluwer, Boston, MA, pp. 115-26.

Gendreau, M., Guertin, F., Potvin, J.Y. and Séguin, R. (1998), "Neighborhood search heuristics for a dynamic vehicle-dispatching problem with pick-ups and deliveries", *Technical Report* CRT-98-10, Centre de Recherche sur les Transports, Université de Montréal, Montréal.

Ghiani, G., Guerriero, F., Laporte, G. and Musmanno, R. (2003), "Real-time vehicle routing: solution concepts, algorithms and parallel computing strategies", *Research Report*, Center of Excellence for High Performance Computing, University of Calabria, Arcavacata di Rende.

Goldberg, D.E. (1989), *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, MA.

Goldberg, D.E., Korb, B. and Deb, K. (1990), "Messy genetic algorithms: motivation, analysis, and first results", *Complex Systems*, Vol. 3, pp. 493-530.

Holland, J.H. (1975), *Adaptation in Natural and Artificial Systems – An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*, The University of Michigan Press, Ann Arbor, MI.

Jaw, J.J., Odoni, A.R., Psaraftis, H.N. and Wilson, N.H.M. (1986), "A heuristic algorithm for the multi-vehicle advance request dial-a-ride problem with time windows", *Transportation Research*, Vol. 20, Part B, pp. 243-57.

Jih, W.-R. and Hsu, Y.-J. (1999), "Dynamic vehicle routing using hybrid Genetic Algorithms", in IEEE Computer Society (Ed.), *Proceedings of the 1999 IEEE International Conference on Robotics & Automation*, IEEE Computer Society, Los Alamitos, CA, pp. 453-8.

Kopfer, H., Pankratz, G. and Erkens, E. (1994), "Die entwicklung eines hybriden genetischen algorithmus für das tourenplanungsproblem", *OR Spektrum*, Vol. 16, pp. 21-32.

Larsen, A. (2000), "The Dynamic vehicle routing problem" PhD thesis, Institute of Mathematical Modelling (IMM), Technical University of Denmark Lyngby.

Li, H. and Lim, A. (2001), "A metaheuristic for solving the pickup and delivery problem with time windows", in IEEE Computer Society (Ed.), *Proceedings of the 13th IEEE International Conference on Tools with Artificial Intelligence (ICTAI)*, IEEE Computer Society, Los Alamitos, CA, pp. 160-7.

Lund, K., Madsen, O.B.G. and Rygaard, J.M. (1996), "Vehicle-routing problems with varying degrees of dynamism", *Technical Report*, No. IMM-REP-1996-1, Institute of Mathematical Modelling (IMM), Technical University of Denmark, Lyngby.

Madsen, O.B.G., Ravn, H.F. and Rygaard, J.M. (1995), "A heuristic algorithm for a dial-a-ride problem with time windows, multiple capacities and multiple objectives", *Annals of Operations Research*, Vol. 60, pp. 193-208.

Pankratz, G. (2005), "A grouping genetic algorithm for the pickup and delivery problem with time windows", *Operations Research Spectrum*, Vol. 27, pp. 21-41.

Psaraftis, H.N. (1980), "A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem", *Transportation Science*, Vol. 14, pp. 130-54.

Psaraftis, H.N. (1988), "Dynamic vehicle routing problems", in Golden, B.L. and Assad, A.A. (Eds), *Vehicle Routing: Methods and Studies*, North-Holland, Amsterdam, pp. 223-48.

Psaraftis, H.N. (1995), "Dynamic vehicle-routing: status and prospects", *Annals of Operations Research*, Vol. 61, pp. 143-64.

Rayward-Smith, V.J. (1994), "A unified approach to Tabu search, simulated annealing and genetic algorithms", Unicom Seminars Ltd (Ed.), *Adaptive Computing and Information Processing*, Vol. 1, Unicom Seminars Ltd, London, pp. 55-78.

Savelsbergh, M.W.P. and Sol, M. (1995), "The general pickup and delivery problem", *Transportation Science*, Vol. 29, pp. 17-29.

Savelsbergh, M.W.P. and Sol, M. (1998), "DRIVE: dynamic routing of independent vehicles", *Operations Research*, Vol. 46, pp. 474-90.

Solomon, M. (1987), "Algorithms for the vehicle-routing and scheduling problem with time window constraints", *Operations Research*, Vol. 35 No. 2, pp. 254-65.

Swihart, M.R. and Papastavrou, J.D. (1999), "A stochastic and dynamic model for the single-vehicle pick-up and delivery problem", *European Journal of Operational Research*, Vol. 114, pp. 447-64.

Syswerda, G. (1989), "Uniform cross-over in genetic algorithms", in Shaffer, J.D. (Ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 2-9.

Whitley, L.D. (1989), "The GENITOR algorithm and selection pressure: why rank-based allocation of reproductive trials is best", in Shaffer, J.D. (Ed.), *Proceedings of the 3rd International Conference on Genetic Algorithms*, Morgan Kaufmann, San Mateo, CA, pp. 116-21.

Wilson, N.H.M. and Colvin, N.H. (1977), "Computer control of the Rochester dial-a-ride system", *Report*, No. R77-31, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA.

Wilson, N.H.M. and Weissberg, H. (1976), "Advanced dial-a-ride algorithms research project: final report", Report R76-20, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA.

Wilson, N.H.M., Sussman, J.M., Wong, H.K. and Higonnet, T. (1971), "Scheduling algorithms for dial-a-ride systems", *Urban Systems Laboratory Report*, No. USL TR-70-13, Massachusetts Institute of Technology, Cambridge, MA.

Yang, J., Jaillet, P. and Mahmassani, H.S. (2004), "Study of a real-time multi-vehicle truckload pickup-and-delivery problem", *Transportation Science*, Vol. 38 No. 2.

Further reading

Bell, W.J., Dalberto, L.M., Fisher, M.L., Greenfield, A.J., Jaikumar, R., Kedia, P., Mack, R.G. and Prutzman, P.J. (1983), "Improving the distribution of industrial gases with an online computerized routing and scheduling optimizer", *Interfaces*, Vol. 13, pp. 4-23.