A hybrid genetic algorithm for the heterogeneous dial-a-ride problem

A Hybrid Genetic Algorithm for the Heterogeneous Dial-A-Ride Problem

Mohamed Amine Masmoudi[a,1], Kris Braekers[b, c], Malek Masmoudi[d], Abdelaziz Dammak[a]

[a] Laboratory of Modeling and Optimization for Decisional, Industrial and Logistic Systems (MODILS),Faculty of Economics and Management Sciences, University of Sfax, Airport Street, km 4, Post Office Box 1088, 3018 Sfax, Tunisia

[b] Research Group Logistics, Hasselt University, Campus Diepenbeek, Agoralaan Gebouw D, 3590 Diepenbeek, Belgium

[c] Research Foundation Flanders (FWO), Egmontstraat 5, 1000 Brussels, Belgium

[d] Université de Lyon, F-42023, Saint Etienne, France; Université de Saint Etienne, Jean Monnet, F-42000, Saint-Etienne, France; LASPI, F-42334, IUT de Roanne

Abstract

This paper investigates the Heterogeneous Dial-A-Ride Problem (H-DARP) that consists of determining a vehicle route planning for heterogeneous users' transportation with a heterogeneous fleet of vehicles. A hybrid Genetic Algorithm (GA) is proposed to solve the problem. Efficient construction heuristics, crossover operators and local search techniques, specifically tailored to the characteristics of the H-DARP, are provided. The proposed algorithm is tested on 92 benchmarks instances and 40 newly introduced larger instances. Computational experiments show the effectiveness of our approach compared to the current state-of-the-art algorithms for the DARP and H-DARP. When tested on the existing instances, we achieved average gaps of only 0.47% to the best-known solutions for the DARP, and 0.05% to the optimal solutions for the H-DARP, compared to 0.85% and 0.10%, respectively, obtained by the current state-of-the-art algorithms. For the 40 newly generated instances, average gaps of the hybrid GA are 0.35% smaller compared to the current state-of-the-art method. Besides, our method provides best results for 31 of these instances and ties with the existing method on 8 other instances.

Keywords: Heterogeneous Dial-A-Ride Problem (H-DARP), Genetic Algorithm (GA), Construction heuristics, Local Search (LS), Hybrid algorithm.

1. Introduction

The transportation of people with reduced mobility is an important branch of day-to-day transportation. The problem of establishing a transport planning to meet particular users' demands with a limited fleet of vehicles is called the Dial-A-Ride Problem (DARP) in the literature. It is a variant of the Pickup and Delivery Problem with Time Windows (PDPTW), which is usually concerned with freight transportation. Common objectives of DARPs are to minimize the total routing costs and to maximize the service quality. The DARP is more complicated than the traditional Vehicle

[1] Corresponding author: M.A. Masmoudi (E-mail: masmoudi_aminero@hotmail.fr, Tel. : +21654493673)

Routing Problem (VRP), which is already NP-hard (Cordeau and Laporte, 2007), due to its specific transportation conditions.

The DARP has been widely studied since it was introduced by Wilson et al. in 1971. Most research concerns the management of door-to-door transportation services for the elderly, the handicapped and the disabled; e.g., the transportation of patients from their houses to hospitals or care centers (Schilde et al., 2011, Zhang et al., 2015). It is applied in many countries such as the USA (Karabuk, 2009), Belgium (Rekiek et al., 2006), Italy (Bologna) (Toth and Vigo, 1996, 1997) and Germany (Borndörfer et al., 1997). For more details on the DARP, interested readers are referred to the surveys of Cordeau and Laporte (2007), Parragh et al. (2008), and Doerner and Salazar-Gonzalez (2014).

DARP variants can be classified based on several characteristics, e.g., single vehicle or multiple vehicles, static or dynamic users' demands, and homogeneity or heterogeneity of the vehicle fleet and users' demands. The single vehicle DARP, which is a special case of the multiple vehicles DARP and mainly studied in early contributions, has been proved to be NP-hard by Psaraftis (1980). For the static DARP, all transportation demands are known in advance, while for the dynamic DARP, some requests are expressed progressively during the day and the transportation planning has to be established in real time. As in most papers, we will consider the static multiple vehicles variant. For an overview of contributions on the dynamic DARP, we refer to Berbeglia et al. (2010). The homogeneous DARP considers a single kind of users and a homogeneous fleet of vehicles. It is widely treated in the literature (Cordeau, 2006; Parragh et al., 2010; Luo and Schonfeld, 2011; Parragh and Schmid, 2013; Kirchler and Wolfler Calvo, 2013; Chassaing et al., 2016). However, we will mainly focus on the heterogeneous variant of the problem, as this is more realistic for many applications.

As indicated by Parragh (2011), in practice, service providers often use a variety of vehicles to transport users with different requirements. For example, in the context of patient transportation, a patient may demand to be transported seated, on a stretcher, or in a wheelchair. Additionally, the patient may need an accompanying person. Several authors, e.g., Wong and Bell (2006) and Xiang et al. (2006), have considered heterogeneity in DARPs for specific applications. A formal definition of the Heterogeneous DARP (H-DARP), integrating heterogeneous users and vehicles into the DARP, was introduced by Parragh (2011), where two types of vehicles and four different resources (staff seats, user seats, stretchers and wheelchair places) were considered. The author proposed 2-index and 3-index mathematical formulations for the H-DARP and developed a Branch-and-Cut (B&C) and a Variable Neighboorhood Search (VNS) algorithm for it. The algorithms were tested on 36 instances with up to four vehicles and 48 requests, generated from those proposed by Cordeau (2006) for the homogeneous DARP. Parragh et al. (2012) extended this work by considering drivers' work duration limits and lunch breaks, and provided an algorithm combining column generation with a VNS method. Qu and Bard (2013) developed an Adaptive Large Neighborhood Search (ALNS) algorithm to solve the Heterogeneous Pickup and Delivery Problem with configurable vehicle capacity (H-PDP). Braekers et al. (2014) introduced the Multi-Depot H-DARP (MD-H-DARP). They provided a 2-index mathematical formulation, and solved it exactly using a Branch-and-Cut algorithm, while heuristic

solutions are obtained by a Deterministic Annealing (DA) algorithm. The developed algorithms were evaluated on the DARP benchmark instances with up to 13 vehicles and 144 requests proposed by Cordeau and Laporte (2003) and the H-DARP instances of Parragh (2011). To the best of our knowledge, these algorithms currently provide the best results on the H-DARP instances, while best results on the DARP instances are provided on by the Evolutionary Local Search (ELS) algorithm of Chassaing et al. (2016).

The contributions of our paper are as follows: *i*) a hybrid Genetic Algorithm (GA) for the H-DARP is proposed, in which efficient heuristics are used to generate initial solutions, and adapted crossover operators based on the characteristics of H-DARP are applied. *ii*) A local search strategy is used to further enhance the best solution proposed by the GA. *iii*) Computational experiments indicate that our approach is more effective than current state-of-the-art algorithms for the DARP and H-DARP. Average gaps with the optimal or best known solutions are 0.47% and 0.05%, respectively, compared to 0.85% and 0.10% for the current state-of-the-art algorithms. *iv*) 40 new, larger, instances with up to 13 vehicles and 144 requests for the H-DARP are generated, in a similar way as the instances of Cordeau and Laporte (2003) for the DARP. Average results of the hybrid GA on these instances are 0.35% (0.26% for the data set E and 0.44 for the data set I) better than when applying the DA algorithm proposed by Braekers et al. (2014). Besides, the hybrid GA provides best results for 31 of these instances and ties on 8 other instances. *v*) Computational experiments show the positive contribution to solution quality of the proposed construction heuristics, crossover operators and local search techniques in the hybrid GA.

The rest of the paper is organized as follows. Section 2 presents a brief description of the H-DARP. Section 3 presents the developed Hybrid GA. Section 4 reports the numerical experiments. Section 5 concludes the paper and gives future research directions.

2. Problem description

In this section, we recall the H-DARP definition proposed by Parragh (2011). Consider a graph $G = (V, A)$ with a set of nodes $V = \{0,1, \dots ,2n\}$ and a set of edges $A = \{(i, j): i, j \in V: i \neq j\}$. The cost and travel time on arc (i, j) are denoted by c_{ij} and t_{ij}, respectively. The node pair $(i, i + n)$ corresponds to the pickup and delivery points of user $i = 1, \dots , n$. The depot, in which a heterogeneous fleet of K vehicles is available, is denoted by node 0. Each vehicle $k = 1, \dots , K$ has a capacity $Q^{r,k}$ for resource type r=0, 1, 2 and 3. The four resource types represent staff seats, patient seats, stretchers and wheelchairs, respectively. Let q_i^r denote the demand of user i for resource r. Similarly, user i liberates $q_{i+n}^r = -q_i^r$ at node $i + n$.

The H-DARP consists of determining a route planning for satisfying the users' demands while minimizing the total routing cost. A route planning solution should satisfy the following constraints:

(1) Each route begins and ends at the depot 0;

(2) The vehicle capacity $Q^{r,k}$(0,1,2,3 and k=1,…,K) must be respected at each node i=0,…$2n$;

(3) Each node i must be visited within its time window $[e_i;\ l_i]$ where $i \in V$ and $i \neq 0$. If a vehicle arrives early, it must wait until the beginning of the time window e_i to begin the service;

(4) Let a_i^k denote the arrival time of vehicle k at i, and s_i the service time; the departure time $b_i^k \geq \max\{a_i^k, e_i\} + s_i$;

(5) A pickup and delivery node pair $(i, i+n)$ must be visited in the same route, and the pickup node i must be visited before the delivery node $i+n$, i.e., $b_i^k \leq a_{i+n}^k$

(6) Let l_i^k=$(b_{i+n}^k - s_{i+n})$-b_i^k denote the ride duration of user i; l_i^k may not exceed the maximum ride time L_{max}, i.e., $l_i^k \leq L_{max}$;

(7) If a vehicle travel k travels along a directed edge (i, j), we have $a_j^k = b_i^k + t_{ij}$;

(8) The duration of each route is strictly limited by T_{max}.

For a detailed description and mathematical formulation of H-DARP, interested readers are referred to Parragh (2011) and Braekers et al. (2014). Note that we only consider the situation of minimizing routing costs, while waiting is allowed at no cost.

3. Developed approach

The concept of Genetic Algorithms (GAs) was first introduced by Holland (1975). The metaheuristic has been successfully used to deal with a large variety of combinatorial optimization problems, including the DARP (e.g., Cubillos et al., 2007; Jorgensen et al., 2007; Wang and Chen, 2012).

The basic GA design performs well in global search but spends much time to converge to reasonable solutions (Minocha and Tripathi, 2011). In contrast, local search techniques are often able to find optimal solutions in small search spaces very quickly. Thus, we provide a hybrid GA by incorporating effective local search techniques into a Genetic Algorithm in order to improve the convergence and reduce the computation time.

The proposed hybrid GA framework is presented in Algorithm 1, and detailed in the following subsections. Four effective heuristics are used to provide the initial population of size *N* (Subsection 3.2). Next, the hybrid GA runs for a number of iterations. In each iteration, the following steps take place. First, each individual in the population is evaluated based on its fitness (Subsection 3.3). Second, S individuals are selected by tournament selection (Subsection 3.4). Third, two parents are selected randomly from the S individuals, and two crossover operators (sequencing and assigning) are applied to these parents to produce four new solutions, called children. The two parents are then deleted from S. This process is repeated until all parents in S have been selected and then deleted (Subsection 3.5). Fourth, if an obtained solution (child) is not feasible, a reparation phase is applied until it becomes feasible (Subsection 3.6). Fifth, the new solutions (children) in the population are improved by local search techniques (Subsection 3.7). The feasible solutions are then inserted into the population. Sixth, an elitist approach is applied to guarantee an improvement from one generation to the next (Subsection 3.8). Seventh, at every generation, a mutation phase (Subsection 3.9) is applied to h individuals, with h being a randomly chosen value between 1 and 0.4*N. These h individuals are randomly selected from the worst 40% of the new population, excluding the best individuals obtained after crossover. This is done to add new properties and to diversify the population. On each of these h

individuals, t_{mut} mutation iterations are applied, where each iteration consists of applying four mutation operators in a fixed sequence. If the solution obtained after mutation is better than the solution it started from, the latter is replaced by the former in the population; else, the solution after mutation replaces a randomly selected worst individual that was not selected for mutation. Next, we proceed to the next generation. Finally, the best solution is returned when the stopping criterion of the hybrid GA is met.

The main contributions of our method are in adapting the hybrid GA operators to the particular requirements of the H-DARP and in proposing a design for the overall organization of the hybrid algorithm to face the challenges posed by the specific aspects of the problem under study.

Algorithm 1: Pseudo-code of the proposed hybrid Genetic Algorithm

Begin
Initial population: Generate the initial population of *N* individuals/solutions, partly randomly and partly using a set of construction heuristics;
  Repeat
    Evaluation: Evaluate the *N* individuals according to the fitness function;
    Selection: Select by tournament selection (binary technique) *S* individuals from the current population;
      Repeat
        Select Parents: randomly select two parents from *S*;
        Sequencing Crossover: perform crossover operator by sequencing on the selected parents;
        Assigning Crossover: perform crossover operator by assigning on the selected parents;
        Delete Parents: Delete the parents from *S*;
      Until all parents are selected;
    Reparation: Repair infeasible solutions/children;
    Improvement: Improve the quality of all children using local search (intra- and inter-route);
    Replacement: Apply the Elitist approach;
    Mutation: Select randomly h individuals from the worst individuals;
      For each of the *h* individuals
        Repeat
          Perform intra-route mutation;
          Perform inter-route mutation;
        Until the number of iterations t_{mut} is reached;
        If an improvement is realized on the current solution **Then**
          The previous individual is replaced by the new one;
        Else
          The new individual replaces a randomly selected individual from the other (non-selected) worst individuals**;**
        End If
      End For
  Until the maximum number of generations n_{HGA} is reached;
Output the best individual as a result;
End.

3.1. Chromosome encoding

A chromosome encoding with a sequence of available vehicles v_k starting from the depot is considered. A route is represented by an ordered list of pickup and delivery nodes. Let d_i^+ and d_i^- the pickup and the delivery node of each user i, respectively. The depot is denoted 0. Each vehicle starts from the depot 0 and returns back to the same depot 0, e.g., for a solution with two routes and five requests the chromosome coding is as follows : $0,4^+,4^-,3^+,5^+,3^-,5^-,0$ for vehicle v_1 and $0,2^+,1^+,2^-,1^-,0$ for vehicle v_2.

3.2. Initial population

As recommended by Liu et al. (2009), Liu et al. (2013), Nguyen et al. (2014) and Koc et al. (2015), several different construction heuristics (between two and four) should be used to construct the initial population. Thus, we propose four different fast construction heuristics, each generating a single initial solution. They are slightly modified, in order to quickly generate a set of four initial solutions of good quality and high diversity. These heuristics are described in the following subsections:

- Two modified versions of the Sequential Construction Heuristic (SCH) of Solomon (1987) are used. The first is based on the Euclidian distance between two users denoted by "SCH 1", while the second, denoted by "SCH 2", is based on the earliest starting time (e_i) of the pickup node of each user i.
- Two modified versions of the Parallel Insertion Heuristic (PIH) reported by Fu (2002) are applied in order to find the best position of insertion for the pickup and delivery of each user in every route. According to Parragh et al. (2008), and Cordeau and Laporte (2003), this PIH is fast and efficient.

To complete the initial population of size N, N-4 solutions are generated randomly. For each solution, users are selected in a random order and inserted in a route that already exists. A new route is created to accommodate a user whenever any constraint of the existing routes is violated. The procedure stops when all users are inserted.

In the H-DARP, each user is defined by a pickup node and drop-off node. For each insertion method, inserting a user is equivalent to the insertion of its pickup node first, and then its drop-off node (not necessarily immediately after the pickup node), unless mentioned otherwise.

3.2.1. Sequential Construction Heuristic 1 (*SCH* 1)

This heuristic is a modified version of the one proposed by Solomon (1987). The closeness of two users is based on the Euclidian distance between their corresponding pickup nodes. A vehicle starts at the depot and visits the nearest user (the first in the list); then, the following in the list, etc. At the end of the route, the vehicle returns back to the depot. We consider as many vehicles as necessary, until all users are assigned, while respecting capacity, ride time, maximum route duration and time window constraints.

The Sequential Construction Heuristic 1 (SCH 1) works as follows: We initialize the set of non-assigned users sorted in increasing order of the Euclidean distance between their pickup location and the depot. We repeat the following until all users are assigned: We select the first non-assigned user in the list, and try to insert its pickup and drop-off nodes after the last inserted user in the first activated vehicle. If the insertion is not possible, we check with the next activated vehicle if this exists. If all activated vehicles are checked and the user is not inserted, we activate a new vehicle and insert the user in this new vehicle. We update the list of non-assigned users and the list of activated vehicles after the assignment of a user and the activation of a new vehicle, respectively.

3.2.2. Sequential Construction Heuristic 2 (*SCH* 2)

The structure of the SCH 2 is similar to SCH 1; only step 1 differs. All the requests are listed in increasing order of their pickup times (the earliest starting time (e_i) within the time window).

3.2.3. Parallel Insertion Heuristic 1 (*PIH 1*)

We have slightly modified the heuristic applied by Fu (2002) for the Dial-A-Ride problem with varying and stochastic travel times. The set of users are sorted in increasing order of the earliest starting time of their pickup node (e_i). The m routes ($0<m<K$ with K being the number of vehicles and m selected randomly) are initialized and designed in parallel. Each of the m first users is assigned to a different route. Next, we try to insert the rest of users, one by one, into the routes while respecting the vehicle capacity, ride time, time window and maximum duration constraints. The insertion is based on two steps. First, routes are sorted by the distance between the last assigned user to this route and the new user to insert. Second, route per route, it is tested whether the new user can be feasibly inserted in the route. If this is the case, the pickup and drop-off nodes of the user are inserted at their best positions in this route. If not, the next route is examined. If some users are still not assigned, a new route is opened and the same insertion technique is applied, until all users are assigned.

3.2.4. Parallel Insertion Heuristic 2 (*PIH* 2)

The structure of this heuristic is similar to PIH 1. The difference is that here the users are sorted by the Euclidean distance between their pickup locations and the depot. We consider all the K routes. The routes are initialized and designed in parallel. Each of the K first users is assigned to a different route. The rest of users are inserted into the routes while respecting the vehicle capacity, ride time, time window and maximum duration constraints (See Algorithm 2).

Algorithm 2: Parallel Insertion Heuristic 2 (PIH 2)

```
Begin
Initialize the set of non-assigned users sorted in increasing order of the Euclidean distance between their pickup
locations and the depot;
Assign the first K users to the K available vehicles (one user per vehicle);
    Repeat
        Select the first non-assigned user from the list;
        Sort vehicles in the increasing order of distance between the last assigned user in the vehicle and the selected
        user;
        Select the first vehicle in the list;
            Repeat
                Check the insertion feasibility of the user's pickup and drop-off nodes in the selected vehicle's route,
                starting from the beginning of the vehicle's route, until the end;
                If one or more feasible options are found Then
                    Insert the user in the best position;
                Else
                    Select the next vehicle in the list;
            Until the user is inserted
        Remove the user from the list of non-assigned users;
    Until all users are assigned;
Output the solution as a result;
End.
```

3.3. Evaluation: fitness function

In evolutionary algorithms, the fitness function measures the quality of each individual. The traditional function used in the literature is to calculate the lengths of all routes (Choi et al., 2003; Wink et al., 2012), sometimes increased with a dynamically adapted penalty to penalize infeasible routes like in Cao and Lai (2007) and Li (2009). In addition to the measurement of the individual's quality, the provided fitness function in this study allows for a better exploitation and a wider diversity of the search during the selection phase (Buro, 1997).

To measure the quality of a solution I, the fitness function (1) is considered:

$$Fit\ (I)=\frac{1}{1+exp^{(f_{RC}(I))}} \quad (1)$$

This fitness function contains an exponential argument in the denominator for reducing fitness differences between individuals. Hence, the genetic algorithm behaves like a random search algorithm for a better space exploration (Buro, 1997). Here, the objective function value $f_{RC}(I)$ is the total routing cost over all vehicles.

3.4. Selection: Tournament

The selection by tournament provided by Miller and Goldberg (1995) is considered in our algorithm. It has demonstrated effectiveness for several transportation problems (Freitas, 2013). The principle is to choose a subset of (s) individuals from the population (called the tournament size), and then select the individual in the group which has the highest fitness value. This process is repeated until the number of required individuals (S) is reached.

3.5. Crossover

In the crossover operator, genes of two parent solutions are recombined to form new descendants, called children. In the literature, the majority of developed GAs uses a single type of crossover. This, to some extent, reduces the search space of the algorithm. To better explore the search space and to enhance the quality of solutions, we propose applying two different crossover operators (crossover by sequencing followed by crossover by assigning) on each pair of parents selected from the S individuals. The integration of these two crossover operators is one of the originalities of our GA and provides a better balance between exploitation and exploration. The first crossover operator is intended for creating and exploring a new search space, while the second is applied to exploit the characteristics of the selected parents.

3.5.1. Crossover by sequencing

The crossover by sequencing operator (See Figure 1) creates two children from two parents (P1 and P2). The first child (E1) is generated by the simplified procedure OX proposed by Prins (2004). In this procedure we use a random one-point crossover p, with a slight modification, in order to respect the constraint that the corresponding pickup and delivery nodes must be visited in the same route (vehicle). The first child inherits all the genes located in the first parent P1 before the crossover point

p. The remaining elements of the first child are inherited from the first parent in the same order of their appearance in the second parent beginning from the first route.

The second child (E2) is generated symmetrically while exchanging the roles of the two parents.

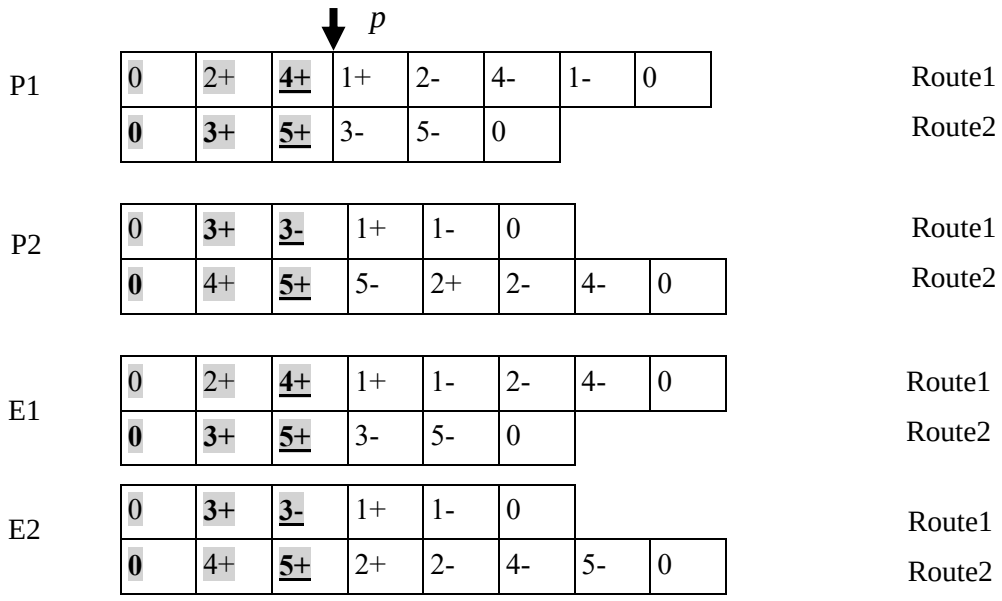
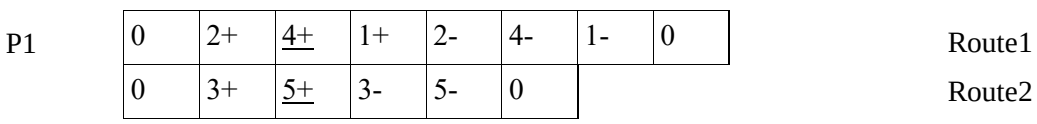


Figure 1: Crossover by sequencing

3.5.2. Crossover by assigning

The crossover by assigning operator is obtained by slightly modifying the Merge Crossover operator (MX1) suggested by Blanton and Wainwright (1993). The MX1 operator consists of generating one child from two parents by comparing genes one by one. The gene whose time window is earlier is inserted into the child, while the other gene is swapped in the corresponding parent as explained below. This crossover operator is modified so as to create two children (Hosny and Mumford, 2010); the first child is obtained by a comparison between the two genes according to their earliest time values (e_i), and the second according to their latest time values (l_i) (respecting the precedence constraints). The corresponding process is explained in Figure 2 for the first child. When comparing the first genes of both parents P1 and P2, suppose $e_{2+} < e_{3+}$. Hence, gene 2+ of P1 is inserted into the child, while in parent P2 gene 3+ swaps location with gene 2+ to maintain validity. For the second gene, suppose $e_{4+} < e_{5+}$, and so on. The third gene is the same for both parents (1+) and is inserted in the child. In the case of comparison with node 0 (depot), we keep this node (pickup or drop-off) and copy it in the child E1. If the related node (delivery or drop-off) is not present in child E1, then we add it just before or after the copied node. The resulting children may not be feasible with respect to time windows, capacity and ride time constraints, though. In that case, the infeasible solution is repaired as explained in Subsection 3.6.



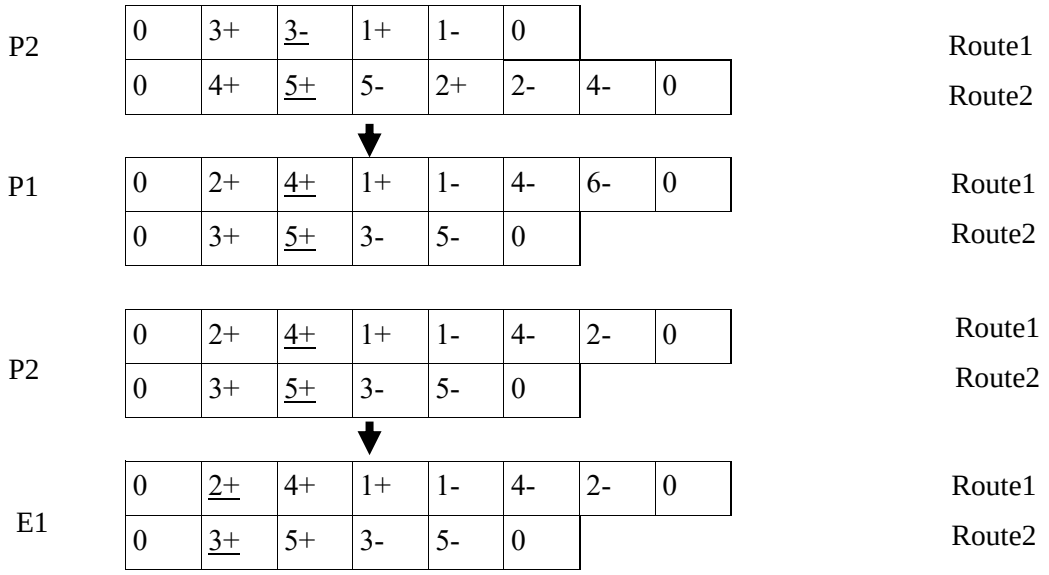


Figure 2: Crossover by assigning: child E1

3.6. Repairing infeasible solutions

To obtain a feasible solution, many constraints have to be respected in the H-DARP: capacity, time windows, ride time, route duration and precedence constraints. After the process of crossover, some solutions may be of low quality and sometimes infeasible. In this case, in order to repair the solution, an amendment phase is applied to all infeasible routes. First, m nodes are selected randomly from each infeasible route. The value of m is randomly selected in the interval $[z_l, z_u]$, where z_l and z_u are respectively the lower and upper bounds that are calculated as a percentage of the total number of nodes in this route. Then, the "best position" procedure is applied to try to insert these m nodes. This operator consists of finding for every node j the most appropriate insertion position, in the same route from which it was removed. This insertion respects the ride time, precedence and time window constraints. The best option (least increase in costs) is then selected and performed. This process continues until all nodes have been inserted.

If the insertion of a node is infeasible, we place it with its corresponding pickup/drop-off node in a list L. At the end, when all infeasible routes have been considered, the pairs of pickup and drop-off nodes in list L are inserted one at a time in any route using the "best position" described before, while respecting the feasibility of the solution. All possible insertion positions are identified, and the one increasing the routing cost the least is selected. If, after these attempts, the solution is still infeasible, we randomly select one of the initial solutions provided by our construction heuristics, and improve it by randomly selecting a local search among those provided in Subsection 3.7. The infeasible solution is then replaced by this newly obtained solution. This replacement procedure is a modified version of Merz and Katayama (2004).

3.7. Improvement: Local Search

To improve the quality of each child that is generated by the crossover operators, several well-known local search operators are applied to explore the search space by performing simple moves.

Two are intra-route operators: the 2-opt operator of Lin (1965) and a relocate operator of Savelsbergh (1992). Three are inter-route operators: the 2-opt* operator of Potvin and Rousseau (1995), a relocate operator, and the remove two insert one operator of Xiang et al. (2006).

The proposed local search starts by sorting routes in decreasing order of the distance. The idea is that long routes often contain remote users that cause high routing costs. Thus, these routes need to be primarily improved.

The inter- and intra-route local search operators are discussed in Subsections 3.7.1 and 3.7.2, respectively. The applied search strategy is presented in Subsection 3.7.3.

3.7.1. Inter-route local search

In the inter-route phase, we use three local search operators in sequence to explore all movement of users/edges between routes:

- First, the 2-opt* operator is applied on each combination of pairs of arcs in different routes.
- Second, a relocation operator that relocates a user's pickup and delivery nodes from one route to another is applied. The nodes are inserted in their best possible position, while respecting the feasibility of the solution.
- Third, the "Remove two insert one" operator provided by Xiang et al. (2006) is applied for each pair of routes from the solution. It consists of removing two randomly selected users from a route and inserting them one by one (in a random order) in another route, while maintaining feasibility.

3.7.2. Intra-route local search

Another improvement phase is considered in our approach and consists of improving each route separately. For this purpose, we use two traditional local search operators. First, we apply the 2-opt operator on each pair of arcs; next, a relocate operator is applied on each user in the route by removing the user and reinserting it in the best possible position.

3.7.3. Framework of the proposed local search strategy

To obtain high quality solutions, the proposed local search procedure is repeated for a fixed number of iterations. The structure of our local search is explained in Algorithm 3, and computational experiments are provided in Table 6 to show the efficiency of the applied local search operators.

Algorithm 3: The local search strategy

Begin
  Repeat
    Sort routes $\{R_0,...,R_k\}$ in descending order of distance;
      For each route $R_i \in \{R_1,...,R_k\}$
        For each route $R_j \in \{R_{i+1},...,R_k\}$
          Apply the *2-opt* operator* on routes R_i and R_j;
      For each route $R_i \in \{R_1,...,R_k\}$
        For each route $R_j \in \{R_{i+1},...,R_k\}$
          Apply the inter-route *relocate operator* (try to remove a user from R_i and insert it in route R_j);

For each route $R_i \in \{R_1, ..., R_k\}$

For each route $R_j \in \{R_{i+1}, ..., R_k\}$

Apply the *remove two insert one operator*;

For each route $R_i \in \{R_1, ..., R_k\}$

Apply the *2-opt operator* on route R_i;

For each route $R_i \in \{R_1, ..., R_k\}$

Apply the intra-route *relocate operator*;

Until maximum iterations is reached (100) or no improvement for ten consecutive iterations;

Output the solution as a result;

End.

3.8. Replacement: Elitism strategy

The "Elitism strategy" is considered to determine the individuals that will disappear from the population. To create the population of the new generation μ^*, the tournament technique described in Subsection 3.4 is applied. We select s individuals from the current population of children (∂) and the best one is put in μ^*. A similar procedure is applied on the current population (μ). This process is repeated until μ^* is filled. This strategy ensures the diversity of solutions throughout the search.

3.9. Mutation

The use of several mutation operators is recommended in the literature (e.g., Vidal et al., 2012, 2013; Tasan and Gen, 2012) in order to incorporate new characteristics in the population and enlarge the feasible search space (Wang and Chen, 2012). Thus, in our algorithm, four mutation operators are used; two inter-route mutation operators and two intra-route mutation operations. The four mutations are applied in a fixed sequence, from one to four, for each selected solution. We apply specific modifications to some traditional mutation operators as described in the following subsections.

3.9.1. Inter-route mutation

As the first inter-route mutation operator, the famous 2-opt exchange heuristic is adapted to deal with heterogeneous vehicles (See Figure 3). Two randomly selected users, assigned to different routes, are swapped, if possible. The application of this operator is repeated until we find a feasible option.

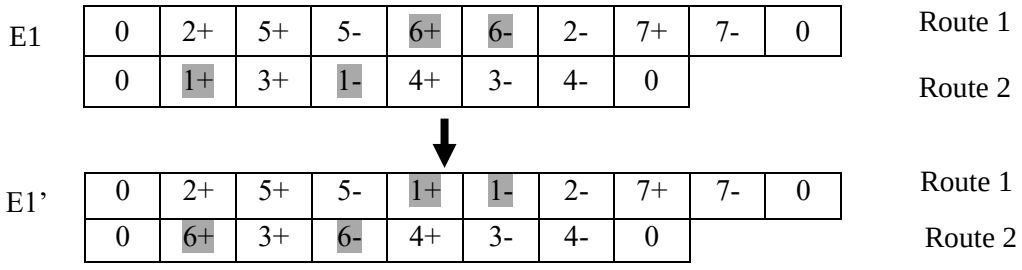


Figure 3: Inter-route mutation: Exchange between (1+, 1-) and (6+, 6-)

A second inter-route mutation operator consists of removing a randomly chosen user from the vehicle with the largest number of users, and inserting it in the vehicle that contains the smallest number of users. The insertion position is chosen randomly out of those that respect the time and

vehicle capacity constraints. If no feasible position exists, a new user is selected. This operator allows balancing the loads between the different vehicles.

3.9.2. Intra-route mutation

In the first intra-route mutation two nodes of a single route are selected randomly. Next, their respective positions are exchanged, under the condition that the resulting solution is feasible (See Figure 4).

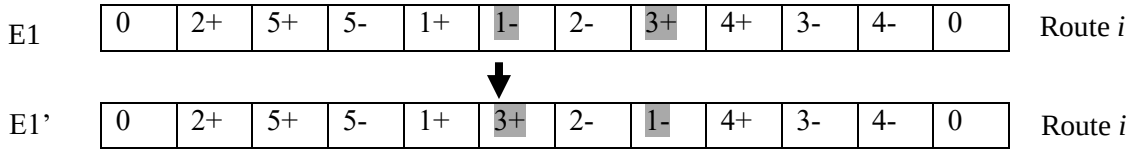


Figure 4: Intra-route mutation: exchanging the positions of two nodes 1- and 3+ within the same route

A second intra-route mutation operator (See Figure 5) is considered and consists of randomly choosing a delivery node, changing its position with a later delivery node (chosen randomly), while respecting the feasibility of the final solution. If no feasible option exists, we choose another node. This operation is repeated until we get a feasible solution or all delivery nodes have been considered.

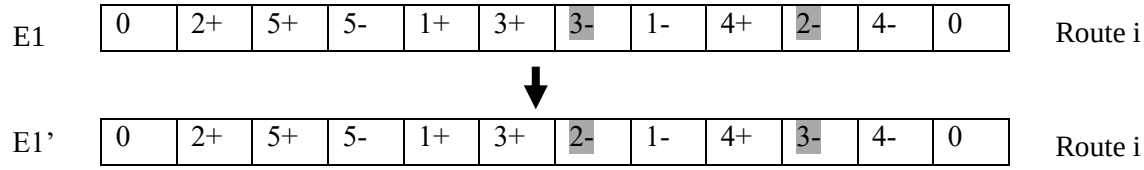


Figure 5: Intra-route mutation: node reinsertion (the position of node 2- is exchanged with the position of node 3-)

4. Computational experiments

The hybrid Genetic Algorithm was implemented in C. The experiments were conducted on a Fujitsu Siemens laptop with Intel Celeron 4 GHz and 1.86 GB of RAM. Six artificial benchmark data sets are considered. Results are compared to those of the Deterministic Annealing (DA) algorithm of Braekers et al. (2014) and the Evolutionary Local Search (ELS) algorithm of Chassaing et al. (2016).

In Subsection 4.1 the benchmark data sets are discussed. Parameter settings and design decisions for the algorithm are discussed and analyzed in Subsections 4.2 and 4.3, respectively. Finally, the results are presented in Subsection 4.4.

4.1. Benchmark instances

For small instances, we consider the benchmark data of Parragh (2011) for the H-DARP, containing three artificial data sets (U, E, I). These instances are based on modifying the instances provided by Cordeau (2006) for the standard DARP, in which the characteristics of heterogeneous vehicles and users are introduced. The instances contain 2–4 vehicles and 16–48 requests. Braekers et al. (2014) generated medium-sized instances for the H-DARP in a similar way. These contain 5-8

vehicles and 40-96 requests. For both the small and medium instances, the user time window length is 15 minutes, the maximum user ride time L_i= 30 minutes, and the service time s_i= 3 minutes. Two vehicle types are considered, with four types of resources: staff seats, patient seats, stretcher places and wheelchair places. To generate the instances (U, E, I), Parragh (2011) considered the probabilities shown in Table 1.

Table 1: Probabilities used to generate instances by Parragh (2011)

Instance	Patient request probabilities			Probability for companion (%)	Vehicle fleet
Set	% Seat	% stretcher	% wheelchair		
U	1.00	0.00	0.00	0.00	Homogeneous (T0)
E	0.50	0.25	0.25	0.10	Homogeneous (T2)
I	0.83	0.11	0.06	0.50	Heterogeneous (T1, T2)

In data set U, homogeneous users and vehicles of types T0 are considered, while heterogeneous users are used in data set E with homogeneous vehicles of type T2. Vehicles of type T0 only have 3 patient seats, while vehicles of type T1 have a capacity configuration of 2 staff seats, 1 patient seat, 1 stretcher, and 1 place for wheelchair. In data set I, heterogeneous users and vehicles (T1, T2) are included, in which vehicles of type T2 provide 1 staff seat, 6 patient seats, 0 stretchers, and 1 place for a wheelchair.

For larger instances, we considered the same instance generation idea of Parragh (2011) explained in Table 1 and applied it to the 20 benchmark instances provided by Cordeau and Laporte (2003) for the DARP.; i.e., the original homogeneous data set is denoted by set U, while two additional heterogeneous data sets (E and I) have been generated by modifying the instances as in Parragh (2011). The instances contain 3-13 vehicles and 24-144 requests, randomly generated in a [-10, 10]^2 sized Euclidean plane. Service time s_i=3 minutes for all users, the transportation time t_{ij} is equal to the Euclidean distance between i and j (denoted d_{ij}), the route duration limit T_{max}= 480 minutes, and the maximum ride time L_{max}= 90 minutes. The time window range is between 15 and 45 minutes for instances R1a to R10a, and between 30 and 90 minutes for the instances R1b to R10b. For the heterogeneous instances, upgrading conditions are applied as discussed in Parragh (2011).

4.2. Parameter setting

The proposed hybrid GA has several parameters that need to be set. Inspired by the experimental testing of Koç et al. (2015) the number of local search iterations is set to 100, t_{mut}= 10 and $[z_b, z_u]$= [0.2, 0.9]. A sensitivity analysis is performed to find good parameter settings for the other parameters: population size (N), number of solutions chosen for crossover (S), number of iterations (n_{HGA}), and tournament size (s). Prins (2004) provided some guidance for setting these parameter values when solving the vehicle routing problem. We follow this guidance with respect to the small population size, and the large number of solutions chosen for crossover.

The combined impact of the former three parameters is tested on 12 problem instances, which are selected such that the number of requests varies from small to large with different levels of heterogeneity. For each instance from this data set, we ran the algorithm five times for each parameter

setting. For each parameter, three values are considered: number of iterations n_{HGA}= {20,000; 50,000; 100,000}, population size N= {10, 20, 30} and the number of individuals selected for crossover S= {0.7*N, 0.8*N, 0.9*N}. Concerning the number of solutions chosen for crossover (S), we considered a high crossover rate (between 0.6 and 1.0), as recommended by Holland (1975) and Prins (2004). In Table 2, the computational results of the sensitivity analysis are presented. The row "Best" ("Avg") refers to the average over all instances of the best (average) solution value obtained by our hybrid GA algorithm for the corresponding parameter combination, while the row "CPU(s)" gives the average run time in seconds. We note that the detailed results of each instance are available on the website: http://hdarp-results.e-monsite.com.

Table 2: Identification of the best parameter setting for the hybrid GA

n_{HGA}	20,000								
(N,S)	(10,0.7*N)	(10,0.8*N)	(10,0.9*N)	(20,0.7*N)	(20,0.8*N)	(20,0.9*N)	(30,0.7*N)	(30,0.8*N)	(30,0.9*N)
Best	731.92	732.78	733.26	732.62	732.85	731.21	732.97	733.27	732.43
Avg	729.08	730.47	730.48	729.14	730.57	728.99	730.49	728.80	730.22
CPU (s)	92.87	98.62	101.79	101.69	98.56	98.70	101.23	98.80	101.80
n_{HGA}	50,000								
(N,S)	(10,0.7*N)	(10,0.8*N)	**(10,0.9*N)**	(20,0.7*N)	(20,0.8*N)	(20,0.9*N)	(30,0.7*N)	(30,0.8*N)	(30,0.9*N)
Best	730.87	730.11	**729.02**	730.25	729.02	729.02	729.88	729.26	730.09
Avg	730.62	729.20	**727.95**	729.66	729.45	728.49	729.17	728.37	728.45
CPU (s)	106.86	106.85	**106.70**	107.07	106.95	107.94	107.92	108.23	109.44
n_{HGA}	100,000								
(N,S)	(10,0.7*N)	(10,0.8*N)	(10,0.9*N)	(20,0.7*N)	(20,0.8*N)	(20,0.9*N)	(30,0.7*N)	(30,0.8*N)	(30,0.9*N)
Best	729.02	728.90	729.02	729.02	729.02	729.02	729.02	728.77	728.72
Avg	727.89	727.89	727.98	727.80	727.84	727.62	727.40	727.55	726.86
CPU (s)	178.50	195.66	208.24	243.89	252.24	367.88	329.17	400.44	444.49

The number of iterations n_{HGA} and the values of N and S remarkably affect the solution quality. Table 2 shows that the average solution quality is not greatly improved after 50,000 iterations, while the computation time obviously increases with increasing the number of iterations. Therefore, the parameters setting (indicated in bold) n_{HGA}=50,000 iterations, N= 10, and S=0.9*N appears to offer the best trade-off between average solution quality and CPU time. Hence, these values were used in all further experiments.

Finally, experiments were conducted to set the tournament size (s). As Liu et al. (2015) indicated, the larger the tournament size is, the less the opportunity to choose weak individuals. In this study, tournament sizes of two, five, seven and nine are tested on some different instances types. In Table 3, 12 instances of each data set (U, E, I) have been analyzed. These instances were chosen such that the number of requests varies from small to large with various degrees of heterogeneity. Each instance was solved five times for each given parameter value s = {2, 5, 7 and 9}. The columns "Best" ("Avg") report the best (average) solution values and CPU(s) refers to the computational time in seconds.

Table 3: Impact of tournament size on the quality of solution

Instances	s=2			s=5			s=7			s=9		
	Avg	Best	CPU (s)	Avg	Best	CPU (s)	Avg	Best	CPU (s)	Avg	Best	CPU (s)
a4-32 (U)	485.50	**485.50**	37.84	485.50	**485.50**	37.79	485.50	**485.50**	36.68	485.50	**485.50**	38.49
a8-64 (U)	747.46	**747.46**	55.73	747.46	**747.46**	55.52	747.46	**747.46**	54.15	747.46	**747.46**	56.64
a3-30 (E)	503.34	501.87	25.14	500.58	**500.58**	25.16	500.58	**500.58**	24.45	500.58	**500.58**	25.57

a8-96 (E)	1268.71	1266.36	93.14	1270.31	1266.38	92.50	1266.03	**1265.36**	90.61	1267.56	1266.23	94.34
a6-60 (I)	835.09	832.18	59.67	834.42	832.61	59.38	830.49	**830.29**	57.80	830.84	830.56	60.60
a6-72 (I)	940.17	937.45	84.68	938.38	937.69	84.17	936.51	**936.32**	82.13	936.51	936.48	85.60
R9a (U)	662.02	660.24	95.16	660.24	**659.00**	91.34	660.24	660.24	92.20	661.89	661.63	96.79
R5b (U)	582.62	581.42	204.80	583.95	582.46	204.66	582.06	**579.03**	198.19	583.56	581.98	208.41
R2b (E)	314.53	**314.12**	63.96	314.32	**314.12**	63.76	314.12	**314.12**	62.06	314.12	**314.12**	64.91
R3b (E)	554.33	553.25	84.04	554.13	554.08	83.83	553.15	**551.95**	81.32	554.33	552.03	85.09
R6a (I)	845.20	844.62	193.73	844.06	**841. 82**	181.96	844.26	843.27	188.32	845.62	842.78	196.20
R8b (I)	520.94	518.70	116.43	521.16	518.08	115.53	520.01	**517.26**	112.43	521.56	520.34	117.81

Table 3 shows the influence of tournament size on the performance of the hybrid GA. We conclude that the tournament size value s = 7 produces better solutions for most instances, in terms of both best and average results as well as computational time, compared to other tournament sizes.

4.3. Analysis of design decisions

In this section, the effect on performance of the main components of our algorithm is assessed.

4.3.1 Impact of the GA design components on the hybrid GA

In our hybrid GA, many components (sequencing/assigning crossover, intra/inter route mutation and local search operators) are applied. We consider the execution of the hybrid GA with and without the incorporation of the sequencing/assigning crossover and intra/inter route mutation. Thus, several combinations are selected and tested. We also evaluate the impact of the local search operators on the performance of each combination. These combinations can be classified into three categories, as shown in Table 4. The first category (the combinations from 1 to 8 in Table 4) consists of choosing only one crossover operator (sequencing or assigning) and only one mutation (inter or intra- route). The second category (the combinations from 9 to 12 in Table 4) combines only one crossover operator (sequencing or assigning) with the two mutations (intra and inter- route). The third category (the combinations from 13 to 15 in Table 4) combines two crossover operators with only one mutation (intra or inter-route). We note that the final combination in Table 4 represents our hybrid GA following the general structure of Algorithm 1.

In order to determine the best hybridization scheme and the impact of each component on the solution quality, we provide in Figure 6 a comparison between the configurations in terms of the best and the average value of five runs (Avg) on small-medium sized instances (a) and large sized instances (b), for all data sets U, E and I. For more details, the readers can find the detailed results of each data set (U, E, I) in the website.

Table 4: Combination of different component of the GA

Combination	Description
1	GA (sequencing crossover and inter-route mutation operators)
2	GA (sequencing crossover and inter-route mutation operators)+ local search procedure
3	GA (sequencing crossover and intra-route mutation operators)
4	GA (sequencing crossover and intra-route mutation operators)+ local search procedure
5	GA (assigning crossover and intra-route mutation operators)
6	GA (assigning crossover and intra-route mutation operators)+ local search procedure
7	GA (assigning crossover and inter- route mutation operators)
8	GA (assigning crossover and inter- route mutation operators)+ local search procedure

9	GA (assigning crossover and inter and intra- route mutation operators)
10	GA (assigning crossover and inter and intra- route mutation operators)+ local search procedure
11	GA (sequencing crossover and inter and intra- route mutation operators)
12	GA (sequencing crossover and inter and intra- route mutation operators)+ local search procedure
13	GA (two crossovers and inter-route mutation)
14	GA (two crossovers and inter-route mutation)+ local search procedure
15	GA (two crossovers and intra-route mutation)+ local search procedure
16	GA (two crossovers and intra and inter-route mutation)+ local search procedure

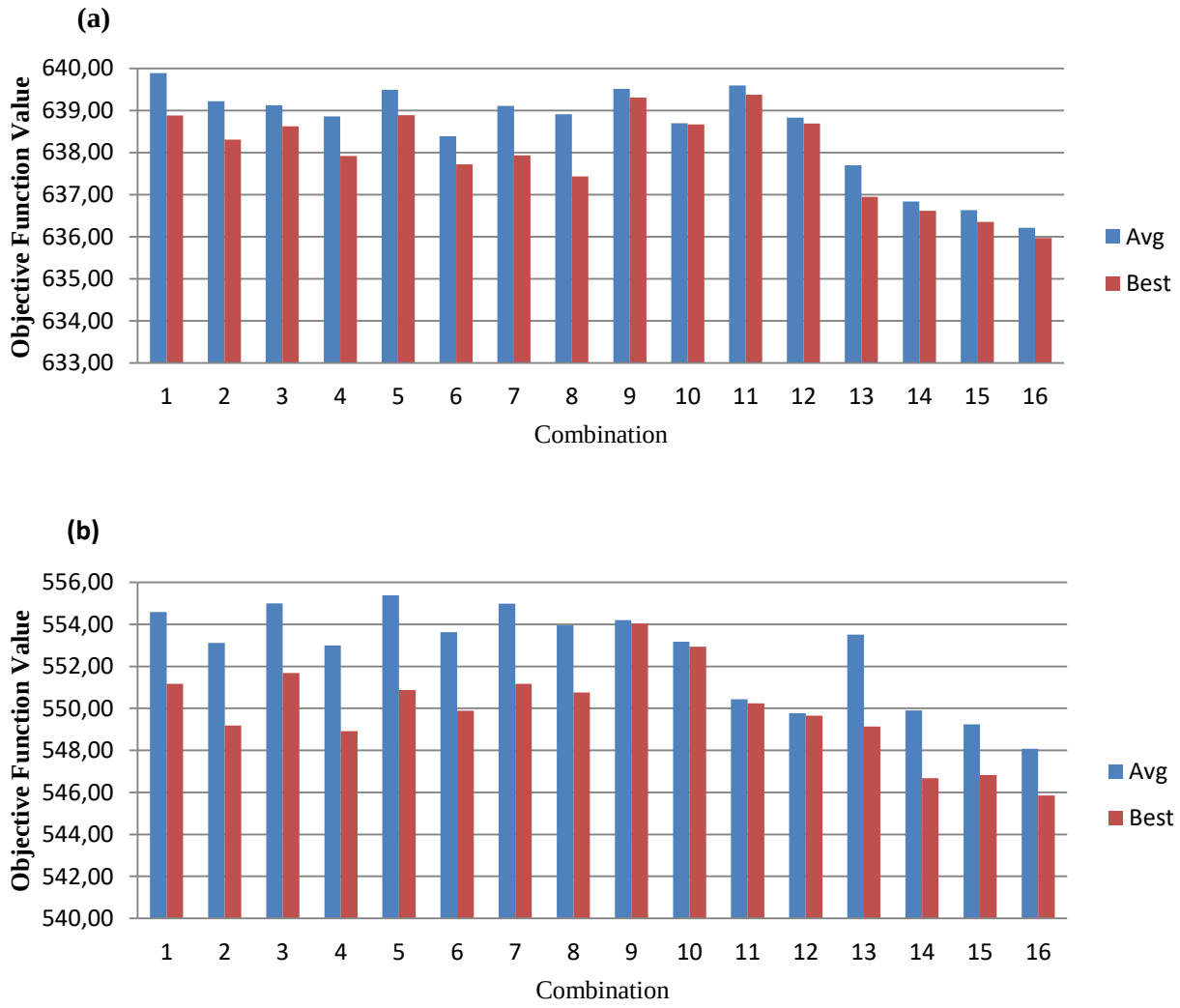


Figure 6: The impact of different components of the hybrid GA on different sized instances

By using only one crossover operator and both intra and inter-route mutations, we observe a slight advantage compared to using only one crossover operator, and either inter or intra-route mutation. However, a big improvement is observed when applying two crossover operators. In addition, the results in Figure 6 show that the solution quality is similar for the two configurations that use two crossovers with inter or intra-route mutations (two crossovers and intra-route mutation, two crossovers and inter-route mutation), which indicates the effectiveness of using the two crossovers.

Figure 6 shows that the incorporation of the different crossover and mutation operators significantly affects the performance of the algorithm. It is also observed that the hybridization with local search on each combination improves the performance of the GA. In fact, the best solution obtained by the configuration "assigning crossover and intra-route mutation" for the small-medium

instances was 638.89, compared to 637.72 achieved when the local search operators were added to this configuration. In addition, applying both assignment and sequencing crossovers and inter and intra-route mutations (last configuration in Figure 6) is the most effective configuration for all instances.

4.3.2. Impact of local search operators on the hybrid GA

The local search operators were explained in Subsection 3.7. They are considered important to achieve good quality solutions. As shown in Figure 7, we compared the objective function values over five runs of different local search combinations (shown in Table 5) on small-medium sized instances (c) and large sized instances (d) of all data sets U, E and I.

Table 5: Combination of local search operators

Combination	Local search(s)
1	2-opt
2	Relocate (Intra-route)
3	2-opt*
4	Remove two insert one
5	2-opt + Relocate (Intra-route)
6	Relocate (Intra-route) + Relocate (Inter-route)
7	2-opt + Relocate (Intra-route) + 2-opt*
8	2-opt + Relocate (Inter-route) + Remove two insert one
9	2-opt+ Relocate (Intra-route) + 2-opt*+ Relocate (Inter-route)
10	2-opt+ Relocate (Intra-route) + 2-opt*+ Relocate (Inter-route) +Remove two insert one

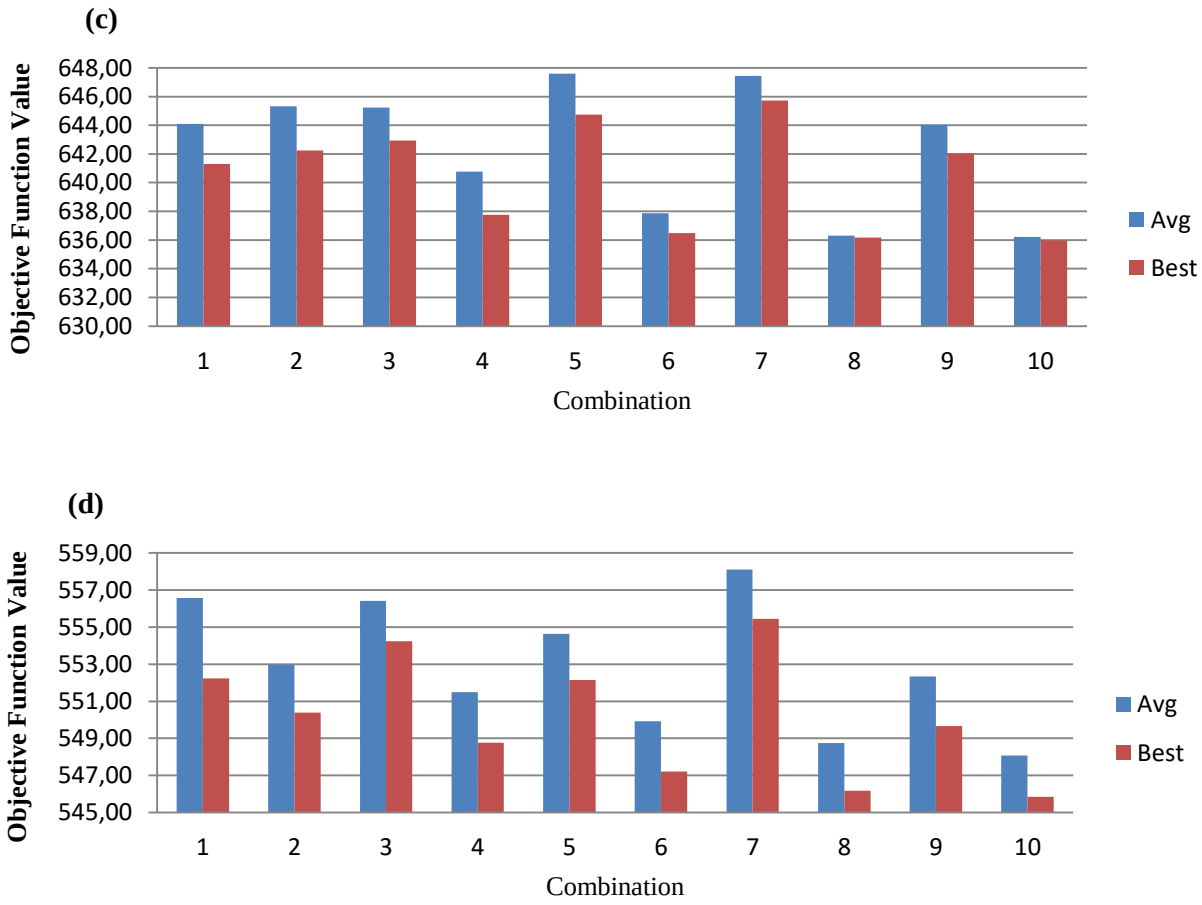


Figure 7: Effect of local search operators

By using only one local search in the hybrid GA, we observe that there is no improvement obtained in the solution quality. However, after the application of four or five operators in the same move, the solution quality is highly affected. The combination of the five local search operators (combination 10) provides the best results.

4.3.3 Effect of the mutation phase on the hybrid GA

In traditional GA, the mutation is usually conducted after the crossover phase and before the replacement phase (Elitist approach) as mentioned in Liu et al. (2009), Liu et al. (2013) and Nguyen et al. (2014). But in our case the mutation is considered after the replacement phase with the intention of increasing diversity and enhancing the performance of the GA.

To test the usefulness of our mutation procedure, we compared the best (average) results obtained by our hybrid GA using mutation after replacement and mutation before replacement, as shown in Table 6. The analysis was performed on 12 instances from each data set (U, E, I). Again, these instances were chosen such that the number of requests varies from small to large with various degrees of heterogeneity. We note that the columns "Best%" ("Avg%") present the percentage of deviation from the best (Avg) solutions found by our hybrid GA with mutation procedure after replacement procedure.

Table 6: Importance of mutation procedure in our hybrid GA

Instances	hybrid GA with mutation procedure after replacement phase			hybrid GA with mutation procedure before replacement phase				
	Avg	Best	CPU (s)	Avg	Avg %	Best	Best %	CPU (s)
a4-40 (U)	557.69	557.69	37.82	560.88	0.57	557.69	0.00	40.61
a8-96 (U)	1231.04	1229.66	95.13	1237.03	0.49	1231.68	0.16	101.81
a5-60 (E)	828.90	828.90	58.10	833.48	0.55	830.03	0.14	65.58
a7-84 (E)	1093.90	1092.90	82.27	1102.13	0.75	1096.07	0.29	87.64
a4-32 (I)	487.14	486.93	35.49	489.96	0.58	487.80	0.18	40.15
a6-48 (I)	604.12	604.12	49.01	609.56	0.90	606.00	0.31	52.26
Avg	***800.47***	***800.03***	***59.64***	***805.51***	***0.63***	***801.55***	***0.19***	***64.68***
R9a (U)	660.24	658.31	92.20	664.33	0.62	660.11	0.27	99.20
R5b (U)	582.06	579.03	198.19	583.74	0.29	581.62	0.45	224.38
R9a (E)	748.87	746.23	97.01	752.10	0.43	749.10	0.38	109.87
R9b (E)	703.15	699.06	170.20	705.31	0.31	702.03	0.43	191.35
R5a (I)	679.11	677.50	142.48	680.87	0.26	679.55	0.30	161.95
R4b (I)	559.12	557.99	222.52	561.45	0.42	560.66	0.48	251.18
Avg	***655.43***	***653.02***	***153.77***	***657.97***	***0.39***	***655.51***	***0.39***	***172.99***

According to Table 6, in all instances (except a4-40 (U)), we notice the existence of positive percent values found by the hybrid GA with mutation before replacement. Thus, it is deduced that the use of mutation before replacement procedure is not effective in avoiding convergence during the evolutionary process. In fact, the performance of the hybrid GA using mutation procedure before replacement phase, in terms of both best solution found and CPU time, is worse than our proposed hybrid GA, as indicated in Table 6.

4.4. Results on the HDARP instances

Our hybrid GA is compared to the ELS algorithm of Chassaing et al. (2016) and the DA algorithm of Braekers et al. (2014). Each instance is computed five times using each method. For each Table in this section, columns "Best" ("Avg") report the best (average) solution values. The columns "Best %" ("Avg %") present the percentage of deviation from the best solutions (BS). CPU(s) refers to the computation time in seconds. We note that with respect to computation times, we cannot fairly compare the performance of our algorithms against those reported in Braekers et al. (2014) and Chassaing et al. (2016). This is due to using a different machine for each of these algorithms. Moreover, the processing times reported in Table 8 cannot be accurately compared, since no relevant information has been reported in Dongarra (2014) and in Linpack (www.roylongbottom.org.uk) regarding the computational power of MFlops and the speed factor of the configuration used for the DA algorithm of Braekers et al. (2014). Similarly, no such information is available for our configuration. Nevertheless, for the ELS algorithm of Chassaing et al. (2016), the number of MFlops was reported as equal to 2,529.

Table 7 shows the results of our hybrid GA and the DA of Braekers et al. (2014) on the small instances of Parragh (2011) and the medium instances of Braekers et al. (2014). We note that, the detailed results of this Table can be found in our website.

Table 7: Comparison of DA and hybrid GA on small and medium instances

Instances	BS[a]	DA (Braekers et al., 2014)[b]					Our hybrid GA				
		Avg	Avg %	Best	Best %	CPU (s)	Avg	Avg %	Best	Best %	CPU (s)
Avg U	627.73[a]	627.84	0.01	627.73	0.00	31.00	627.80	0.01	627.73	0.00	44.35
Avg E	645.97[a]	646.09	0.01	645.98	0.00	30.10	646.15	0.02	645.97	0.00	43.34
Avg I	633.36[a]	636.20	0.29	635.92	0.25	27.50	634.68	0.14	634.21	0.09	46.46
Avg UEI	***635.72***	***636.71***	***0.10***	***636.55***	***0.08***	***29.60***	***636.23***	***0.05***	***635.99***	***0.03***	***44.69***

[a] *Best solutions provided by Braekers et al. (2014) with Branch and Cut algorithm*
[b] *Results of Braekers et al. (2014), programmed in C++ and executed on 2.6 GHz Intel Core laptop with 4 GB RAM.*

Table 7 shows that our hybrid GA is more effective than the Deterministic Annealing algorithm (DA) of Braekers et al. (2014), albeit using slightly higher computation times. The average deviation from the best solution over five runs is 0.05% for our hybrid GA and 0.10% for the DA. The average deviation for the best run is 0.03% for the hybrid GA and 0.08% for DA. As shown in Table 7 in the website, our algorithm performs especially well for instances with heterogeneous users and vehicles (data set I), e.g., for instances a6-72, a7-70, a7-84, a8-80 and a8-96, the best solution of our hybrid GA is better than the one provided by the DA of Braekers et al. (2014).

Table 8 shows the results of our hybrid GA on the large instances of Cordeau and Laporte (2003) for the DARP with a comparison to the state-of-the-art methods in literature: the DA algorithm of Braekers et al. (2014) and the ELS algorithm of Chassaing et al. (2016).

Table 8: Comparison of hybrid GA, DA and ELS algorithms on data set U for the DARP

Inst,	BKS[a,b]	DA (Braekers et al., 2014)[c]					ELS (Chassaing et al., 2016)[d]					Our hybrid GA				
		Avg	Avg%	Best	Best%	CPU (s)	Avg	Avg%	Best	Best%	CPU (s)	Avg	Avg%	Best	Best%	CPU (s)
R1a	190.02[a]	**190.02**	0.00	**190.02**	0.00	16.60	**190.02**	0.00	**190.02**	0.00	15.00	**190.02**	0.00	**190.02**	0.00	21.20
R2a	301.34[a]	**301.34**	0.00	**301.34**	0.00	42.00	**301.34**	0.00	**301.34**	0.00	75.00	**301.34**	0.00	**301.34**	0.00	53.63

R3a	532.00[a]	**533.54**	0.29	532.10	0.02	48.80	533.86	0.35	532.42	0.08	138.00	534.08	0.39	**532.00**	0.00	62.32
R4a	570.25[a]	580.52	1.80	577.16	1.21	74.60	574.47	0.74	570.55	0.05	442.20	**571.45**	0.21	**570.25**	0.00	95.26
R5a	626.93[b]	632.06	0.82	629.80	0.46	89.20	637.59	1.70	630.81	0.62	724.20	**631.39**	0.71	**628.48**	0.25	113.91
R6a	785.26[a]	800.68	1.96	797.78	1.59	107.00	796.10	1.38	792.81	0.96	1315.20	**788.52**	0.42	**787.41**	0.27	136.64
R7a	291.71[a]	292.23	0.18	292.23	0.18	22.60	292.96	0.43	**291.71**	0.00	28.20	**291.79**	0.03	**291.71**	0.00	28.86
R8a	487.84[a]	**491.00**	0.65	490.94	0.64	48.60	493.16	1.09	491.58	0.77	160.80	491.53	0.76	**488.89**	0.22	62.06
R9a	658.31[a]	666.65	1.27	662.64	0.66	72.20	681.35	3.50	672.88	2.21	675.00	**660.24**	0.29	**658.31**	0.00	92.20
R10a	851.82[b]	860.83	1.06	853.98	0.25	114.40	860.68	1.04	857.34	0.65	1279.80	**859.91**	0.95	**853.16**	0.16	146.09
R1b	164.46[a]	**164.46**	0.00	**164.46**	0.00	23.80	**164.46**	0.00	**164.46**	0.00	16.80	**164.46**	0.00	**164.46**	0.00	30.39
R2b	295.66[a]	296.06	0.14	295.69	0.01	51.40	295.72	0.02	**295.66**	0.00	82.20	**295.66**	0.00	**295.66**	0.00	65.64
R3b	484.83[a]	490.03	1.07	488.61	0.78	76.20	490.70	1.21	489.02	0.86	222.00	**487.23**	0.50	**484.83**	0.00	97.31
R4b	529.33[a]	540.99	2.20	534.99	1.07	117.00	**531.98**	0.50	**531.06**	0.33	612.00	532.19	0.54	531.86	0.48	149.41
R5b	577.29[b]	584.33	1.22	581.46	0.72	155.20	**580.23**	0.51	**578.45**	0.20	1195.80	582.06	0.83	579.03	0.30	198.19
R6b	730.69[b]	747.19	2.26	743.56	1.76	180.60	**736.61**	0.81	**731.27**	0.08	1939.20	741.06	1.42	737.03	0.87	230.62
R7b	248.21[a]	249.33	0.45	249.33	0.45	34.00	**248.21**	0.00	**248.21**	0.00	34.80	248.29	0.03	**248.21**	0.00	43.42
R8b	458.73[b]	**462.38**	0.80	461.77	0.66	81.00	**462.38**	0.80	461.22	0.54	259.20	463.32	1.00	**461.11**	0.52	103.44
R9b	593.49[a]	600.63	1.20	598.23	0.80	146.40	597.53	0.68	595.39	0.32	745.80	**595.37**	0.32	**593.49**	0.00	186.95
R10b	785.68[b]	801.89	2.06	795.08	1.20	162.80	803.99	2.33	796.57	1.39	1887.00	**793.64**	1.01	**791.01**	0.68	207.89
Avg	***508.19***	***514.31***	***0.97***	***512.06***	***0.62***	***83.22***	***513.67***	***0.85***	***511.14***	***0.45***	***592.41***	***511.18***	***0.47***	***509.41***	***0.19***	***106.27***

[a] *Best known solutions provided by Parragh and Schmid (2013)*
[b] *Best known solutions provided by Braekers et al.(2014)*
[c] *Results of Braekers et al.(2014), programmed in C++ and executed on 2.6 GHz Intel Core laptop with 4 GB RAM.*
[d] *Results of Chassaing et al. (2016), programmed in C++ and executed on Intel Core i7-3770 CPU with 3.40 GHz (average objective values (Avg) are calculated based on the provided average gaps (Avg%) on their website)*

Table 8 shows that our hybrid GA is more effective than the DA and the ELS algorithms in terms of best and average solution values. In fact, the average deviation of the average results from the best knows solutions are 0.47% for our algorithm, 0.97% for the DA algorithm and 0.85% for the ELS algorithm. The average deviation for the best result over five runs is 0.19% for our hybrid GA, 0.62% for the DA algorithm and 0.45% for the ELS algorithm. In addition, our hybrid GA can find 17 best solutions (Column "Best") compared to 3 found by the DA algorithm and 9 found by the ELS algorithm. Moreover, our hybrid GA can find 13 best solutions, on average over five runs (Column "Avg"), compared to 6 found by the DA algorithm and 8 by the ELS algorithm.

To evaluate the performance of our method on large instances with heterogeneous users and vehicles, we make a comparison to the DA of Braekers et al. (2014) on our newly generated instances. Tables 9 and 10 show the results obtained for data sets E and I, respectively. These data sets were generated by adapting the instances of Cordeau and Laporte (2003) as discussed in Subsection 4.1.

Table 9: Comparison of DA and hybrid GA on large H-DARP instances (data set E)

Instance	BS[a]	DA (Braekers et al., 2014)				hybrid GA				
		Avg	Avg %	Best	CPU	Avg	Avg %	Best	Best %	CPU
R1a	195.97	195.97	0.00	195.97	24.80	195.97	0.00	195.97	0.00	26.64
R2a	336.34	336.34	0.00	336.34	48.20	336.34	0.00	336.34	0.00	51.06
R3a	587.43	588.40	0.17	587.43	52.00	589.86	0.41	586.18	-0.21	56.61
R4a	642.44	644.02	0.25	642.44	82.80	642.56	0.02	640.03	-0.38	88.80
R5a	717.69	718.51	0.11	717.69	102.00	718.51	0.11	714.83	-0.40	109.00
R6a	885.24	888.08	0.32	885.24	133.20	887.65	0.27	883.02	-0.25	143.88
R7a	312.49	312.87	0.12	312.49	27.00	312.96	0.15	312.05	-0.14	28.34
R8a	556.01	557.45	0.26	556.01	46.80	556.23	0.04	553.82	-0.39	48.84
R9a	748.53	754.44	0.79	748.53	88.80	748.87	0.05	746.23	-0.31	97.01
R10a	966.20	971.54	0.55	966.20	108.20	969.22	0.31	963.08	-0.32	137.34
R1b	190.39	190.39	0.00	190.39	33.00	190.39	0.00	190.39	0.00	34.24

R2b	312.92	312.92	0.00	312.92	59.00	314.12	0.38	312.92	0.00	62.06
R3b	554.57	556.04	0.27	554.57	78.40	553.15	-0.26	551.95	-0.47	81.32
R4b	607.97	613.15	0.85	607.97	119.00	610.48	0.41	606.08	-0.31	119.84
R5b	643.68	648.98	0.82	643.68	161.40	642.15	-0.24	641.84	-0.29	175.84
R6b	838.12	840.91	0.33	838.12	200.80	836.32	-0.21	832.53	-0.67	219.52
R7b	277.32	277.32	0.00	277.32	41.80	276.52	-0.29	276.52	-0.29	47.15
R8b	529.81	533.97	0.79	529.81	84.00	532.28	0.47	530.56	0.14	92.00
R9b	700.95	704.43	0.50	700.95	131.40	703.15	0.31	699.06	-0.27	170.20
R10b	907.62	916.85	1.02	907.62	148.20	906.91	-0.08	902.17	-0.60	186.30
Avg	***575.58***	***578.13***	***0.36***	***575.58***	***88.54***	***576.18***	***0.09***	***573.78***	***-0.26***	***98.80***

[a] *New best solutions provided by Braekers et al.(2014) with Deterministic Annealing algorithm programmed in C++ and executed on 2.6 GHz Intel Core laptop with 4 GB RAM.*

Table 10: Comparison of DA and hybrid GA on large H-DARP instances (data set I)

Instance	BS[a]	DA (Braekers et al., 2014)				hybrid GA				
		Avg	Avg %	Best	CPU	Avg	Avg %	Best	Best %	CPU
R1a	193.27	193.27	0.00	193.27	24.80	193.27	0.00	193.27	0.00	29.61
R2a	319.87	320.42	0.17	319.87	53.20	319.87	0.00	319.43	-0.14	63.78
R3a	587.11	587.83	0.12	587.11	57.00	586.11	-0.17	584.84	-0.39	64.92
R4a	594.11	598.79	0.79	594.11	100.40	593.56	-0.09	591.24	-0.48	117.70
R5a	679.51	689.26	1.43	679.51	118.80	679.11	-0.06	677.50	-0.30	142.48
R6a	844.29	849.10	0.57	844.29	152.20	843.27	-0.12	838.26	-0.71	188.32
R7a	328.27	328.95	0.21	328.27	27.00	329.12	0.26	328.10	-0.05	31.68
R8a	554.71	556.41	0.31	554.71	54.40	556.46	0.32	552.35	-0.43	61.96
R9a	715.42	721.33	0.83	715.42	120.60	718.55	0.44	713.55	-0.26	136.67
R10a	936.79	942.10	0.57	936.79	200.40	937.23	0.05	932.83	-0.42	227.79
R1b	177.57	177.57	0.00	177.57	32.40	177.57	0.00	177.57	0.00	35.31
R2b	304.86	305.70	0.28	304.86	63.60	304.02	-0.28	304.02	-0.28	75.17
R3b	552.92	554.88	0.35	552.92	90.60	555.19	0.41	551.13	-0.32	105.46
R4b	560.23	564.99	0.85	560.23	162.80	559.12	-0.20	557.99	-0.40	222.52
R5b	631.07	634.54	0.55	631.07	201.80	630.59	-0.08	628.62	-0.39	238.74
R6b	799.32	802.68	0.42	799.32	224.20	797.57	-0.22	794.03	-0.66	272.35
R7b	297.41	297.41	0.00	297.41	34.20	297.51	0.03	297.41	0.00	37.59
R8b	518.67	520.22	0.30	518.67	89.60	520.01	0.26	517.26	-0.27	112.43
R9b	663.93	668.17	0.64	663.93	169.60	666.44	0.38	662.75	-0.18	208.43
R10b	870.10	884.30	1.63	870.10	244.80	873.18	0.35	865.07	-0.58	279.10
Avg	***556.47***	***559.90***	***0.50***	***556.47***	***111.12***	***556.89***	***0.06***	***554.36***	***-0.31***	***132.60***

[a] *New best solutions provided by Braekers et al.(2014) with Deterministic Annealing algorithm programmed in C++ and executed on 2.6 GHz Intel Core laptop with 4 GB RAM.*

Tables 9 and 10 clearly show that our hybrid GA obtains better results compared to the DA method in terms of solution quality. For data set E, with heterogeneous users and homogeneous vehicles, our algorithm improves the results of Braekers et al. (2014) by 0.26% on average, for both the average and best result over five runs. For data set I, with heterogeneous users and vehicles, our hybrid GA is more efficient than the DA algorithm, with an average improvement of 0.44% for the average and 0.31% for the best result. Combining both data sets, the average improvement is 0.35% and 0.29% for the average and best results, respectively. Besides, our method provides best known results for 31 of these instances and ties with the DA algorithm on 8 other instances. Nevertheless, this comes at the expense of a slight increase in computation time of the hybrid GA compared to the DA, as indicated in Tables 9 and 10.

5. Conclusion

Dial-A-Ride Problems (DARPs) are vehicle routing problems that arise in the management of door-to-door transportation services; for example, for the elderly and the disabled. The Heterogeneous Dial-A-Ride Problem (H-DARP) is a more realistic variant of the standard DARP, in which both heterogeneous vehicles and users with different requirements are considered. In practice, some users may for example need to be transported in a wheelchair or on a stretcher.

In this paper, a new hybrid Genetic Algorithm was proposed to solve the DARP and H-DARP. This algorithm is guided by efficient construction heuristics and efficient crossovers, mutations and local search techniques. Extensive numerical experiments demonstrate that our hybrid Genetic Algorithm is more effective, in terms of both best and average solution quality, compared to the current state of-the-art methods for the DARP and H-DARP, both on existing benchmark instances and on newly generated larger instances for the heterogeneous version of the problem. Also the effectiveness of the crossover operators and local search techniques is validated in this paper.

For future work, we plan to focus on a more complex variant, a Multi-depot H-DARP with synchronization constraints. Other possible directions for future research include the introduction of even more sophisticated local search techniques in the genetic algorithm, and the adaptation and application of our solution method to related problems, such as the Pickup and Delivery Problem with Time Windows.

Acknowledgements

This work is partly supported by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office (research project COMEX, Combinatorial Optimization: Metaheuristics & Exact Methods). The authors thank the reviewers for their input, comments and suggestions. Also, the authors would like to thank Dr. Manar Hosny, Assistant Professor at King Saud University, for her valuable revision and English writing of the paper.

References

Berbeglia, G., Cordeau, J. F., Laporte, G., 2010. Dynamic pickup and delivery problems. *European Journal of Operational Research*, **202**(1), 8-15.

Blanton Jr, J. L., Wainwright, R. L., 1993. Multiple vehicle routing with time and capacity constraints using genetic algorithms. In *Proceedings of the 5th International Conference on Genetic Algorithms* (pp. 452-459). Morgan Kaufmann Publishers Inc.

Borndorfer, R., Grotschel, M., Klostermeiner, F., Kuttner, C., 1997. Telebus Berlin: Vehicle routing scheduling in a dial a ride system. *Konrad Zuse Zentrum für Information Technik Berlin*.

Braekers, K., Caris, A., Janssens, G. K., 2014. Exact and meta-heuristic approach for a general heterogeneous dial-a-ride problem with multiple depots. *Transportation Research Part B: Methodological*, **67**, 166-186.

Buro, M., 1997. Experiments with Multi-ProbCut and a new high-quality evaluation function for Othello. *Games in AI Research*, 77-96.

Chassaing, M., Duhamel, C., Lacomme, P., 2016. An ELS-based approach with dynamic probabilities management in local search for the Dial-A-Ride Problem. *Engineering Applications of Artificial Intelligence*, **48**, 119-133.

Cao, E., Lai, M., 2007. An improved differential evolution algorithm for the vehicle routing problem with simultaneous delivery and pick-up service. In *Third International Conference on Natural Computation* (*ICNC 2007*) (Vol. 3, pp. 436-440). IEEE.

Choi, I. C., Kim, S. I., Kim, H. S., 2003. A genetic algorithm with a mixed region search for the asymmetric traveling salesman problem. *Computers & Operations Research*, **30**(5), 773-786.

Cordeau, J. F., 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Operations Research*, **54**(3), 573-586.

Cordeau, J. F., Laporte, G., 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological*, **37**(6), 579-594.

Cordeau, J. F., Laporte, G., 2007. The dial-a-ride problem: models and algorithms. *Annals of Operations Research*, **153**(1), 29-46.

Cubillos, C., Rodriguez, N., Crawford, B., 2007. A study on Genetic Algorithms for the DARP Problem. In *International Work-Conference on the Interplay Between Natural and Artificial Computation* (pp. 498-507). Springer Berlin Heidelberg.

Doerner, K., Salazar-Gonzalez, J., 2014. In: Toth, P. and Vigo, D. (Eds). Vehicle Routing: Problems, Methods, and Applications, Second Edition. MOSSIAM Series on Optimasation, *Society for Industrial and Applied Mathematics*.

Dongarra, J., 2014. Performance of Various Computers Using Standard Linear Equations Software, (Linpack Benchmark Technical Report, CS-89-85). University of Tennessee, Computer Science Department.

Freitas, A. A., 2013. *Data mining and knowledge discovery with evolutionary algorithms*. Springer Science & Business Media.

Fu, L., 2002. Scheduling dial-a-ride paratransit under time-varying, stochastic congestion. *Transportation* Research *Part B: Methodological*, **36**(6), 485-506.

Holland, J.H., 1975. Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor.

Hosny, M. I., Mumford, C. L., 2010. The single vehicle pickup and delivery problem with time windows: intelligent operators for heuristic and metaheuristic algorithms. *Journal of Heuristics*, **16**(3), 417-439.

Jorgensen, R. M., Larsen, J., Bergvinsdottir, K. B., 2007. Solving the dial-a-ride problem using genetic algorithms. *Journal of the Operational Research Society*, **58**(10), 1321-1331.

Karabuk, S., 2009. A nested decomposition approach for solving the paratransit vehicle scheduling problem. *Transportation Research Part B: Methodological*, **43**(4), 448-465.

Kirchler, D., Calvo, R. W., 2013. A granular tabu search algorithm for the dial-a-ride problem. *Transportation Research Part B: Methodological*, **56**, 120-135.

Koç, Ç., Bektaş, T., Jabali, O., Laporte, G., 2015. A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows. *Computers & Operations Research*, **64**, 11-27.

Li, G., 2009. Research on open vehicle routing problem with time windows based on improved genetic algorithm. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on* (pp. 1-5). IEEE.

Liu, S., Huang, W., Ma, H., 2009. An effective genetic algorithm for the fleet size and mix vehicle routing problems. *Transportation Research Part E: Logistics and Transportation Review*, **45**(3), 434-445.

Liu, R., Xie, X., Augusto, V., Rodriguez, C., 2013. Heuristic algorithms for a vehicle routing problem with simultaneous delivery and pickup and time windows in home health care. *European Journal of Operational Research*, **230**(3), 475-486.

Lin, S., 1965. Computer solutions of the traveling salesman problem. *The Bell System Technical Journal*, **44**(10), 2245-2269.

Lü, Z., Glover, F., Hao, J. K., 2010. A hybrid metaheuristic approach to solving the UBQP problem. *European Journal of Operational Research*, **207**(3), 1254-1262.

Luo, Y., Schonfeld, P., 2011. Online rejected-reinsertion heuristics for dynamic multivehicle dial-a-ride problem. *Transportation Research Record: Journal of the Transportation Research Board*, (2218), 59-67.

Merz, P., Katayama, K., 2004. Memetic algorithms for the unconstrained binary quadratic programming problem. *BioSystems*, **78**(1), 99-118.

Miller, B. L., Goldberg, D. E., 1995. Genetic algorithms, tournament selection, and the effects of noise. *Complex Systems*, **9**(3), 193-212.

Minocha, B., Tripathi, S., 2011. Solution of time constrained vehicle routing problems using multi-objective hybrid genetic algorithm. *International Journal of Computer Science and Information Technologies*, **2**(6), 2671-76.

Muelas, S., LaTorre, A., Peña, J. M., 2013. A variable neighborhood search algorithm for the optimization of a dial-a-ride problem in a large city. *Expert Systems with Applications*, **40**(14), 5516-5531.

Nguyen, P. K., Crainic, T. G., Toulouse, M., 2014. A hybrid generational genetic algorithm for the periodic vehicle routing problem with time windows. *Journal of Heuristics*, **20**(4), 383-416.

Parragh, S. N., 2011. Introducing heterogeneous users and vehicles into models and algorithms for the dial-a-ride problem. *Transportation Research Part C: Emerging Technologies*, **19**(5), 912-930.

Parragh, S. N., Cordeau, J. F., Doerner, K. F., Hartl, R. F., 2012. Models and algorithms for the heterogeneous dial-a-ride problem with driver-related constraints. *OR Spectrum*, **34**(3), 593-633.

Parragh, S. N., Doerner, K. F., Hartl, R. F., 2008. A survey on pickup and delivery problems. *Journal für Betriebswirtschaft*, **58**(1), 21-51.

Parragh, S. N., Doerner, K. F., Hartl, R. F., 2010. Variable neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, **37**(6), 1129-1138.

Parragh, S. N., Schmid, V., 2013. Hybrid column generation and large neighborhood search for the dial-a-ride problem. *Computers & Operations Research*, **40**(1), 490-497.

Potvin, J. Y., Rousseau, J. M., 1995. An exchange heuristic for routeing problems with time windows. *Journal of the Operational Research Society*, **46**(12), 1433-1446.

Prins, C., 2004. A simple and effective evolutionary algorithm for the vehicle routing problem. *Computers & Operations Research*, **31**(12), 1985-2002.

Psaraftis, H. N., 1980. A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. *Transportation Science*, **14**(2), 130-154.

Qu, Y., Bard, J. F., 2013. The heterogeneous pickup and delivery problem with configurable vehicle capacity. *Transportation Research Part C: Emerging Technologies*, **32**, 1-20.

Rekiek, B., Delchambre, A., Saleh, H. A., 2006. Handicapped person transportation: An application of the grouping genetic algorithm. *Engineering Applications of Artificial Intelligence*, **19**(5), 511-520.

Savelsbergh, M. W., 1992. The vehicle routing problem with time windows: Minimizing route duration. *ORSA Journal on Computing*, **4**(2), 146-154.

Schilde, M., Doerner, K. F., Hartl, R. F., 2011. Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. *Computers & Operations Research*, **38**(12), 1719-1730.

Solomon, M. M., 1987. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, **35**(2), 254-265.

Tasan, A. S., Gen, M., 2012. A genetic algorithm based approach to vehicle routing problem with simultaneous pick-up and deliveries. *Computers & Industrial Engineering*, **62**(3), 755-761.

Toth, P., Vigo, D., 1996. Fast local search algorithms for the handicapped persons transportation problem. In *Meta-Heuristics* (pp. 677-690). Springer US.

Toth, P., Vigo, D., 1997. Heuristic algorithms for the handicapped persons transportation problem. *Transportation Science*, **31**(1), 60-71.

Vidal, T., Crainic, T. G., Gendreau, M., Lahrichi, N., Rei, W., 2012. A hybrid genetic algorithm for multidepot and periodic vehicle routing problems. *Operations Research*, **60**(3), 611-624.

Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., 2013. A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows. *Computers & Operations Research*, **40**(1), 475-489.

Wang, H. F., Chen, Y. Y., 2012. A genetic algorithm for the simultaneous delivery and pickup problems with time window. *Computers & Industrial Engineering*, **62**(1), 84-95.

Wilson, N. H., Sussman, J. M., Wong, H. K., Higonnet, T., 1971. *Scheduling algorithms for a dial-a-ride system*. Massachusetts Institute of Technology. Urban Systems Laboratory.

Wink, S., Bäck, T., Emmerich, M., 2012. A meta-genetic algorithm for solving the capacitated vehicle routing problem. In *2012 IEEE Congress on Evolutionary Computation* (pp. 1-8). IEEE.

Wong, K. I., Bell, M. G., 2006. Solution of the Dial-a-Ride Problem with multi-dimensional capacity constraints. *International Transactions in Operational Research*, **13**(3), 195-208.

Xiang, Z., Chu, C., Chen, H., 2006. A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. *European Journal of Operational Research*, **174**(2), 1117-1139.

Zhang, Z., Liu, M., Lim, A., 2015. A memetic algorithm for the patient transportation problem. *Omega*, **54**, 60-71.