Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem

Peer-reviewed author version

Masmoudi, Mohamed Amine; Hosny, Manar; BRAEKERS, Kris & Dammak, Abdelaziz (2016) Three effective metaheuristics to solve the multi-depot multi-trip heterogeneous dial-a-ride problem. In: TRANSPORTATION RESEARCH PART E-LOGISTICS AND TRANSPORTATION REVIEW, 96, p. 60-80.

DOI: 10.1016/j.tre.2016.10.002
Handle: http://hdl.handle.net/1942/22528

Three Effective Metaheuristics to Solve the Multi-Depot Multi-Trip Heterogeneous Dial-a-Ride Problem

Abstract

The Heterogeneous Dial-a-Ride problem (HDARP) is an important problem in reduced mobility transportation. Recently, several extensions have been proposed towards more realistic applications of the problem. In this paper, a new variant called the Multi-Depot Multi-Trip Heterogeneous Dial-a-Ride Problem (MD-MT-HDARP) is considered. A mathematical programming formulation and three metaheuristics are proposed: an improved Adaptive Large Neighborhood Search (ALNS), Hybrid Bees Algorithm with Simulated Annealing (BA-SA), and Hybrid Bees Algorithm with Deterministic Annealing (BA-DA). Extensive experiments show the effectiveness of the proposed algorithms for solving the underlying problem. In addition, they are competitive to the current state-of-the-art algorithm on the MD-HDARP.

Keywords: Vehicle Routing Problem (VRP), Dial-a-Ride Problem (DARP), Metaheuristic, Optimization, Bees Algorithm (BA)

1. Introduction

The transportation of individuals with reduced mobility is an important branch in passenger transportation. It concerns the transportation service that helps the elderly and the disabled people to travel easily and comfortably within their community or to access essential health care services. However, such services are usually very costly, due to specific requirements, such as the need for specially equipped vehicles, highly qualified drivers, accompanying person(s), and the existence of desired service times. Clearly, appropriate transportation planning can greatly improve the service quality and save physical and human resources.

The reduced mobility people transportation planning consists of establishing a transport plan (vehicle itinerary) to meet particular user demands by a fleet of vehicles. This problem is called the Dial-a-Ride problem (DARP) in the literature. The DARP is a variant of the Pickup and Delivery Problem with Time Windows (PDPTW) that concerns the transportation of goods/services between paired pickup and delivery locations. The optimization criterion of the DARP is often to minimize the total transportation distance or duration, while providing high service quality (e.g., small customer waiting times). The DARP is more complicated than the traditional NP-hard Vehicle Routing Problem (VRP), due to its special transportation constraints.

Given the increase in social interest towards improving patients' and disabled persons' transportation conditions, the DARP has indeed many real applications in many countries, such as Italy (Toth and Vigo 1996, 1997), Germany (Borndörfer et al. 1999), US (Karabuk 2009), and Belgium (Rekiek and al. 2006). However, most of the previous research addresses homogeneous vehicles located at a single depot, single

route per vehicle, and only one type of users. Surveys on the Dial-a-Ride-Problem have been presented in Cordeau and Laporte (2007), Parragh et al. (2008), and Doerner and Salazar-Gonzalez (2014).

In recent years, there has been a trend to focus on more realistic versions of the DARP by taking into consideration additional real-life constraints, such as vehicle and user heterogeneity, multiple vehicle depots, and lunch breaks of the drivers. According to Parragh (2011) and Wong and Bell (2006), in reality, the reduced mobility people transportation is often complicated by the presence of several types of users who need specific types of equipment, such as a patient seat, a wheelchair or a stretcher. Consequently, vehicles with special space for this equipment are needed to satisfy different users' requests. This has lead to the emergence of a new variant of the problem, the DARP with heterogeneous users or/and vehicles, which is a generalization of the DARP that has not yet been extensively studied in the literature. Wong and Bell (2006) adopted a parallel insertion heuristic in their solution of the DARP with two types of vehicles (equipped with wheelchair places), and two types of users. The algorithm was tested on artificial instances involving 150 requests. Xiang et al. (2006) developed a parallel insertion heuristic to solve a version of the DARP with several types of users and vehicles. The proposed approach has solved problems between 50 and 2,000 requests. Some authors have also considered realistic real life conditions pertaining to uncertainty and imprecision in pickup and/or drop off times (e.g. Teodorovic and Radivojevic, 2000 and Maalouf et al., 2014), where fuzzy logic has been utilized to provide approximate problem solutions for the dynamic DARP.

A formal definition of the Heterogeneous DARP (HDARP) was first presented in Parragh (2011). In this paper, two types of vehicles are considered; each can have a staff seat, a patient seat, a stretcher and a wheelchair. Also, users can request an accompanying person. The problem is solved using a Branch-and-Cut (B&C) algorithm and Variable Neighborhood Search (VNS) algorithm. The algorithms were tested on 36 instances with up to 4 vehicles and 48 requests, generated from instances proposed by Cordeau (2006) for the homogeneous DARP. Later, Qu and Bard (2013) developed an Adaptive Large Neighborhood Search (ALNS) algorithm to solve the heterogeneous pickup and delivery problem with configurable vehicle capacity. The proposed method was first tested by comparing the obtained results with Parragh (2011). Then, it was applied on data sets provided by the PACE organization. The obtained results are better than the solutions generated manually, with results showing 30–40% cost savings over current practice.

The multi-depot concept has been integrated within the context of the Dial-a-Ride Problem as well to account for the fact that large service providers may have multiple locations for their vehicles, or that drivers are sometimes allowed to take their vehicles home after their shifts. To our knowledge, the only researches that explicitly considered multiple depots for vehicles were conducted by Melachrinoudis et al. (2007), Carnes et al. (2013) and Braekers et al. (2014a). Melachrinoudis et al. (2007) investigated multiple depots and a fleet of vehicles with heterogeneous capacity for aged persons transport from their homes to healthcare centers. A Tabu Search (TS) method was adopted to solve the problem and instances with up to 50 requests were tested. Another practical version of the multi-depot DARP is studied by Carnes et al. (2013). The study aims to build a schedule for air-ambulances assigned to different depots in order to satisfy the demands of a set of patients. To solve the problem, an exact method based on set

partitioning was applied on smaller instances that contain between 10 and 20 requests per day. In the study of Braekers et al. (2014a), multiple depots, heterogeneous vehicles and heterogeneous users are considered. To solve the problem, a Branch-and-Cut and a Deterministic Annealing (DA) algorithm were used to solve small-medium instances containing 2-8 vehicles and 16-96 requests. Their method was also evaluated on the DARP instances proposed by Cordeau and Laporte (2003) and the HDARP instances of Parragh (2011).

Other recent contributions to the literature are made by integrating the multi-trip concept in the DARP and HDARP. This concept is actually very common in practice, where the vehicle can perform several trips per day. Parragh et al. (2012) introduced a variant of the HDARP in which the requirements for assistants and lunch break constraints are taken into account. To solve this problem, a Variable Neighborhood Search (VNS) metaheuristic and a column generation approach are developed and tested on small-medium instances, and both solution methods are combined into a collaborative scheme. Liu et al. (2015) proposed a new variant of realistic DARP that simultaneously considers multiple trips, heterogeneous vehicles/users, a single depot, and configurable vehicle capacity. The authors formulated two mixed integer programming models and introduced 8 classes of valid inequalities to improve the bounds of their Branch-and-Cut algorithm. The proposed method can solve instances with up to 22 requests within four hours. Zhang et al. (2015) studied a real-life public patient transportation problem derived from the Hong Kong Hospital Authority. In their problem, the authors consider that in order to prevent the spread of diseases, ambulances must be returned to the depot during the day for sterilization and for the driver to have a lunch break. A memetic algorithm is developed to solve this problem. The proposed method was tested on real-world data and the standard DARP benchmarks. The results show that the proposed algorithm provides near-optimal solutions for small instances. Another real world application derived from the Non-Emergency Ambulance Transfer Service (NEATS) in Hong Kong is studied by Lim et al. (2016) to solve the Multi-Trip DARP (MT-DARP). Lunch breaks are considered and certain users may require the help of assistants (who occupy a seat during the trip). The assistants can move from a trip of one vehicle to a trip of another vehicle at the depot. The authors consider that the time windows of lunch breaks for the drivers and assistants are heterogeneous. A heuristic with an ad-hoc component to handle the manpower planning problem is developed and tested on a real dataset.

Based on the above literature review, it is concluded that the combination of heterogeneous vehicles/users with either multiple depots or multiple trips has already been studied to some extent. Yet, to the best of our knowledge, a combination of all these aspects has not been studied before. Our newly proposed variant, the Multi-Depot Multi-Trip Heterogeneous Dial-a-Ride Problem (MD-MT-HDARP) with coffee and lunch breaks, aims to fill this gap by combining all these aspects, while accounting for coffee and lunch breaks as well. This problem is very relevant in real life, since these characteristics often occur together in practice. Three metaheuristic-based methods are introduced to solve this new variant, and they are compared in order to decide which one is more effective in solving the problem. The choice of these methods is based on their relative advantages and demonstrated capabilities in solving a variety of VRPs, as described in the excellent survey papers about the implementation of metaheuristics of Laporte et al. (2000), Bräysy et al. (2004a, b), and Pisinger and Ropke (2010). In addition, different types of data

sets are introduced to assess and compare the performance quality of these methods, and to be used as reference for further research.

In reality, the multi-trip feature is needed when the vehicles' fleet size is limited and/or the routes have a limited duration, e.g., due to the existence of lunch breaks which are imposed on the drivers. Our problem is inspired by work on the MT-DARP, in which lunch breaks are considered, such as in Lim et al. (2016) and Zhang et al. (2015). However, here we take into consideration a lunch break and two coffee breaks during the working day of a driver. Each vehicle/driver can perform two trips during its working day: the first trip (r=1) is performed in the morning (before the lunch break), and the second (r=2) is performed in the afternoon (after the lunch break).

During each trip, a short coffee break should be included as well. The motivation behind imposing the requirement for breaks can be related to security issues, e.g., providing comfortable conditions to drivers by ensuring sufficient rest during their working day, especially when traveling long distances (Goel 2010). In this study, we address a generalized case of the MT-HDARP, called the MD-MT-HDARP, in which we consider the traditional definition of the HDARP proposed by Parragh (2011), where heterogeneous vehicles and multiple request types are treated. In addition, inspired by the multi-depot problem of Braekers et al. (2014a), we integrate the multi-depot concept, which makes the problem more challenging and sophisticated than the single-depot DARP. To summarize, the Multi-Depot Multi-Trip Heterogeneous Dial-a-Ride Problem (MD-MT-HDARP) is considered as a combination of MT-HDARP and MD-HDARP. To the best of our knowledge, no work has been conducted to address this problem.

The contributions of this work are as follows: *i*) a general static Multi-Depot Multi-Trip Heterogeneous Dial-a-Ride Problem is introduced and a mathematical formulation of the problem is proposed; *ii*) an Adaptive Large Neighborhood Search (ALNS) and two effective hybrid metaheuristic methods (Hybrid Bees Algorithm with Deterministic Annealing (BA-DA) and Hybrid Bees Algorithm with Simulated Annealing (BA-SA)), are proposed to solve this problem; *iii*) A set of new small, medium and large instances is generated for the MD-MT-HDARP, based on the instances of Parragh (2011) and Braekers et al. (2014a); *iv*) Numerical experiments are applied to demonstrate that all three solution approaches provide high quality solutions both for these new instances and for existing benchmark instances on the related MD-DARP; in particular the two hybrid methods show their efficiency for large instances of the MD-MT-HDARP.

The rest of the paper is organized as follows. Section 2 provides the mathematical formulation of the problem. Section 3 describes the construction heuristics. Sections 4 and 5 describe our proposed solution approaches. Section 6 explains the solution evaluation function. Section 7 reports the computational experimentation, followed by the conclusions in Section 8.

2. Problem description

The MD-MT-HDARP can be formally described as follows. Let $G = (V^+, A)$ be a directed graph. The set V^+ is further partitioned into three subsets; M, N, M'. $M = \{v_1, \dots, v_m\}$ is the set of m (starting) depots. $N = \{v_{m+1}, \dots, v_{m+2n}\}$ corresponds to n users to be served, with $P = \{m+1, \dots, m+n\}$ and $D = \{m+n+1, \dots, m+2n\}$ the sets of vertices corresponding to pickup and delivery locations,

respectively. $M' = \{v_{m+2n+1}, \dots, v_{2n+2m}\}$ is the set of m (finishing) depots. Let $V = M \cup N$ and $V' = M' \cup N$, and $A = \{(i,j: i \in V, j \in V', i \neq j\}$ the set of arcs connecting each pair of nodes. Each user request involves transportation from pickup node $i \in P$ to delivery node $i + n \in D$. To each arc $(i,j) \in A$ are associated a non-negative travel time t_{ij} and a routing cost c_{ij}. A fleet of heterogeneous vehicles K is available at the m depots to serve the n users. Each vehicle $k \in K$ starts from a depot and ends at the same depot of departure (starts at i and finishes at $2n+m+i$, where $i=1,\dots,m$). Each depot has a limited number of vehicles P_m. The starting depot and returning depot of a vehicle k are represented by $m_{(k)}$ and $m'_{(k)}$, respectively. We impose that $m'_{(k)} = 2n + m + m_{(k)}$. Each vehicle k has a capacity Q^{ek} that gives the amount of equipment e available on the vehicle, where e = {0, 1, 2, 3} denotes the types of equipment. Let q_i^e denote the load of passenger i for equipment e; e.g., q_i^0= 1, q_i^1= 1, q_i^2= 1 and $q_i^3 = 1$ $(i = 1, \dots, n)$ mean that passenger i needs an accompanying person seat, a disabled seat, a stretcher and a wheelchair, respectively. Each delivery node $i + n \in D$ has a load $q_{i+n}^e = -q_i^e$. Every user either specifies a time window $[e_i, l_i]$ on the departure (pickup) or the arrival (delivery). The service has to start within this time window; i.e., if the vehicle arrives earlier than e_i, it must wait. At each node, loading or unloading operations last for a given service time s_i. A maximum ride time for each user, denoted by L_{max}, is considered in order to provide high service quality.

Note that each vehicle is assumed to have a fixed driver. Hence, both terms (vehicles and drivers) are used interchangeably. For each driver, the maximum working time per day is limited by T_{max}. During this time, each driver can perform up to two trips, e.g., one in the morning and one in the afternoon. All drivers must have a lunch break at their depot between these two trips. This lunch break has to start and end within a given time window $[EL, LL]$ and has a duration of TL =30 minutes. Additionally, during each trip $r \in \{1,2\}$, a driver should take a coffee break of duration TC within a time window $[EC^r, LC^r]$. For example, during the morning trip, the driver must take a coffee break of 15 minutes within the time window [10 am, 10:30 am].

Because of the presence of ride times, time windows, breaks, and the limited duration of each working day, the scheduling subproblem and hence checking the feasibility of a solution is more complex than in other routing problems. A feasible solution consists of a set of routes satisfying the following constraints: (*i*) a pickup node and its corresponding delivery node must be visited in the same trip, and the pickup node must be visited before the delivery node; (*ii*) the vehicle capacity must be respected at each node for each type of equipment; (*iii*) each node must be visited in its time window, so if a vehicle comes early, it must wait until the beginning of the time window; (*iv*) the ride time of any user should not be exceeded; (*v*) the total duration of the working day of each vehicle (maximum route duration) is strictly limited by T_{max}; (*vi*) every vehicle starts and ends at the same depot; (*vii*) the schedule should include a lunch break and two coffee breaks (in the morning and afternoon).

The MD-MT-HDARP consists of designing the routes to satisfy all requests while minimizing the routing costs. The variables used in the mathematical model are shown in Table 1.

Table 1
Definition of variables used for the MD-MT- HDARP

x_{ij}^k: A binary variable that is equal to 1 if arc (i,j) is traversed by vehicle k, and 0 otherwise;

λ_i^{1k}: A binary variable that is equal to 1 if the first (morning) coffee break of vehicle k is held either directly after servicing node i or directly after traveling from node i to the next node, 0 otherwise;

λ_i^{2k}: A binary variable that is equal to 1 if the second (afternoon) coffee break of vehicle k is held either directly after servicing node i or directly after traveling from node i to the next node, 0 otherwise;

μ_{ij}^k: A binary variable that is equal to 1 if the lunch break of vehicle k is held directly after node i and before traveling to node j, 0 otherwise;

B_i^k: A non-negative variable indicating the time that vehicle k begins service at node i;

$SB_i^{1,k}$: A non-negative variable indicating the time that vehicle k begins the first coffee break;

$SB_i^{2,k}$: A non-negative variable indicating the time that vehicle k begins the second coffee break;

Q_i^{ek}: A non-negative variable indicating the load of equipment e on vehicle k immediately after visiting node i;

L_i^k: A non-negative variable indicating the ride time of user $i \in P$ on vehicle k;

δ_i^k: A binary variable that is equal to 1 if vehicle k is not empty directly after i, 0 otherwise.

θ_i^1 (θ_i^2): A binary variable (only relevant when the coffee break is taken after node i) that is 0 if the coffee break is taken directly after service at node i, and 1 if the coffee break is taken after service at node i and traveling to the next node.

Inspired by the formulations for the HDARP of Parragh (2011) and the MT-DARP of Zhang et al. (2015), the MD-MT-HDARP can be formulated as follows:

Minimize $\sum_{k\in K}\sum_{i\in V}\sum_{j\in V'} c_{ij}x_{ij}^k + \sum_{k\in K}\sum_{i\in V}\sum_{j\in V'}(c_{i,m'_{(k)}} + c_{m_{(k),j}} - c_{ij})\mu_{ij}^k$ (1)

Subject to

$\sum_{k\in K}\sum_{j\in N} x_{ij}^k = 1$ $\forall i \in P$ (2)

$\sum_{k\in K}\sum_{j\in N} x_{ji}^k = 1$ $\forall i \in D$ (3)

$\sum_{j\in N} x_{ij}^k = \sum_{j\in N} x_{j,n+i}^k$ $\forall k \in K, \forall i \in P$ (4)

$\sum_{j\in P\cup\{m'_{(k)}\}} x_{m_{(k)},j}^k = 1$ $\forall k \in K$ (5)

$\sum_{i\in D\cup\{m_{(k)}\}} x_{i,m'_{(k)}}^k = 1$ $\forall k \in K$ (6)

$\sum_{j\in V} x_{ji}^k = \sum_{j\in V'} x_{ij}^k$ $\forall k \in K, \forall i \in N$ (7)

$x_{ij}^k = 1 \Rightarrow Q_j^{ek} \geq q_j^e + Q_i^{ek}$ $\forall k \in K, \forall (i,j) \in A, e \in \{0,1,2,3\}$ (8)

$0 \leq Q_i^{ek} \leq Q^{ek}$ $\forall k \in K, \forall i \in N, e \in \{0,1,2,3\}$ (9)

$Q_{m_{(k)}}^{ek} = 0$ $\forall k \in K, e \in \{0,1,2,3\}$ (10)

$L_i^k = B_{n+i}^k - (B_i^k + s_i \sum_{j\in N} x_{ij}^k)$ $\forall k \in K, \forall i \in P$ (11)

$t_{i,n+i} \leq L_i^k \leq L_{\max}$ $\forall k \in K, \forall i \in P$ (12)

$x_{ij}^k = 1 \Rightarrow B_j^k \geq B_i^k + s_i + t_{ij}$ $\forall k \in K, \forall (i,j) \in A$ (13)

$x_{ij}^k = 1 \wedge (\lambda_i^{1k} = 1 \vee \lambda_i^{2k} = 1) \Rightarrow B_j^k \geq B_i^k + s_i + t_{ij} + TC$ $\forall k \in K, \forall (i,j) \in A$ (14)

$\mu_{ij}^k = 1 \Rightarrow B_j^k \geq B_i^k + s_i + t_{i,m'_{(k)}} + TL + t_{m_{(k)}j}$ $\forall k \in K, \forall (i,j) \in A$ (15)

$e_i \leq B_i^k \leq l_i$ $\forall k \in K, \forall i \in V^+$ (16)

$\lambda_i^{1k} = 1 \Rightarrow EC^1 \leq SB_i^{1,k} \leq LC^1$ $\forall k \in K, \forall i \in V^+$ (17)

$\lambda_i^{1k} = 1 \wedge x_{ij}^k = 1 \Rightarrow SB_i^{1,k} = B_i^k + s_i + \theta_i^1 t_{ij}^k$ $\forall k \in K, \forall (i,j) \in A$ (18)

$\lambda_i^{2k} = 1 \Rightarrow EC^2 \leq SB_i^{2,k} \leq LC^2$ $\forall k \in K, \forall i \in V^+$ (19)

$\lambda_i^{2k} = 1 \wedge x_{ij}^k = 1 \Rightarrow SB_i^{2,k} = B_i^k + s_i + \theta_i^2 t_{ij}^k$ $\forall k \in K, \forall (i,j) \in A$ (20)

$$\sum_{j \in V'} \mu_{ij}^k = 1 \Rightarrow EL \le B_i^k + s_i + t_{i,m'_{(k)}}^k \le LL \qquad \forall k \in K, \forall i \in V \qquad (21)$$

$$B_{m'_{(k)}}^k - B_{m_{(k)}}^k \le T_{\max} \qquad \forall k \in K \qquad (22)$$

$$\lambda_i^{1k} = 1 \qquad \forall k \in K, \forall i \in V \qquad (23)$$

$$\lambda_i^{2k} = 1 \qquad \forall k \in K, \forall i \in V' \qquad (24)$$

$$\sum_{(i,j) \in A} \mu_{ij}^k = 1 \qquad \forall k \in K \qquad (25)$$

$$\mu_{ij}^k \le x_{ij}^k \qquad \forall k \in K, (i,j) \in A \qquad (26)$$

$$\sum_{j \in V'} \mu_{ij}^k \le 1 - \delta_i^k \qquad \forall k \in K, \forall i \in V^+ \qquad (27)$$

$$\sum_{e=0}^{3} Q_i^{ek} \le \sum_{e=0}^{3} Q^{ek} \delta_i^k \qquad \forall k \in K, i \in V^+ \qquad (28)$$

$$x_{ij}^k, \mu_{ij}^k \in \{0,1\} \qquad \forall k \in K, \forall (i,j) \in A \qquad (29)$$

$$\delta_i^k, \lambda_i^{1k}, \lambda_i^{2k} \in \{0,1\} \qquad \forall k \in K, \forall i \in V^+ \qquad (30)$$

$$\theta_i^1, \theta_i^2 \in \{0,1\} \qquad \forall i \in V^+ \qquad (31)$$

The objective function (1) is minimizing the total routing costs. Constraints (2)-(4) guarantee that each pickup and delivery pair must be served by exactly one and the same vehicle. Constraints (5)-(6) guarantee that each vehicle k starts at the origin depot and ends at the corresponding destination depot, while constraint (7) ensures flow conservation. Constraints (8) and (9) ensure that the capacity constraint is respected. Constraint (10) sets the load variable of each depot to zero, ensuring that the vehicle leaves the depot with empty load. Constraint (11) represents the ride time of the user on the route, which is bounded by constraint (12). The latter two constraints also ensure that the precedence relationship between the pickup and delivery nodes is respected. Constraints (13), (14) and (15) define the beginning of service at each node and the consistency of the time variables, while accounting for coffee and lunch breaks. Also, these constraints ensure subtour elimination. Constraint (16) imposes time window observance. Furthermore, constraints (17), (18), (19) and (20) guarantee that the coffee break between two services i and j starts before or after the transportation between the two services within a coffee break time window $[EC^r, LC^r]$ ($r \in \{1, 2\}$). Constraint (21) enforces that each driver must take a lunch break at the depot of departure during the working day in time window [*EL*, *LL*]. Total route duration is limited by (22). Constraints (23) and (24) guarantee that the coffee breaks are planned, while constraints (25) force drivers to have a lunch break. Constraints (26), (27) and (28) guarantee that the lunch break can only take place after a node where the vehicle is empty. Finally, constraints (29), (30) and (31) guarantee that the decision variables are binary.

Note that constraints (8), (13)-(15) and (17)-(20) can easily be linearized using the big *M*- method.

3. Construction heuristic

To obtain an initial solution to the MD-MT-HDARP, an effective heuristic that is inspired by Braekers et al. (2014a) is used.

First, the heuristic creates a list *L* of users to be served. Then, we repeat the following steps: an empty route is initialized with time windows for coffee and lunch breaks pre-defined in the route. The insertion of users is performed by adding a user to the route in the following manner: a user *i* is selected at random from the list *L* and inserted in already existing routes, such that its pickup and delivery nodes are inserted

in their best positions, while respecting the feasibility of the solution and taking into account time windows of coffee and lunch breaks. A new vehicle is added in case a user request cannot be inserted in one of the existing routes. This procedure is applied until all users have been served. An example of an initial solution is shown in Figure 1. It contains two routes, which are served by vehicles V1 and V2, assigned to depots m=1 and m=2, respectively. Each route is divided into two trips (r=1, 2), where each trip consists of a set of pickup (i^+) nodes and delivery (i^-) nodes for each user i. Between two consecutive trips, the driver of the vehicle must take a lunch break at its home depot. In addition, for each trip ($\forall\, r = 1,2$) a coffee break Cof is taken into account.

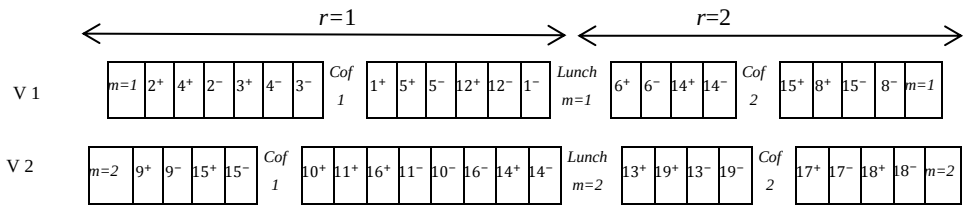

Figure 1: An example of initial solution

We note that our construction heuristic is applied to generate each solution in the population N of our hybrid Bees Algorithm (BA). For the ALNS algorithm, this heuristic is applied only once to generate the starting solution.

4. Hybrid Bees Algorithms for the MD-MT-HDARP

In this section, we propose two versions of a hybrid Bees Algorithm (BA). The BA is a metaheuristic algorithm that is inspired from the behavior of honey bees in nature while they search for food. There are several versions of Bees-inspired Algorithms in the literature, but we follow here the method proposed by Pham et al. (2005). The BA can be also considered as an improved version on the well-known Genetic Algorithm as studied in Yuce et al. (2013). The BA has been tried on some hard optimization problems, producing good results (see, e.g. Pham et al., 2005; Yuce et al., 2013). We have chosen in this research the BA as our proposed approach, for the following reasons:

- The steps of the BA are simple and easy to understand.
- The BA combines random search with neighborhood search. This has the advantage of providing a balance between intensification and diversification, which is needed in any effective metaheuristic (Talbi, 2009).
- BA has a distinguishing feature than other bees-inspired algorithms, which is allowing more intensification of the search around elite solutions (best of the best). This property can help the algorithm discover better solutions more rapidly.
- The BA can be easily hybridized with other search methods. For example, the intensification around the elite solutions can be conducted using any single solution based metaheuristic. Hybridization of more than one metaheuristic is beneficial in handling hard optimization problems, as is the case in our problem.

Since we didn't encounter any application of the BA on the Dial-a-ride problem (DARP) or its variants, we reckoned that it is an attractive research area for further investigation. Since the BA is a

population based metaheuristic, it is more diversification oriented, while being relatively poor in exploitation. To overcome this weakness, we propose a modification of the classical BA by hybridizing it with single-solution based metaheuristics, to help it converge more rapidly to good solutions. We selected Deterministic Annealing (DA) and Simulated Annealing (SA) for this purpose. Our strategies are referred to as Hybrid Bees Algorithm with DA (BA-DA) and Hybrid Bees Algorithm with SA (BA-SA). For more details about the BA, the interested readers are referred to Pham et al. (2005).

The main steps of the proposed hybrid BA are shown in Algorithm 1 and the flowchart of the algorithm is given in Figure 2.

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

Begin
Initial population: Generate the initial population of *N* solutions (by a construction heuristic);
Repeat
Step 1: Evaluate the fitness of each solution in the population *N*
Step 2: Select (be) solutions from the current population using tournament selection, and sort them in ascending order
Step 3**:** Diversification and Improvement phase
Step 3.1: Apply DA (SA) on each of the best (es) (from the (be)) solutions.
Step 3.2**:** Improve the quality of each $(be - es)$ solution using local search.
Step 4**:** Memorize the (be) new solutions and select the best one
if the new best solution is better than the best so far one **Then**
replace the best so far solution with this new one
End if
Step 5: Insert the (be) new solutions in the population *N*
Step 6: Generate $(N - be)$ new solutions to complete the population *N*
Until No improvement after ten consecutive iterations
Output best solution
End.

The BA starts with an effective heuristic to generate the initial population of size *N* (see Section 3). Then, for a pre-specified number of iterations, the following steps are followed: In Step 1, all solutions in the population *N* are evaluated based on fitness. In Step 2, we select a set of (be) best solutions using tournament selection, which are then ordered according to fitness value (from lowest to highest). In Step 3, we select the first (es) solutions from (be) to be improved. Improvement is done by applying one of our single-solution based metaheuristics (DA/SA) on the set (es), in order to obtain high quality solutions. In contrast, the remaining $(be - es)$ solutions are improved by simple local search techniques. In Step 4, (be) new best solutions are obtained and the best of them is selected. If the solution is better than the best so far solution, it replaces the best solution. In Step 5, the (be) new solutions obtained after improvement are injected into population *N*, so that the best solutions survive to the next iteration. In Step 6, to complete the population *N*, $(N - be)$ new solutions are created by a simple heuristic. The algorithm stops when the best solution is not improved for ten consecutive iterations. Finally, the best solution is returned as the problem solution.

It should be noted that in our algorithm we only keep feasible solutions, i.e., solutions that violate any constraints are always rejected. Therefore, we need to frequently check the feasibility of routes. The procedure of checking the feasibility of a route is rather complicated, due the existence of ride time constraints, along with time windows and route time limits, in comparison with other routing problems (Parragh and Schmid, 2013). In this paper, the eight-step evaluation scheme proposed by Parragh et al.

(2010), to test the feasibility of a given path for the DARP introduced by Cordeau and Laporte (2003), is applied and described in Section 6.

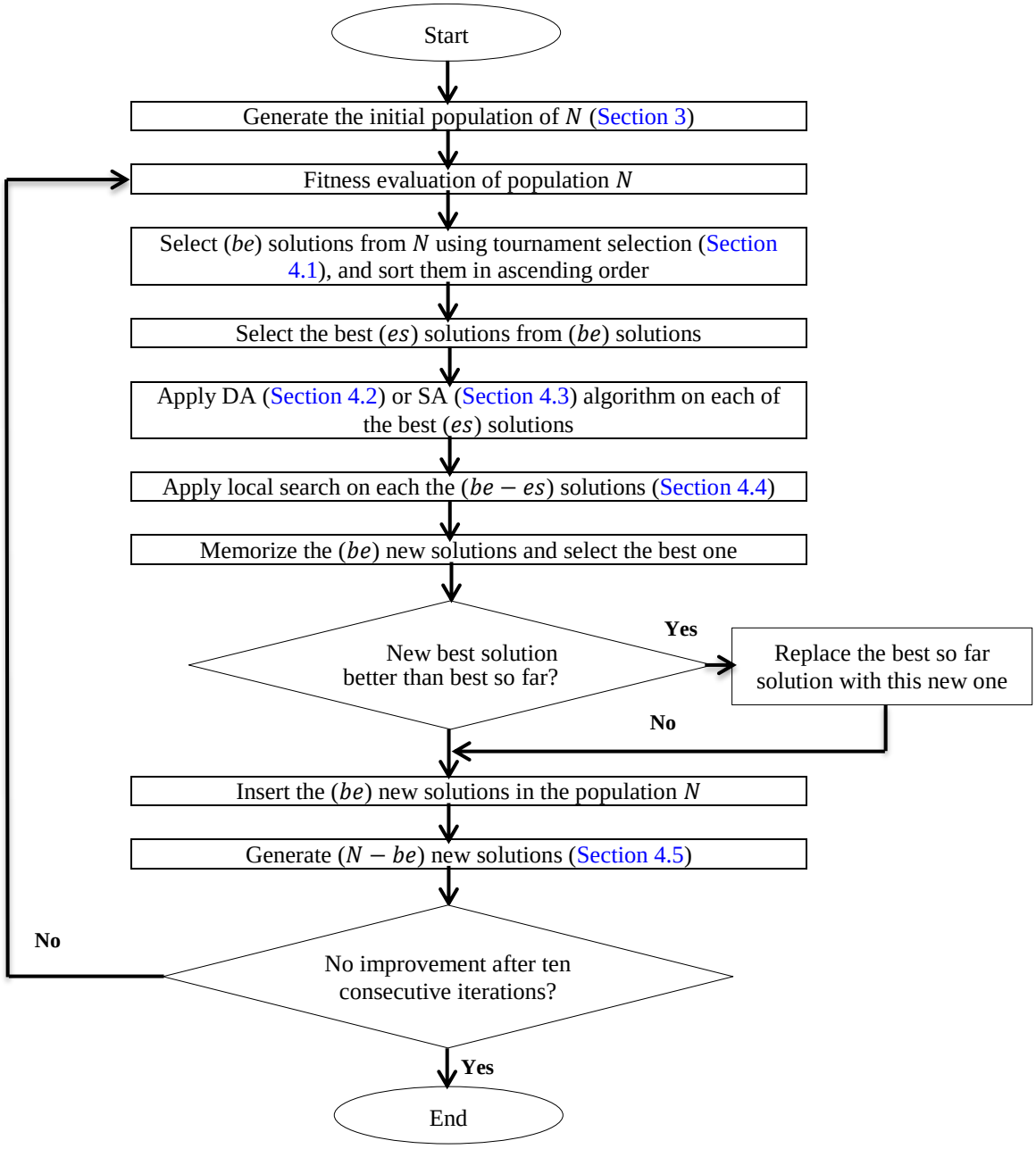

Figure 2: The flowchart of our hybrid Bees Algorithm

4.1. Selection: Tournament

To select the best solutions from the population N (Step 2 of Algorithm 1), we adopted the selection by tournament as proposed by (Miller et Goldberg, 1995), which is one of the most common selection methods in evolutionary algorithms. This type of selection has demonstrated great effectiveness for the selection of good solutions at each new generation (Freitas, 2013) for several transportation problems. The principle is to randomly choose a subset of solutions s from the population N, and then select the best individual in the group who has the highest fitness value. This process is repeated until the number of solutions required (be) is attained. Here, the tournament size s is considered to be seven.

4.2. Deterministic Annealing

Deterministic Annealing (DA), also known as Threshold Accepting, is applied as an improvement phase on each of the (*es*) solutions obtained by our Bees Algorithm (Step 3.1 in Algorithm 1). This method was first proposed by Dueck and Scheuer (1990), as a variant of Simulated Annealing (SA). In each step of the DA, a new solution *x'* is generated from the current solution *x*. If the objective value of *x'* is better than that of the current solution *x*, this solution is accepted. Else, *x'* is accepted provided that the deterioration of the objective function value, calculated by $\Delta = C(x') - C(x)$, is less than the threshold value *T*. This threshold value is gradually reduced during the search until only solutions that improve the objective function are accepted (Caris and Janssens, 2010). The DA algorithm has demonstrated effectiveness for solving a variety of routing problems, by providing high quality solutions within a short computation time (Tarantilis et al., 2004; Bräysy et al., 2008; Nikolakopoulos and Sarimveis, 2007; Braekers et al., 2013, 2014a, 2014b).

The benefit of the DA method is that it is simple to understand and apply. Another advantage is that it relies on only one parameter (*T*). Thus, it requires much less parameter tuning in comparison with other algorithms (Braekers et al., 2013, 2014a, 2014b). In addition, the deterministic acceptance function of the DA is computationally simpler than the stochastic function of the SA, which may speed up the performance of the algorithm (Talbi, 2009).

The proposed framework of the DA metaheuristic is presented in Algorithm 2 and is based on Braekers et al. (2014a). It is applied to each of the *es* solutions individually. Let x be the current solution and x_{best} the best solution found so far (initially set to the initial solution obtained from the considered es solution). Threshold T is set to its maximum value T_{max}. The proposed DA algorithm runs for a number consecutive steps n_{DA}. At each step, several local search operators (I1, I2, I3, and I4) are applied in a random order on the current solution. Each operator returns a new solution x', which is checked for feasibility and evaluated using the evaluation function described in subsection 3.4. If *x'* is feasible and the objective function value of x' is less than that of x plus the threshold T, x' becomes the new current solution. When accepting the new solution x', it is verified if a new best global solution x_{best} has been found; in this case, this solution becomes the new best solution. In case we did not obtain a new best solution, the threshold T is reduced by the threshold reduction parameter ΔT. Every time T becomes negative, it is reset to $T_{max} * \beta$, where β is a randomly generated number between 0 and 1. In case T becomes negative and no improvement of the best solution x_{best} has been found for a consecutive number of iterations n_{imp}, the search is restarted from the best solution.

Algorithm 2: Pseudo-code of the proposed DA algorithm

Initialization: Threshold $T = T_{max}$, $i_{last} = 0$ and $x = x_{best}$ = the current solution selected from the (*es*) solutions;
Repeat
$i_{last} \leftarrow i_{last} + 1$
For j = 1 to n_{oper} **do**
Perform a local search operator (I1, I2, I3 or I4) on x to obtain a new solution *x'* and accept or reject *x'*
If *x'* is accepted **then**
$x \leftarrow x'$
If $C(x) < C(x_{best})$ **then**
$x_{best} \leftarrow x$

$i_{last} \leftarrow 0$
End if
End if
End for
If $i_{last} > 0$ **then**
$T \leftarrow T - \Delta T$
If $T < 0$ **then**
$\beta \leftarrow$Random number in the range [0,1]
$T \leftarrow \beta \times T_{max}$
If $i_{last} > n_{imp}$ **then**
$x \leftarrow x_{best}$
$i_{last} \leftarrow 0$
End if
End if
End if
Until the number of steps n_{DA} is reached
Return x_{best}

4.3. Simulated Annealing

The second metaheuristic that was hybridized with the BA in our work is Simulated Annealing (SA). Similar to DA, it is applied to improve each of the best (es) solutions of the general BA. The SA method was first proposed by Kirkpatrick et al. (1983). In recent years, this method has been successfully applied to solve a wide number of problems with complex search spaces (Lin and Vincent, 2015; Xiao and Konak, 2015 and Vincent et al., 2016).

The SA algorithm typically performs better than simple local search techniques, due to its ability to avoid the trap of local optima (Busetti, 2003). Unlike simple local search, the algorithm does not only accept solutions that improve the objective function; rather, using the Boltzmann function of Metropolis et al. (1953), it occasionally accepts worse solutions as well, which may help in jumping out of local optima and diversify the search towards more promising solution areas.

Our SA structure is inspired by the SA algorithm proposed by Vincent et al. (2016). Let x_{best} and x be the initial solution and T_0 the initial temperature. At each temperature T, n_{seq} iterations are considered. To explore the search space further around each solution, our SA considers four different moves (I1, I2, I3 and I4). During each iteration, our SA selects one of them randomly to explore a larger search space. After applying the local search operator, if the new solution is better than the current solution (i.e. the cost is lower), it is accepted. On the other hand, if the cost is higher, the new solution may be accepted subject to the simulated annealing acceptance criterion proposed by Metropolis et al. (1953) e^{Δ/T_i}, where $\Delta = f(x) - f(x')$ represents the difference in the objective function between the current solution x and the new one, and T_i is the current temperature. The temperature cooling schedule is as follows: $T_i = \delta * T_{i-1}$, where δ is the cooling rate, and i is the iteration number. If the new solution obtained from the local search move is better than x_{best}, it replaces x_{best}.

The proposed algorithm is terminated when T_i value becomes less than 0.01. The framework of the proposed SA is shown in the Algorithm 3.

Algorithm 3: Pseudo-code of the proposed SA algorithm

Initialization: T_i=T_0 and x_{best} =x = the current initial solution from the *es* solutions
Repeat
For i =1 to n_{seq} **Do**
Perform a local search move {I1, I2, I3 or I4} on the current solution x to obtain x'

If *x'* is feasible and accepted **Then**
$x \leftarrow x'$
If $C(x) < C(x_{best})$ **then**
$x_{best} \leftarrow x$
End if
End If
End for
$T_i = \delta * T_{i-1}$
Until T_i value is less than 0.01
Return x_{best}

4.4. Local search operators

During many steps of our developed methods, intra- and inter-route movements of users/edges are performed in order to explore the search space of the current solution. Four local search operators are applied. It should be noted that in the proposed operators, we adopt a best insertion strategy, in which all combinations of insertion positions for the pickup and delivery nodes of user i in the current route or in other routes are evaluated and checked for feasibility by the scheduling algorithm of Parragh et al. (2010).

2-opt operator (I1): This operator is proposed in Lin (1965). It consists of replacing two edges with two new ones. One route from the solution is selected randomly in each iteration to apply this move.

Relocation operator inter-vehicle (I2): This operator relocates a user's pickup and delivery nodes assigned to one vehicle (selected randomly) to another. The nodes are inserted in their best possible position in another route (vehicle) if possible.

Relocation operator intra-vehicle (I3): This one is similar to the previously described operator, but it is applied in the same vehicle. This operator is applied on each user in the vehicle by removing the user and reinserting it in the best possible position. In this case, three types of moves of a user *i* are considered: the first is to relocate only the pickup node; the second is to relocate only the delivery node; while the third is to relocate both, with the delivery node inserted immediately after the pickup node.

Remove two insert one operator (I4): This operator is adopted from Xiang et al. (2006). It consists of removing two randomly selected users from the vehicle and inserting them one by one in other vehicles, while maintaining feasibility. This operator is applied for all vehicles belonging to one depot selected randomly.

4.5. Local search strategy for the hybrid BA

In addition to using DA or SA in our hybrid BA algorithm, a local search procedure is applied for a fixed number of iterations (50) on the $(be - es)$ solutions in order to explore the local search space further around these solutions. At each iteration, the local search operators I1, I2, I3 and I4 are executed in a random order to obtain a higher quality solution.

4.6. Generation of new solutions

Step 5 of Algorithm 1 has generated *be* new solutions that are inserted in the population. To complete the population *N*, we need to generate (*N-be*) new solutions (step 6 of Algorithm 1). A simple heuristic is applied to try to obtain high-quality solutions for the next generation of the Bees Algorithm. For each of the old $(N\text{-}be)$ solutions, u users are selected randomly from the solution; they are removed and added to

the removal list *L*. Then, all users from *L* are re-inserted at the most appropriate insertion positions in all vehicles (including the ones from which users were removed), while respecting the feasibility of the solution.

5. Adaptive Large Neighborhood Search Algorithm for the MD-MT-HDARP

In this section, we present an Adaptive Large neighborhood Search (ALNS) algorithm for solving the MD-MT-HDARP. The ALNS heuristic was first proposed by Ropke and Pisinger (2006a), as an extension of Large Neighborhood Search (LNS) proposed by Shaw (1998). Many researchers have highlighted the excellent capabilities of ALNS in solving large-size and hard optimization problems, such as Pickup and Delivery Problems (see, e.g., Ropke and Pisinger, 2006a, b; and Li et al., 2015; and Ghilas et al., 2016) and Dial-a-Ride Problems (see, e.g., Qu and Bard, 2013; Masson et al., 2013; and Li et al., 2016). The advantage of ALNS is that, in each iteration, only a smaller size decomposition of the problem is investigated, which can be more efficient than an aggregate problem solving method (Schrimpf et al., 2000).

The structure of our ALNS algorithm is similar to that proposed by Li et al. (2015) and Li et al. (2016), and is shown in Algorithm 4. To find a global best solution x^* with cost $f(x^*)$, the algorithm executes n_{ALNS} iterations. Let x be the initial solution of our ALNS to which also the current best solution (x_{best}) is initialized. First, the temperature T is initialized to T_{max} and each removal and insertion operator's weight is initialized. The weights and scores of the removal and insertion operators are updated during the search. In each iteration, in principle, one removal and one insertion operator are applied. However, if no improvement of x_{best} is obtained in the last ten iterations, the algorithm performs two removal operators and one insertion operator. This technique is similar to the one proposed by Li et al. (2015). The choice of the operators in each iteration is based on the roulette wheel mechanism (described in the subsection 5.1). When a new solution x' is obtained, we decide to accept or reject it as follows: if the objective function of x' is better than that of the current solution, x' is accepted and becomes the current solution; otherwise, x' is accepted only if it satisfies the SA acceptance criterion $e^{(f(x)-f(x')/T)}$. The temperature reduction factor α was set to 0.99975, as suggested by Ropke and Pisinger (2006a).

Algorithm 4: Pseudo-code of the ALNS algorithm

Initialize: $x = x_{best}$ = initial solution obtained by our construction heuristic, weights of removal and insertion operators to initial values, temperature $T = T_{max}$; cooling rate α =0.99975
Repeat
Select and perform the removal operator(s) on the current solution *x*;
Select and perform one insertion operator in order to obtain a new solution *x';*
If *x'* is feasible and accepted **Then**
$x \leftarrow x'$
If $C(x) < C(x_{best})$ **then**
$x_{best} \leftarrow x$
End if
End If
$T = \alpha * T$
Adjust the weights of removal and insertion operators using the scores obtained after n_{seq} consecutive iterations
Reset the scores of the removal and insertion operators to zero after n_{seq} consecutive iterations
Until number of iterations n_{ALNS} is reached
Output x_{best}

5.1. Adaptive weight adjustment procedure

We have five removal and four insertion operators in our ALNS. To choose among them, we use roulette wheel selection. Following Ropke and Pisinger (2006a), the probability of choosing an operator d at iteration t is given by: $P_d^{t+1} = P_d^t(1 - r_p) + r_p\pi_i/\vartheta_i$, where $r_p$is the roulette wheel parameter, π_i is the score of operator i, and ϑ_i is the number of times operator i has been used in the last n_{seq} iterations. The score of an operator is increased by π_1, if the current pair of removal-insertion operators finds a new best solution. On the other hand, if an improved solution is found, the score is increased by π_2, while if a non-improving solution that is accepted through the SA acceptance condition is found, the score of the operator is increased by π_3. After n_{seq} iterations, the new weights are adjusted using the calculated scores, and the scores of the removal and insertion operators are reset to zero.

5.2. Removal and insertion operators

In each iteration, a set of nodes/users are removed from a current solution x and added to a list L by some removal operators, in order to re-insert them using several repair operators to obtain a new solution x'. All operators are inspired by and adopted from the literature such as Ropke and Pisinger (2006 a, b), Pisinger and Ropke (2007) and Demir et al. (2012). The first five operators (R1 to R5) are applied to destroy the current solution x. While, the latter four operators (P1 to P4) are implemented in order to re-insert the removed nodes/users, forming a new solution.

Random-user (R1): This operator consists of randomly selecting u users from the solution and putting them in list L, thus attempting to help the diversification mechanism.

Path-removal (R2): This operator is inspired from Demir et al. (2012). Let route r ={0, i,…,n,n+1,…,2n,0} contain n users to be served. Each user i is associated with a pickup node $i^+ \in P$ and a delivery node $i^- \in D$. We denote by $\varphi(i)$ a path that starts at node i^+ and ends at node i^-. The principle of this operator is to remove the path $\varphi(i)$ of a randomly chosen user i and then to insert all users in this path in the list L.

Related removal (R3): This operator is based on the Shaw removal operator proposed by Ropke and Pisinger (2006a). After selecting a random user i, we define the relatedness $R(i, j)$ between two requests as the distance between users i and j, which is used for removing nodes. The relatedness function is given by: $R(i,j) = d_{ij}^k - d_{i+n,j+n}^k + \rho(|B_i^k - B_j^k| + |B_{i+n}^k - B_{j+n}^k|)$, where d_{ij}^k is the distance between i and j, and ρ is a control parameter with $\rho = [0, 1]$.

Time-oriented removal (R4): This operator is considered as a special case of operator R3, in which users that are serviced at approximately the same time are selected for removal. The difference in time between two users i and j ($\Delta t(i,j)$) is given by: $\Delta t(i,j) = |B_i^k - B_j^k| + |B_{i+n}^k - B_{j+n}^k|$.

Distance-oriented removal (R5): This operator is also based on operator R3. Here, the set of users to be removed are selected based on relatedness in terms of distance, where the difference in distance is measured as: $\Delta d(i,j) = d_{ij}^k + d_{i+n,j+n}^k$.

Basic greedy heuristic (P1): This heuristic is proposed by Ropke and Pisinger (2006a) for the PDPTW. It tries to insert the removed requests in a way that the additional cost is minimized, while satisfying all

constraints. Thus, the user that can be inserted at the lowest cost is inserted at its best location, i.e., the user i to be inserted and the vehicle k to insert it in are selected as follows: $(i,k) := \arg\min_{i\in L,k\in K} \Delta c_{i,k}$, where $\Delta c_{i,k}$ is the difference in cost before and after insertion of user i, in the cheapest position in vehicle k. This heuristic is repeated until all removed users in L are inserted.

Best position intra-route (P2): First a user i from list L is chosen randomly. Next, routes are considered one by one, and user i is assigned to the first route in which it can feasibly be inserted. The pickup and delivery nodes of i are then inserted at the best possible position in this route, while respecting precedence constraints. This procedure is repeated until no more users in list L.

Sorting time insertion (P3): The operation of (P3) is similar to (P2), but here all users in L are first sorted based on the start of the time window at their pickup node (e_i).

Best position inter-route (P4): The operation of (P4) is similar to (P2) but rather than checking only one particular vehicle as in (P2), we check all routes forming the solution in (P4).

6. Evaluation function

During the search in each method developed in this work, a new solution is generated, and it must be evaluated to check the feasibility of this solution before it can become the best global solution. We evaluate the solution by the following evaluation function based on Cordeau and Laporte (2003):

$$f(x) = c(x) + \sum_{e=0}^{3} \alpha q_e(x) + \beta d(x) + \gamma w(x) + \tau a(x)$$

Each new solution x is evaluated by the routing cost $c(x)$ plus a penalty for load violations $q_e(x) = \sum_{i=1}^{2n}(Q_i^e - Q^e)^+$, duration violations $d(x) = \sum_{k=1}^{K}(B_{2n+1}^k - B_0^k - T_{max})^+$, time windows violations $w(x) = \sum_{i=1}^{2n}(B_i - l_i)^+$ and ride time violations $a(x) = \sum_{i=1}^{n}(L_i - L_{max})^+$. Note that these terms are applied only for all $i \in N$ where $s^+ = \max\{0, s\}$. For the corresponding notation see Section 2. The associated penalty parameters α, β, γ and τ are dynamically adjusted throughout the search. For more details, the readers are referred to Parragh et al. (2010, 2012), and Cordeau and Laporte (2003). Note that each new solution can only become the current new best solution x_{best}, if this solution is acceptable with the acceptance criteria in each method developed and $q_e(x) = d(x) = w(x) = a(x) = 0$, for e=0, 1, 2, 3.

To evaluate a route, we use an evaluation procedure proposed by Cordeau and Laporte (2003), which consists of eight steps, as described in the Algorithm 5. This evaluation scheme applies the concept of forward time slack F_i for a node $i \in N$, which was originally proposed by Savelsbergh (1992) for the VRP with time windows, adapted to the DARP:

$$F_i = \min_{i\le j\le y}\{\sum_{i\le p\le j} W_p + (\min\{l_j - B_j, L - P_j\})^+\}$$

Where W_p represents the waiting time at node p, y is the last node in the route, and P_j represents the ride time of the user whose destination $j \in \{n+1,\ldots,2n\}$, given that $j - n$ is visited before i on the route; P_j=0 for all other j. F_i represents the maximum amount of time, where the departure from node i can be deferred, without violating the time window and passenger maximum ride time constraints.

Algorithm 5: The eight-step evaluation scheme

1. Set departure time $D_0 := e_0$
2. Compute arrival time (A_i), waiting time (W_i), beginning service (B_i), departure time (D_i) and load vehicle

(Q_i^e) for each node i along the route
If some $B_i > l_i$, or $Q_i^e > Q^{ek}$, Go to step 8
3. Compute F_0
4. Set $D_0 := e_0 + \min\{F_0, \sum_{0<p<y} W_p\}$
5. Update A_i, W_i, B_i and D_i for ech node on the route
6. Compute L_i for each request on the route
If all $L_i \leq L_{max}$ Go to step 8
7. For evry node j that is an origin
(a) Compute F_j
(b) Set $W_j := W_j + \min\{F_j, \sum_{j<p<y} W_p\}$; $B_j := A_j + W_j$; $D_j := B_j + s_j$
(c) Update A_i, W_i, B_i and D_i for each node that comes after j in the route
(d) Update L_i for each request i whose destination is after j
If all $L_i \leq L_{max}$ of requests whose destinations lie after j, Go to step 8
8. Compute changes in violation for load, duration, time windows and ride time constraints

7. Computational experimentation

In this section, we present the results obtained by our methods. All algorithms are implemented in C on a computer with Intel inside 4 GHz and 4 GB of RAM, operating Windows 8 with 64 bits. To test our methods, we generated new data sets as described in the following subsection. Additionally, we validated our algorithms by running them on existing benchmark instances for a simplified version of our problem, the MD-DARP.

7.1. Data and experimental setting

Our test data is divided into small, medium and large instances. The small instances are based on the benchmark instances generated by Parragh (2011) for the HDARP. These instances are divided into three sets (U, E, I), which are in turn modifications of instances created by Cordeau (2006) for the standard DARP, where heterogeneous vehicles and users are introduced. They contain 2–4 vehicles and 16–48 requests. On the other hand, our medium instances are based on Braekers et al. (2014a), which were generated for the HDARP and MD-HDARP. These contain 5-8 vehicles and 40-96 requests. For both the small and medium instances, the user time window is taken as 15 min, the maximum user ride time L_{max}=30 minutes, and the service time s_i = 3 minutes. Up to two vehicle types are considered, with four types of resources: 1) staff seats, 2) patient seats, 3) stretcher space and 4) wheelchair space.

To generate the instances, Parragh (2011) considered the probabilities of patients' requests for facilities and for companions as shown in Table 2.

Table 2
Probabilities used to generate instances by Parragh (2011)

Instance set	Patient request probabilities			Probability for companion (%)	Vehicle fleet
	% 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)

Vehicles of type T0 have three patient seats only. Vehicles of type T1 provide one staff seat, six patient seats, no stretchers, and one place for wheelchair, while vehicles of type T2 have two staff seats, one patient seat, one stretcher, and one wheelchair space. As shown in Table 2, data set U, assumes homogeneous users and vehicles of types T0, while data set E assumes heterogeneous users with

homogeneous vehicles of type T2. On the other hand, data set I has heterogeneous users and uses T1 and T2 vehicles.

To create large instances for our problem, we considered the same method of introducing heterogeneity as in Parragh (2011), explained in Table 2, and applied it to the 20 benchmark instances of Cordeau and Laporte (2003) for the DARP, i.e., we denoted the original homogeneous data set as set *U*, while the two additional heterogeneous data sets *E* and *I* have been generated by modifying the instances as done in Parragh (2011). The size of these instances of the problem is ranging from 24 to 144 requests. The number of vehicles used to serve the transport demands varies from 3 to 13 vehicles. These instances represent problems with a single depot. The distance between any two locations *i* and j is set to be the Euclidean distance between the coordinates of locations *i* and *j*. Service time s_i is 3 minutes for all users, and the transportation time t_{ij} is equal to the Euclidean distance between i and j (d_{ij}). The maximum duration of the working day (for each vehicle) T_{max} is 480 minutes and the maximum ride time L_{max} is 90 minutes. In the first 10 instances (R1a-R10a), the time windows range between 15 and 45 minutes. In the second 10 instances (R1b-R10b), the time windows are between 30 and 90 minutes.

All these instances have been adapted to the multi-depot case as in Braekers et al. (2014a), i.e., our new datasets contain four depots situated respectively at the following coordinates: [-5, -5], [-5, 5], [5, -5] and [5, 5]. In each instance the vehicles are distributed as follows: the first vehicle is assigned to the first depot, second vehicle to second depot,…, the fifth vehicle to the first depot, etc. In addition, time windows for lunch break and coffee break for all instances are introduced. The lunch time window $[EL, LL]$ is set to [240, 360] with a duration *TL* of 30 minutes. The coffee break time windows $[EC^r, LC^r]$ are [120,150] and [450,480] for $r = 1,2$, respectively. The coffee break duration T^r is 10 min ($\forall\, r = \{1, 2\}$). In all instances, a 60 minutes period is added to the maximum working day duration in order to respect the number of vehicles in the original data and to be able serve all users.

7.2. Parameter setting

The parameters of our algorithms were set based on recommendations from the literature, some basic experiments considering the tradeoff between solution quality and computation time, as well as our intuition.

Concerning the SA algorithm, the parameters suggested by Vincent et al. (2016) are primarily applied: cooling rate δ=0.99, n_{seq}=3,000 iterations and T_0=25. The original value T_0=100 proposed by Vincent et al. (2016) for the simple SA was divided by four in our algorithm. This latter value was chosen after testing several other initial temperature values (10, 25 and 50), and was found to best suit our hybrid BA-SA framework, since our SA is considered only secondary to the BA and needs to do less effort to find the best solution than a standalone SA. Similarly, for the DA algorithm, we have adopted the parameter values of Braekers et al. (2013), T_{max} is equal to 4, $\Delta T = T_{max}/2{,}500$ and i_{imp} equal to five times the number of vehicles used in the current best solution. In addition, for the number of iterations n_{DA}, the original value of Braekers et al. (2013) is equal to 50,000 iterations, but in our hybrid BA-DA, we have chosen n_{DA} equal to 5,000 iterations. This reduction of the number of iterations was intended to adapt the DA to our hybrid BA-DA framework.

For our ALNS, we adopt the parameters values that have been applied in Li et al (2016) for the static DARP using the ALNS approach: n_{ALNS}=25,000 iterations, p_d^0=0.10 for the removal operators, and 0.125 for the insertion operators, r_p=0.7, π_1=15, π_2=5, π_3=10. We note that in Li et al (2016), the parameters π_1, π_2 and π_3, are consistent with $\pi_1 \geq \pi_2 \geq \pi_3$. But in our case, and following some preliminary tuning experiments shown in Table 7 (and discussed later on), we have adopted $\pi_1 \geq \pi_3 \geq \pi_2$. Due to the complexity of studying the effect of the temperature T_{max} separately, we have decided to apply a large T_{max} value equal to 100, as in the standard SA algorithm, in order to initially allow accepting worse solutions and to permit our ALNS to escape local optima.

Moreover, various experiments were performed in order to ensure that the parameters for the hybrid Bees Algorithm BA-DA (BA-SA) are properly set and tuned. These parameters are N, be and es. Ten problem instances from each data set (U, E, I) were chosen for the purpose of tuning the parameters. They are selected such that the number of requests varies from small to large with different levels of heterogeneity. For the small and medium instances, it appeared that there is no significant difference shown during the tuning of parameters for a variety of combinations of *N*, *be* and *es*. This is mostly due to the already good performance of the standard DA and SA algorithms in reaching the best solutions for these specific instances. However, the results of the large instances were slightly more affected by the tuning of parameters. A summary of a sensitivity analysis experiment on the parameters *N*, *be* and *es* is displayed in Table 3 for a large instance (R6a) from data set U. "Best" ("Avg") indicates the best (average) solution value obtained by the BA-DA and BA-SA algorithms for each combination of the parameters; whereas, the average run time in minutes is denoted in the column CPU.

Table 3
Identification for the best parameters setting for the hybrid BA-DA (BA-SA)

N		10	20				30			
(be, es)		(5,3)	(15,10)	(15,5)	(10,5)	(10,3)	(20,10)	(20,5)	(15,10)	(15,5)
BA-SA	Best	813.15	**811.88**	**811.88**	**811.88**	812.05	**811.88**	**811.88**	**811.88**	**811.88**
	Avg	815.63	813.96	**814.23**	815.06	814.88	813.21	814.64	814.51	815.34
	CPU	18.75	30.05	**25.76**	27.98	32.75	35.87	31.42	39.53	43.86
BA-DA	Best	812.75	812.03	**811.88**	**811.88**	**811.88**	**811.88**	**811.88**	**811.88**	**811.88**
	Avg	813.12	814.23	**815.32**	815.75	814.63	813.29	814.32	815.65	815.82
	CPU	17.96	27.23	**25.65**	26.53	31.02	30.86	35.29	43.23	38.12

The solution quality depends not only on the influence of the population size *N*, but also on the impact of the values of *be* and *es*. It can be observed from Table 3 that by making the population size *N* equal to 20 or 30 (instead of 10), improved best solutions are obtained for the majority of combinations of the pair (*be*, *es*). Results slightly differ in the average solution quality and computation time. To obtain the best set of parameters, we use as a guide producing a high quality solution in a short period of CPU time. Therefore, the following parameter values were selected: N=20, be=15 and es=5.

Table 4 provides in brief the results of the parameter tuning of π_1, π_2 and π_3 for the ALNS algorithm. Based on these results, these values have been fixed at 15, 5 and 10, respectively. Note that the score for obtaining a non-improving solution which is accepted (π_3) is larger than that for obtaining an improved solution, which is not a new global best one (π_2), meaning that diversification is highly rewarded. The

usefulness of such an approach has been discussed before by e.g., Ropke and Pisinger (2006a) and Demir et al. (2012).

Table 4
Parameters tuning results summary

Inst.(Data)	Effect of π_1. π_2 and π_3 on the solution									
	(5, 10,15)	(1, 5,10)	(1, 5,5)	(5, 15,10)	(5, 10,1)	(5, 15,10)	(10, 5,1)	(15, 10,5)	(15, 5,10)	(10, 1,5)
a3-36 (U)	643.09	641.84	642.23	642.17	642.40	642.49	643.01	641.72	**641.41**	641.52
a7-56 (E)	816.14	815.43	815.45	815.09	814.12	816.44	815.77	815.58	**814.10**	815.01
a5-60 (I)	899.65	899.28	899.37	899.03	899.20	898.61	898.15	898.40	898.25	**897.68**
R2a (U)	312.59	311.79	312.67	312.54	312.41	311.26	312.41	311.00	**310.96**	311.56
R4a (E)	671.60	670.99	669.94	**668.60**	670.66	671.03	671.38	671.30	671.63	670.91
R3b (I)	572.94	571.95	571.99	571.95	571.90	571.73	571.44	572.93	**570.56**	571.02

According to Pisinger and Ropke (2007), and Ribeiro and Laporte (2012), the number of users to be removed in an ALNS iteration *u* does not have to be very large. In this context, the following strategy was taken into consideration: *u* is selected randomly in the interval [5, 10] if the number of users in the instance is less than 50 users; otherwise, *u* is chosen in randomly in the interval [5, 20].

Tables 5 and 6 indicate the frequency of application of each operator in the ALNS algorithm as a percentage of 25,000 iterations, for one instance of each type of data set (i.e., U, E and I). The total time performed by each operator is indicated in the parentheses. Table 5 demonstrates that all operators obtain almost identical frequency of use. The sum is larger than 100% due to the use of two operators in the same iteration in most iterations of our algorithm.

Table 5
Frequency of use as a percentage of 25,000 iterations and the computation time needed by each removal operator

Inst. (Data)	Removal operators				
	R1	R2	R3	R4	R5
a3-36 (U)	28.65 (1.19)	31.31 (0.95)	25.7 (0.93)	28.31 (1.04)	25.31 (1.44)
a7-56 (E)	30.31 (1.33)	26.97 (1.28)	27.02 (1.17)	23.97 (1.21)	25.34 (1.64)
a5-60 (I)	21.01 (1.12)	25.02 (1.04)	23.46 (1.11)	25.94 (1.34)	27.97 (1.77)
Avg	***26.66 (1.21)***	***27.77 (1.09)***	***25.39 (1.07)***	***26.07 (1.20)***	***26.21 (1.62)***
R2a (U)	23.02 (0.89)	22.36 (0.83)	24.31 (1.33)	21.31 (1.23)	21.65 (1.95)
R4a (E)	24.97 (1.11)	26.54 (0.94)	24.31 (1.78)	21.64 (1.15)	17.85 (1.75)
R8b (I)	27.97 (1.95)	21.68 (1.13)	28.24 (2.25)	19.89 (1.77)	24.98 (2.14)
Avg	***25.32 (1.32)***	***23.53 (0.97)***	***25.62 (1.79)***	***20.95 (1.13)***	***21.49 (1.95)***

Table 6
Frequency of use as a percentage of 25,000 iterations and the computation time needed by each insertion operator

Inst. (Data)	Insertion operators			
	P1	P2	P3	P4
a3-36 (U)	25.34 (2.02)	24.02 (2.24)	26.71 (3.12)	23.93 (4.11)
a7-56 (E)	26.36 (6.64)	24.63 (3.12)	21.34 (3.65)	27.67 (9.42)
a5-60 (I)	27.65 (21.97)	25.65 (13.65)	20.35 (10.53)	26.35 (19.85)
Avg	***26.45 (10.21)***	***24.77 (6.34)***	***22.80 (5.77)***	***25.98 (11.13)***
R2a (U)	28.21 (11.34)	23.23 (9.24)	19.23 (10.31)	29.33 (15.93)
R4a (E)	26.28 (25.98)	24.34 (21.02)	20.01 (15.22)	29.37 (33.03)

R3b (I)	27.70 (31.11)	21.23 (24.96)	20.02 (14.01)	31.05 (41.38)
Avg	***27.40 (22.81)***	***22.93 (18.41)***	***19.75 (13.18)***	***29.92 (30.11)***

With respect to the insertion operators in Table 6, the results (and particularly the computed averages) demonstrate that the P1 and P4 operators are slightly more utilized than P2 and P3. Obviously, the time consumed by P1 and P4 compared to that of P2 and P3, is slightly higher.

Table 7 presents the number of times a new best solution was found by each operator respectively. Additionally, the number in parentheses denotes the number of times the current solution was improved.

Table 7
Number of global best solutions found and number of improved solutions found by each operator

Inst. (Data)	Removal operators					Insertion operators			
	R1	R2	R3	R4	R5	P1	P2	P3	P4
a3-36 (U)	4(105)	1(134)	2(98)	2(65)	1(32)	3(129)	2(68)	1(104)	4(133)
a7-56 (E)	7(103)	4(142)	2(102)	3(89)	2(79)	7(176)	0(87)	3(46)	8(206)
a5-60 (I)	7(128)	5(98)	1(63)	4(33)	3(42)	5(88)	2(66)	2(80)	11(130)
R2a (U)	5(165)	4(175)	2(122)	3(141)	4(150)	6(301)	0(129)	4(152)	8(172)
R4a (E)	4(332)	3(265)	3(241)	3(155)	3(221)	8(214)	2(212)	4(119)	6(669)
R3b (I)	6(217)	7(322)	3(195)	2(188)	2(168)	5(134)	3(134)	5(198)	7(624)

Based on the results from Tables 6 and 7, it is clear that the insertion operators P2 and P3 are scarcely applied due to their limited ability to find new best solutions. However, ALNS could still take advantage of these operators, as they may diversify the search. To sum up, the removal and insertion operators are effective in getting high quality solutions for the MD-MT-HDARP as shown in the results.

7.3. Computational results

This section presents an overview of the experimental results obtained by our algorithms when tested on both newly generated instances for the MD-MT-HDARP and the instances for the MD-HDARP of Braekers et al. (2014a). For each table, detailed results per instance are available on our website.

7.3.1. Results on the MD-MT-HDARP

To test the performance of our algorithms, we ran each algorithm five times on each instance. In all tables reported in this section, columns "Best" ("Avg") show the best (average) solution objective values. The column "%" following each of the "Best" ("Avg") columns indicates the percentage of deviation of the preceding column from the Best Solution value (BS) obtained by any of the three developed algorithms for a given instance, while CPU shows the average run time in minutes.

Tables 8 and 9 show average results of our algorithms when run on the small-medium and large instances. As previously explained, three different data sets (U, E and I) are tested.

Table 8
Comparison of our three algorithms on small and medium instances

Inst.	BS	ALNS					BA-DA					BA-SA				
		Best	%	Avg	%	CPU (min)	Best	%	Avg	%	CPU (min)	Best	%	Avg	%	CPU (min)
$\bar{U}$	690.38	690.59	0.02	691.82	0.16	2.90	690.55	0.02	691.23	0.09	6.09	690.64	0.03	691.50	0.12	7.28
$\bar{E}$	710.38	711.26	0.08	712.71	0.23	2.88	710.67	0.03	711.50	0.12	6.31	710.69	0.03	711.61	0.12	6.37
$\bar{I}$	697.84	697.98	0.01	699.04	0.13	4.12	698.01	0.02	698.75	0.10	9.04	698.17	0.03	699.04	0.13	9.67

$\overline{UEI}$	***699.53***	***699.94***	***0.04***	***701.19***	***0.17***	***3.30***	***699.74***	***0.02***	***700.49***	***0.10***	***7.15***	***699.83***	***0.03***	***700.72***	***0.12***	***7.77***

Table 8 (and results in the website) show that the results of all our algorithms are comparable to each other, and each algorithm can find the best solution in the majority of instances. The hybrid BA-DA can find the best solution in at least one of the five runs for 62 instances (out of 72) compared to 59 instances for the ALNS and the hybrid BA-SA. The three methods were able to find the same best solution in 46 instances.

As can be noticed from the last row of Table 8, the overall average deviation from the BKS for all instances in the five runs is 0.17%, 0.10% and 0.12% for our ALNS, BA-DA and BA-SA algorithms, respectively; whereas, the average of the best runs deviates from the BS by 0.04%, 0.02% and 0.03%, respectively. For the average processing time, we note that the hybrid approaches are comparable, while the ALNS is slightly faster, with 3.30 minutes for ALNS, 7.09 minutes for BA-DA and 7.79 minutes for BA-SA.

Table 9
Comparison of our three algorithms on large instances

Inst.	BS	ALNS					BA-DA					BA-SA				
		Best	%	Avg	%	CPU (min)	Best	%	Avg	%	CPU (min)	Best	%	Avg	%	CPU (min)
$\bar{U}$	525.37	527.28	0.25	531.07	0.96	14.68	526.23	0.14	528.14	0.50	18.70	525.52	0.03	527.91	0.47	20.54
$\bar{E}$	591.63	592.85	0.17	596.94	0.74	17.66	591.67	0.01	594.05	0.35	23.00	591.97	0.05	594.86	0.55	26.05
$\bar{I}$	571.41	572.33	0.12	577.39	0.92	19.39	571.51	0.02	573.99	0.43	25.20	571.63	0.05	574.56	0.57	25.57
$\overline{UEI}$	***562.80***	***564.15***	***0.18***	***568.47***	***0.88***	***17.24***	***563.14***	***0.06***	***565.39***	***0.43***	***22.30***	***563.04***	***0.04***	***565.78***	***0.53***	***24.05***

By exploring the results in Table 9, the ALNS algorithm can find the best solution for 43 instances compared to 48 instances for BA-DA, and 40 instances for BA-SA. In 19 cases, all our methods are capable of finding the same best solution. Taking the average values over five runs for each algorithm, the average gap (%) is equal to 0.88% for ALNS, 0.43% for the BA-DA and 0.53% for the BA-SA. On the other hand, the average of the best result over five runs deviates from the BKS by 0.18% for the ALNS algorithm, 0.06% for hybrid BA-DA algorithm and 0.04% for hybrid BA-SA algorithm.

The results show that, on average over all instances, the hybrid BA-DA and the hybrid BA-SA algorithms outperform the ALNS for both the average and best solutions, albeit with a small difference in terms of the number of best found solutions (as can be seen from the results reported in the website). The superior performance of the hybrid methods compared to the ALNS can be attributed to the population-based nature of these algorithms, which allows it to explore several solutions in parallel, as well as to the benefit of the hybridizing two metaheuristic, as opposed to improving just one solution using one metaheuristic, which is the case in the ALNS. This, however, comes at the expense of a slight increase in computation time, as expected in most population-based metaheuristics. In general, our three algorithms are efficient and provide high-quality solutions for all instances. The fact that the algorithms were not able to produce the same results in all instances can be explained by the complexity of problem, especially for large instances, and the stochastic behavior of the solution algorithms.

To assess the effectiveness of integrating the DA and SA algorithm within the Bees Algorithm, we compared our hybrid BA-DA and BA-SA algorithms with only DA and SA without hybridization and with a simple (non-hybrid) BA, as shown in Tables 10 and 11. The parameters used for our simple SA and

DA methods are the same as those suggested by Vincent et al. (2016) and Braekers et al. (2013) for their SA and DA methods, respectively, as previously explained in Section. 4.2. The parameters and Pseudocode of the traditional BA are presented in Table 14 and Algorithm 6 in the Appendix, respectively.

Table 10
Comparison of BA-DA (BA-SA) performance with standard DA (SA)

Instance	Data	DA					SA				
		Best	%	Avg	%	CPU (min)	Best	%	Avg	%	CPU (min)
Small-	Type-U	692.50	0.28	695.10	0.56	1.96	691.38	0.11	693.92	0.35	2.84
Medium	Type-E	711.82	0.16	714.63	0.44	2.13	712.43	0.24	714.82	0.45	3.24
	Type-I	699.83	0.26	702.74	0.57	3.74	699.74	0.22	702.78	0.54	4.44
Avg		***701.38***	***0.23***	***704.16***	***0.52***	***2.61***	***701.18***	***0.19***	***703.84***	***0.45***	***3.51***
Large	Type-U	528.76	0.48	532.39	0.80	12.76	529.04	0.67	533.40	1.04	13.52
	Type-E	596.71	0.85	602.58	1.44	15.45	597.46	0.93	602.52	1.29	17.33
	Type-I	577.75	1.09	581.99	1.39	19.47	575.06	0.60	580.57	1.05	16.09
Avg		***567.74***	***0.81***	***572.32***	***1.21***	***15.89***	***567.19***	***0.73***	***572.16***	***1.12***	***15.65***

Table 10 compares the best (average) result for all instances of each data set, using each algorithm. Columns "Best" ("Avg") report the best (average) solution values of our DA and SA without hybridization. Columns "%" presents the percentage of deviation of the best (average) solutions obtained by our DA and SA, compared to the hybrid BA-DA and hybrid BA-SA, respectively. On the other hand, in Table 11, the columns "%BA-DA" and "%BA-SA" show the percentage of deviation from the best solutions obtained by our BA compared to the best (average) solution found by our hybrid BA-DA and BA-SA algorithms, respectively.

According to the results in Table 10, our hybrid BAs clearly outperform the non-hybrid DA (SA) algorithms, both in terms of best and average solution quality. For the small and medium instances (average of three data sets), using hybrid BA-DA (BA-SA) improves the results of the DA (SA), but with a small percentage. In fact, the average gap with the best solution obtained with hybrid BA-DA (hybrid BA-SA) is just 0.23% (0.19%) for these instances. The average deviation for the DA and SA compared to the average results of five runs of the hybrid BA-DA and hybrid BA-SA are 0.52% and 0.45%, respectively. However, a considerable improvement can be achieved with the use of our hybridization strategies for large instances. In this regard, our proposed hybrid algorithms improve solutions with 0.81% compared to the DA and 0.73% compared to the SA, respectively. The average deviation from the average solutions compared to our hybrid algorithms for the total routing cost, based on the average calculated for five runs, was 1.21% for DA and 1.12% for SA. This in fact indicates that the hybrid algorithms are also more stable in terms of finding high quality solutions in most of the runs compared to the standalone DA and SA.

Table 11
Comparison of BA-DA and BA-SA performance with BA

Instance	Data	Standard BA						
		Best	% BA-DA	% BA-SA	Avg	% BA-DA	% BA-SA	CPU (min)
Small-	Type-U	696.99	0.93	0.92	701.16	1.44	1.40	6.37
Medium	Type-E	717.03	0.89	0.89	721.43	1.40	1.38	5.75
	Type-I	704.16	0.88	0.86	708.22	1.36	1.31	8.14
Avg		***706.06***	***0.90***	***0.89***	***710.27***	***1.40***	***1.36***	***6.75***
Large	Type-U	541.36	2.88	3.01	550.43	4.22	4.27	53.21
	Type-E	608.17	2.79	2.74	615.23	3.57	3.42	52.51
	Type-I	586.85	2.68	2.66	593.67	3.43	3.33	51.74
Avg		***578.79***	***2.78***	***2.80***	***586.44***	***3.74***	***3.67***	***52.49***

Similar to the conclusion obtained from Table 10, Table 11 shows that both BA-DA and BA-SA algorithms outperform the traditional BA method. For the small and medium instances, using the DA (SA) algorithm, in conjunction with the BA, improves the results of the stand-alone BA. In fact, the average gap with the best solution obtained with the BA method amounts to 0.90% (0.89%) in comparison to the hybrid BA-DA (hybrid BA-SA). The average deviation of five runs for the simple BA was 1.40% (1.36%) compared to the BA-DA (BA-SA) algorithms. For large instances, the integration of the DA and SA improves solutions with a percentage of 2.78% for the hybrid BA-DA and 2.80% for hybrid BA-SA. The average deviation of the basic BA, from the average value, calculated for five runs was 3.74 % (3.67%) in comparison with the hybrid BA-DA (BA-SA).

From Tables 10 and 11, we conclude that the improvement phase in our new hybrid Bees Algorithm is important for all instances.

7.3.2. Results on the MD-HDARP

In order to further assess and validate the performance of our algorithms, we applied them on the instances of Braekers et al. (2014a) for the MD-HDARP. For this purpose, we assumed that the lunch break and coffee break time windows are $[EL, LL]$= $[EC^r, LC^r]$ = [0, 0] ($r = \{1, 2\}$), the lunch duration and coffee break duration are set to $TC = TL$ = 0. This being done, the MD-MT-HDARP is transformed to the MD-HDARP. We use the same parameter settings as before.

Tables 12 and 13 show the results of our algorithms on the small-medium instances of Braekers et al. (2014a) for the MD-HDARP. The results are compared with those of the Deterministic Annealing (DA) algorithm of Braekers et al. (2014a), which is considered the current state-of-the-art algorithm for solving the MD-HDARP. Each instance is solved five times using each of the algorithms (including the DA algorithm) as done in Braekers et al. (2014a). Column "Best" ("Avg") report the best (average) solution values, and "CPU" indicates CPU time in minutes. We note that all results of the algorithms are compared with the optimal and/or lower bound solution found by the B&C algorithm of Braekers et al. (2014a). The detailed results of each algorithm are reported in the website.

It should be noted that since a different machine has been used to generate the results for our algorithms than that of the existing DA algorithm of Braekers et al. (2014a), we are not able to accurately compare the relative performance of our algorithms in terms of processing time compared to that of

Braekers et al. (2014a). In addition, the speed factor of the configuration material applied by Braekers et al. (2014a) cannot be estimated by using Dongarra (2014) table, due to lack of relevant information in Dongarra (2014) and in Linpack (2016). Thus, we report in Table 12 the computational time only for the record, and not for a direct comparison with the previously published results.

Table 12
Comparison of our three methods on the MD-HDARP instances of Braekers et al. (2014a) on the best solutions

Inst.	B&C[a]	DA[a]			ALNS			BA-DA			BA-SA		
	Lower bound	Best	%	CPU (min)	Best	%	CPU (min)	Best	%	CPU (min)	Best	%	CPU (min)
$\bar{U}$	604.18	605.15	0.09	0.44	604.82	0.06	2.58	604.29	0.01	5.26	604.62	0.04	6.42
$\bar{E}$	623.24	623. 68	0.04	0.48	623.94	0.06	2.55	623.30	0.00	5.43	623.51	0.02	5.72
$\bar{I}$	613.71	617.00	0.34	0.45	615.26	0.15	3.61	614.12	0.04	7.90	614.50	0.08	8.34
$\overline{UEI}$	***613.71***	***615.29***	***0.16***	***0.46***	***614.67***	***0.09***	***2.92***	***613.91***	***0.02***	***6.19***	***614.21***	***0.05***	***6.83***

[a] *Best known solutions provided from Braekers et al. (2014a) on 2.6 GHz Intel Core laptop with 4 GB RAM.*

Table 13
Comparison of our three methods on the MD-HDARP instances of Braekers et al. (2014a) on the average solutions

Inst.	B&C[a]	DA[a]		ALNS		BA-DA		BA-SA	
	Lower bound	Avg	%	Avg	%	Avg	%	Avg	%
$\bar{U}$	604.18	605.66	0.15	606.17	0.11	605.77	0.07	605.66	0.10
$\bar{E}$	623.24	624.30	0.11	624.43	0.14	624.68	0.07	624.34	0.09
$\bar{I}$	613.71	617.83	0.43	617.83	0.26	618.11	0.11	617.83	0.15
$\overline{UEI}$	***613.71***	***615.93***	***0.23***	***615.94***	***0.17***	***616.07***	***0.09***	***615.94***	***0.11***

[a] *Best known solutions provided from Braekers et al. (2014a) on 2.6 GHz Intel Core laptop with 4 GB RAM.*

The results of Tables 12 and 13 show clearly that our proposed hybrid methods outperform the current state-of-the-art algorithm on the MD-HDARP in terms of solution quality. This applies to both the average deviation and the best deviation. In addition, regarding the number of best solutions found over five runs (Tables 12 and 13 in the website), the BA-DA (BA-SA) reports 65 (63) best solutions among 72 instances, compared to 62 best solutions for the DA of Braekers et al. (2014a). The average deviation from the best result over five runs for our Hybrid BA-DA (BA-SA) is 0.02% (0.05%) compared to 0.16% for the DA of Braekers et al. (2014a).The average deviation from the average results of the optimal solutions found by B&C are 0.09% (0.11%) for the BA-DA (BA-SA) algorithms, compared to 0.23% for the DA.

When compared to the DA, our ALNS also gives reasonable results, albeit with slightly less number of best solutions (61). Nevertheless, our ALNS outperforms the DA in terms of both average and best deviations. Regarding the average deviation from the best result in five runs, the ALNS is slightly better than the DA with 0.09%, while in the average deviation from the average results for the best known solutions, our ALNS surpasses the DA with 0.17%. In terms of computation time, as we previously mentioned, the average time of 0.46 minutes for Braekers et al. (2014a) cannot be fairly compared to our algorithms' time consumption, since the speed factor of their configuration is unknown.

In conclusion, we believe that our methods are fairly effective in obtaining high quality solutions for the MD-DARP.

8. Conclusion

Heterogeneous Dial-A-Ride Problems (HDARPs) are vehicle routing problems that arise in the management of door-to-door transportation services, which are mostly offered to the elderly and the disabled. In this paper, we proposed a more realistic variant of the HDARP, in which multi-depots, multi-trips, and coffee/lunch breaks are considered. The rationale is that in practice some dial-a-ride service providers have many depots in the same country. In addition, legislation rules may oblige the drivers to take lunch, and if necessary pause for coffee during their working day.

In this paper, we propose three different metaheuristics to solve the Multi-Depot Multi-Trip Heterogeneous Dial-a-Ride Problem (MD-MT-HDARP), and we assess their capabilities in handling the problem effectively. The three methods proposed are two new hybrid Bees Algorithms: Hybrid Bees Algorithm with Deterministic Annealing (BA-DA), and Hybrid Bees Algorithm with Simulated Annealing (BA-SA), and an Adaptive Large Neighborhood Search (ALNS). Different types of data sets have been used to test the performance of the above three methods. The obtained results indicate that our algorithms provide high quality solutions on newly generated (small-medium and large) instances. The hybrid algorithms were also superior to standalone algorithms in this respect. This can be attributed to the benefit of integrating single-solution based algorithms (i.e., DA and SA) within a population-based metaheuristic (i.e., the BA), which has demonstrated its potential in solving difficult problems, in comparison to standalone single-solution based metaheuristics. The results show that our BA-DA and BA-SA can effectively deal with the problem, giving high quality solutions on almost all instances. On the other hand, our ALNS algorithm is just slightly less efficient than our hybrid algorithms. This could be due to that the ALNS probably needs more removal and insertion operators for even more diversification of the search, which could make its behavior more robust to be able to compete with our population-based hybrid methods. Moreover, when compared on the related MD-HDARP, our methods are competitive to the state-of-the-art algorithm of Braekers et al. (2014a) on this problem. In fact, our algorithms provide solutions of better quality than the competitive metaheuristic of Braekers et al. (2014a), which shows the potential of our hybrid methods in solving other problem variants besides the problem described in this research, i.e., the MD-MT-HDARP.

For future work, we plan to focus on another complex variant, a Multi-depot HDARP with synchronization constraints between pickups and deliveries. Other possible directions for future research include the introduction of even more sophisticated local search techniques in the Bees Algorithm, and to apply this algorithm in a real environment for the transport of handicapped persons.

Acknowledgements

This work is partly supported by the University of Sfax and by the Interuniversity Attraction Poles Programme initiated by the Belgian Science Policy Office (research project COMEX, Combinatorial Optimization: Metaheuristics & Exact Methods). The authors would like to thank Professor Jiuh-Biing Sheu and the three anonymous referees for their valuable comments.

References

Borndörfer, R., Grötschel, M., Klostermeier, F., Küttner, C., 1999. *Telebus Berlin: Vehicle scheduling in a dial-a-ride system* (pp. 391-422). Springer Berlin Heidelberg.

Braekers, K., Caris, A., Janssens, G. K., 2013. Integrated planning of loaded and empty container movements. *OR spectrum*, *35*(2), 457-478.

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

Braekers, K., Caris, A., Janssens, G. K., 2014b. Bi-objective optimization of drayage operations in the service area of intermodal terminals. *Transportation Research Part E: Logistics and Transportation Review*, *65*, 50-69.

Bräysy, O., Dullaert, W., Gendreau, M., 2004a. Evolutionary algorithms for the vehicle routing problem with time windows. *Journal of Heuristics*, *10*(6), 587-611.

Bräysy, O., Hasle, G., Dullaert, W., 2004b. A multi-start local search algorithm for the vehicle routing problem with time windows. *European Journal of Operational Research*, *159*(3), 586-605.

Bräysy, O., Dullaert, W., Hasle, G., Mester, D., Gendreau, M., 2008. An effective multirestart deterministic annealing metaheuristic for the fleet size and mix vehicle-routing problem with time windows. *Transportation Science*, *42*(3), 371-386.

Busetti, F., 2003. Simulated annealing overview. *JP Morgan, Italy*.

Caris, A., Janssens, G.K., 2010. A Deterministic annealing algorithm for the pre- and end-haulage of intermodal container terminals. *International Journal of Computer Aided Engineering and Technology, 2*(4), 340–355.

Carnes, T.A., Henderson, S.G., Shmoys, D.B., Ahghari, M., MacDonald, R.D., 2013. Mathematical programming guides air-ambulance routing at ornge. *Interfaces, 43*(3), 232–239.

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*, *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.

Demir, E., Bektas, T., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, *223*(2), 346–359.

Doerner, K., Salazar-Gonzalez, J., 2014. In: Toth, P. and Vigo, D. (Eds). Vehicle Routing: Problems, Methods, and Applications, Second Edition. MOSSIAM Series on Optimation, *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, http://www.netlib.org/ benchmark/performance.pdf.

Dueck, G., Scheuer, T., 1990. Threshold accepting: a general purpose optimization algorithm appearing superior to simulated annealing. *Journal of computational physics*, *90*(1), 161-175.

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

Ghilas, V., Demir, E., Van Woensel, T. 2016. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows and scheduled lines. *Computers & Operations Research*, *72*, 12-30.

Goel, A., 2010. Truck driver scheduling in the European Union. *Transportation Science*, *44*(4), 429-441.

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

Kirkpatrick, S., 1984. Optimization by simulated annealing: Quantitative studies. *Journal of Statistical Physics*, *34*(5-6), 975-986

Laporte, G., Gendreau, M., Potvin, J. Y., Semet, F., 2000. Classical and modern heuristics for the vehicle routing problem. *International transactions in operational research*, **7**(4-5), 285-300.

Li, B., Krushinsky, D., Van Woensel, T., Reijers, H. A., 2016. An adaptive large neighborhood search heuristic for the share-a-ride problem. *Computers & Operations Research*, *66*, 170-180.

Li, Y., Chen, H., Prins, C., 2015. Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. Forthcoming in *European Journal of Operational Research*.

Lim, A., Zhang, Z., Qin, H., 2016. Pickup and delivery service with manpower planning in hong kong public hospitals. Forthcoming in *Transportation Science*

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

Lin, S. W., Vincent, F. Y., 2015. A simulated annealing heuristic for the multiconstraint team orienteering problem with multiple time windows. *Applied Soft Computing*, *37*, 632-642.

Liu, M., Luo, Z., Lim, A., 2015. A branch-and-cut algorithm for a realistic dial-a-ride problem. *Transportation Research Part B: Methodological*, *81*, 267-288.

Maalouf, M., MacKenzie, C. A., Radakrishnan, S., Court, M., 2014. A new fuzzy logic approach to capacitated dynamic Dial-a-Ride problem. *Fuzzy Sets and Systems*, *255*, 30-40.

Masson, R., Lehuédé, F., Péton, O., 2013. An adaptive large neighborhood search for the pickup and delivery problem with transfers. *Transportation Science*, *47*(3), 344-355.

Melachrinoudis, E., Ilhan, A. B., Min, H., 2007. A dial-a-ride problem for client transportation in a health-care organization. *Computers & Operations Research*, *34*(3), 742-759.

Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., Teller, E., 1953. Equation of state calculations by fast computing machines. *The journal of chemical physics*, *21*(6), 1087-1092.

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

Nikolakopoulos, A., Sarimveis, H., 2007. A threshold accepting heuristic with intense local search for the solution of special instances of the traveling salesman problem. *European journal of operational research*, *177*(3), 1911-1929.

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. Part II: transportation between pickup and delivery locations. *Journal für Betriebswirtschaft*, *58*(2), 81–117.

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.

Pham, D. T., Ghanbarzadeh, A., Koc, E., Otri, S., Rahim, S., Zaidi, M., 2005. The bees algorithm. Technical note. *Manufacturing Engineering Centre, Cardiff University, UK*, 1-57.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. *Computers & Operations Research*, *34*(8), 2403–2435.

Pisinger, D., Ropke, S. 2010. Large neighborhood search. In *Handbook of metaheuristics* (pp. 399-419). Springer US.

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.

Ribeiro, G. M., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the cumulative capacitated vehicle routing problem. *Computers & Operations Research*, *39*(3), 728-735.

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.

Ropke, S., Pisinger, D., 2006a. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation Science* (4), 455–472.

Ropke, S., Pisinger, D., 2006b. A unified heuristic for a large class of vehicle routing problems with backhauls. *European Journal of Operational Research*, *171*(3),750–775.

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

Schrimpf, G., Schneider, J., Stamm-Wilbrandt, H., Dueck, G., 2000. Record breaking optimization results using the ruin and recreate principle. *Journal of Computational Physics*, *159*(2), 139-171.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In *Principles and Practice of Constraint Programming—CP98* (pp. 417-431). Springer Berlin Heidelberg.

Talbi, E. G., 2009. *Metaheuristics: from design to implementation* (Vol. 74). John Wiley & Sons.

Tarantilis, C. D., Kiranoudis, C. T., Vassiliadis, V. S., 2004. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research*, *152*(1), 148-158.

Teodorovic, D., Radivojevic, G., 2000. A fuzzy logic approach to dynamic dial-a-ride problem. *Fuzzy sets and systems*, *116*(1), 23-33.

Toth, P., Vigo, D., 1996. Fast local search algorithms for the handicapped persons transportation problem. In: Osman, I.H., Kelly, J.P. (Eds.), Meta-Heuristics: Theory and Applications. Kluwer, Boston, pp. 677–690.

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

Vincent, F. Y., Jewpanya, P., Redi, A. P., 2016. Open vehicle routing problem with cross-docking. *Computers & Industrial Engineering*, *94*, 6-17.

Wong, K.I., Bell, M.G.H., 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.

Xiao, Y., Konak, A., 2015. A simulating annealing algorithm to solve the green vehicle routing & scheduling problem with hierarchical objectives and weighted tardiness. *Applied Soft Computing*, *34*, 372-388.

Yuce, B., Packianather, M. S., Mastrocinque, E., Pham, D. T., Lambiase, A., 2013. Honey bees inspired optimization method: the bees algorithm. *Insects*, *4*(4), 646-662.

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

Appendix: Parameters and Pseudocode of the BA

The parameters used in our tradional BA are shown in Table 14.

Table 14
Parameter values of traditional Bees Algorithm

Parameters	Value
Population size (*N*)	30
be	15
es	3
n_{bees} for small (large) instances	20000 (50000)
n_{be}	10
n_{be-es}	5

The difference between our traditional BA and the hybrid BA presented in Section 4 is in step 3 and the stopping criteria of the algorithm. Instead of applying DA (SA) on each of these (be) solutions, we simply apply n_{be} iterations of local search (described in subsection 4.4). Also, instead of the algorithm returning the best solution after ten non-improvement iterations, the algorithm outputs the best solution after n_{bees} consecutive iterations. The proposed traditional BA is shown in Algorithm 6.

Algorithm 6: Pseudo-code of the traditional Bees Algorithm

Begin
Initial population: Generate a population of *N* solutions, using a set of construction heuristics;
Repeat
Step 1: Evaluate fitness of each solution in the population *N*
Step 2: Sort the solutions in *N* in ascending order according to fitness and select the first (*be*) solutions from *N*
***Step 3*:** Select the first *es* solutions from (*be*)
Step 4: **For** each *es* solution **Do**
Repeat
Apply any local search operator {I1, I2, I3 or I4} to the current (*es*) solution and memorize the new solution
Until n_{es} is reached
Select the best one from the memorized solutions and record it
End For
Step 5: **For** each (*be-es*) solution Do
Repeat
Apply any local search operator {I1, I2, I3 or I4} to the current (*be-es*) solution and memorize the new solution
Until n_{be-es} is reached
Select the best one from the memorized solutions and record it
End For
Step 6: Select the best one from the new (*be*) solutions
If the current best solution is better than the best one **Then**
Replace the current solution with this new one
End If
Step 7: Replace the (*be*) new solutions into the population *N*
Step 8: Generate (*N- be*) new solutions
Until the maximum number of iteration n_{bees} is reached
Output the best solution as a result
End.