A hybrid metaheuristic approach for the capacitated arc routing problem

Yuning Chen, Jin-Kao Hao, Fred Glover

A Hybrid Metaheuristic Approach for the Capacitated Arc Routing Problem

Yuning Chen [a], Jin-Kao Hao [a,b,*], Fred Glover [c]

[a] *LERIA, Université d'Angers, 2 Bd Lavoisier, 49045 Angers, Cedex 01, France*

[b] *Institut Universitaire de France, Paris, France*

[c] *OptTek Systems, Inc., 2241 17th Street Boulder, Colorado 80302, USA*

Abstract

The capacitated arc routing problem (CARP) is a difficult combinatorial optimization problem that has been intensively studied in the last decades. We present a hybrid metaheuristic approach (HMA) to solve this problem which incorporates an effective local refinement procedure, coupling a randomized tabu thresholding procedure with an infeasible descent procedure, into the memetic framework. Other distinguished features of HMA include a specially designed route-based crossover operator for solution recombination and a distance-and-quality based replacement criterion for pool updating. Extensive experimental studies show that HMA is highly scalable and is able to quickly identify either the best known results or improved best known results for almost all currently available CARP benchmark instances. In particular, it discovers an improved best known result for 15 benchmark instances (6 classical instances and 9 large-sized instances all with unknown optima). Furthermore, we analyze some key elements and properties of the HMA-CARP algorithm to better understand its behavior.

Keywords: Capacitated arc routing problem; Memetic search; Tabu thresholding.

1 Introduction

The capacitated arc routing problem (CARP) has been the subject of a large number of studies during the last decades due to its wide applicability in logis-

* Corresponding author.

Email addresses: yuning@info.univ-angers.fr (Yuning Chen), hao@info.univ-angers.fr (Jin-Kao Hao), glover@opttek.com (Fred Glover).

tics, such as household waste collection, product distribution, winter gritting and postal deliveries, among others [10]. The CARP model can be informally described as follows. We are given a graph with a set of vertices and edges, where each edge has a predefined traversal cost and where a subset of edges, which are required to be serviced by some vehicles, are additionally associated with a service cost and a demand. A fleet of identical vehicles with a limited capacity is based at the depot vertex. The objective of CARP is to find a set of vehicle routes with a minimum cost such that: 1) each required edge is serviced on one of the routes; 2) each route must start and end at the depot vertex; and 3) the total demand serviced on the route of a vehicle must not exceed the vehicle capacity.

From a theoretical point of view, CARP is known to be NP-hard [12], and hence is not expected to be solved by any exact algorithm in a polynomial time in the general case. The computational difficulty of solving CARP is also confirmed in practice. Indeed, the best existing exact algorithms are limited to moderate instances with only 140 vertices and 190 edges [3, 6, 7]. For these reasons, intensive research has been devoted to developing heuristic and metaheuristic methods. Representative heuristic methods include Augment-Merge [12], Path-Scanning [17], the route first-cluster second heuristic [33] and Ulusoys Heuristic [35]. Among the metaheuristic methods, neighborhood search approaches are popular, e.g., tabu search [4, 19, 24], variable neighborhood search [20, 30], guided local search [2], GRASP with evolutionary path relinking [36]. As another class of popular metaheuristics for tackling CARP, population-based algorithms generally achieve better performances, such as the memetic algorithm [22,34], the ant colony algorithm [32] and the cooperative co-evolution algorithm [26]. Among these methods, two population-based algorithms (MEANS [34] and Ant-CARP [32]) and one local search algorithm (GLS [2]) represent the state-of-the-art solution methods for the classical test instance set, while RDG-MEANS [26] is the current best performing algorithm for the large-scale CARP (LSCARP) instances. Finally, for a thorough and up-to-date discussion of arc routing problems, the reader is referred to the recent book [8] edited by Á. Corberán and G. Laporte and in particular, chapter 7 by C. Prins dedicated to heuristic approaches.

In this work, we investigate a new population-based algorithm under the memetic search framework [27]. Memetic algorithms (MAs) have been proved to be very effective for solving a large number of difficult combinatorial optimization problems [28, 29], including CARP [22, 34]. The success of a MA highly depends on a careful design of two key search components: the crossover operator and the local refinement procedure [18]. Based on our previous experiences on MAs applied to various combinatorial problems, we go one step further by innovating especially on these two key components (crossover and local refinement) with the goal of elaborating a powerful memetic algorithm able to surpass the current state-of-the-art CARP methods.

The main contributions of our work can be summarized as follows.

- From the algorithmic perspective, the proposed population-based hybrid metaheuristic approach (HMA) combines a powerful local refinement procedure to ensure an effective search intensification with a dedicated crossover operator specially designed for CARP to guarantee a valid search diversification. The local refinement procedure couples a randomized tabu thresholding procedure to locate high-quality feasible solutions, with an infeasible descent procedure to enable tunneling between feasible and infeasible regions. The dedicated crossover operator relies on route information that can be embodied in exchanges of parent solutions to create new promising solutions. Additionally, to maintain a healthy population diversity and to avoid premature convergence, HMA employs a quality-and-distance strategy to manage the pool of solutions using a dedicated distance measure.
- In terms of computational results, extensive experiments carried out on 8 sets of widely used benchmarks show the competitiveness of the proposed method compared to the state-of-the-art CARP algorithms in solution quality and computational efficiency. For the 7 sets of 181 small-sized and medium-sized instances, HMA consistently matches or improves on all the best known results. In particular, HMA discovers a new best known result (new upper bound) for 6 well-studied instances. For the last set of 10 large-sized CARP benchmarks, HMA exhibits an even better performance. It easily dominates the state-of-the-art algorithms, including those specially designed for these CARP instances, by finding 9 new best known solutions, demonstrating the outstanding scalability of the proposed method.

The rest of the paper is organized as follows. Section 2 introduces preliminary notation and the solution representation. Sections 3 and 4 are dedicated to the description of the main HMA algorithm. Section 5 presents the computational results. Section 6 investigates some key elements of HMA, followed by the conclusions in Section 7.

2 Notation and solution representation

We are given a graph $G(V, E)$ with a set of vertices (V), a set of edges (E), a set of required edges ($E_R \subset E$) and a fleet of identical vehicles with a capacity of Q that is based at the depot vertex v_d ($v_d \in V$). Each edge $e = (i, j) \in E$ is represented by a pair of arcs $< i, j >$ and $< j, i >$. A required edge is said to be served if and only if one of its two arcs is included in one vehicle route of the routing plan. For the sake of simplicity, we use the term *task* to represent a required edge hereafter. Let n be the number of tasks, i.e., $n = |E_R|$. Each arc of a task, say u, is characterized by four elements: the head vertex ($head(u)$), the tail vertex ($tail(u)$), the traversal cost ($tc(u)$) and the demand ($q(u)$).

To represent a CARP solution, we assign to each task (i.e., a required edge) two IDs $(i, i+n)$ where i is an integer number in $[1, n]$, i.e., one ID for each arc of the task. We also define a dummy task with 0 as its task ID and both its head and tail vertices being the depot vertex v_d. This dummy task is to be inserted somewhere in the solution as a trip delimiter. Suppose a solution S involves m vehicle routes, S can then be encoded as an order list of (n+m+1) task IDs among which (m+1) are dummy tasks: $S = \{S(1), S(2), ..., S(n+m+1)\}$, where $S(i)$ denotes a task ID (an arc of the task or a dummy task) in the i^{th} position of S. S can be also written as a set of m routes (one route per vehicle): $S = \{0, R_1, 0, R_2, 0, ..., 0, R_m, 0\}$, where R_i denotes the i^{th} route composed of $|R_i|$ task IDs (arcs), i.e., $R_i = \{R_i(1), R_i(2), ..., R_i(|R_i|)\}$, with $R_i(j)$ being the task ID at the j^{th} position of R_i. Let $dist(u, v)$ denote the shortest path distance between the head vertex of arc u ($head(u)$) and the tail vertex of arc v ($tail(v)$), the total cost of a solution S can be calculated as:

$$f(S) = \sum_{i=1}^{n+m} (tc(S(i)) + dist(S(i), S(i+1))) \quad (1)$$

The total load $load(R_i)$ of a route R_i can be calculated as:

$$load(R_i) = \sum_{j=1}^{|R_i|} q(R_i(j)) \quad (2)$$

3 A hybrid metaheuristic algorithm for CARP

In this section, we describe the proposed hybrid metaheuristic algorithm (HMA) for CARP including the main procedure, the procedure for generating initial solutions, the specific route-based crossover as well as the quality-and-distance based pool maintenance procedure. The local refinement procedure of HMA is presented in Section 4.

3.1 Main scheme

Our HMA algorithm can be considered as a hybrid steady-state evolutionary algorithm which updates only one population solution at each generation of the evolution process [14]. Algorithm 1 shows the main scheme of the HMA algorithm. HMA starts with an initial population of solutions (Line 1 of Algo. 1) which are first generated by a random path scanning heuristic (Sect. 3.2) and further improved with the local refinement procedure (Sect. 4). Before entering the main loop, HMA initializes a counter array Cnt (Lines 3-4) which

is used to record the accumulated number of successful pool updates with the related threshold ratio value in a given set Sr (an external input).

At each generation, HMA randomly selects two parent solutions S^1 and S^2 from the population (Line 6), and performs a route-based crossover (RBX) operation (Line 7, see Sect. 3.3) to generate an offspring solution S^0. RBX basically replaces one route of one parent solution with one route from the other parent solution, and repairs, if needed, the resulting solution to ensure the feasibility of S^0. HMA then applies the local refinement procedure (Line 10, Sect. 4) to further improve S^0. The local refinement procedure involves two sub-procedures, namely a randomized tabu thresholding procedure (RTTP) and an infeasible descent procedure (IDP), which can be carried out in two possible orders: RTTP followed by IDP (RTTP→IDP) and IDP followed by RTTP (IDP→RTTP). The applied order is determined randomly before running the local refinement procedure (Line 8, see Sect 4.4). RTTP requires a threshold ratio which is probabilistically chosen among the values of a given set Sr according to the probability formula: $Pr(i) = Cnt(i)/\sum_{i=1}^{|Sr|} Cnt(i)$, where $Pr(i)$ denotes the probability of selecting the i^{th} value of Sr (Line 9).

Algorithm 1: Main Scheme of HMA for CARP

Data: P - a $CARP$ instance; $Psize$ - population size; St - a set of tabu timing values; W - the number of non-improving attractors visited; Sr - a set of threshold ratios; LB - the best known lower bound of P

Result: the best solution S^* found

```
// Population initialization, Section 3.2
1  POP ← Pool_Initialization(Psize) ;
2  S* ← Best(POP) ;        /* S* records the best solution found so far */
3  for i = 1 to |Sr| do
4      Cnt(i) = 1 ;                    /* Initialize the counter array Cnt */
   // Main search procedure
5  while stopping condition not reached do
6      Randomly select two solutions S^1 and S^2 from POP ;
7      S^0 ← Crossover(S^1, S^2) ;        /* Route-based crossover, Sect. 3.3 */
8      Od ← random_determine(RTTP → IDP, IDP → RTTP) ;   /* Determine
       the order of conducting RTTP and IDP, Sect. 4.4 */
9      k ← probabilistic_select(Cnt) ;          /* Select a ratio, Sect. 4.2  */
10     S^0 ← Local_refine(P, S^0, St, Sr(k), W, Od) ;  /* Improve S^0, Sect. 4  */
11     if f(S^0) = LB then
12         return S^0
13     (S*, POP) ← Pool_Updating(S^0, POP) ; /* Pool updating, Sect. 3.4 */
14     if pool updating is successful then
15         Cnt(k) ← Cnt(k) + 1 ;
16 return S*
```

If the improved solution reaches the lower bound LB, HMA terminates immediately and returns this solution (Line 11-12). Otherwise, HMA ends a generation by updating the recorded best solution and the population with the offspring solution S_c (Line 13, see Sect. 3.4). If S^0 is successfully inserted into the population, the counter in relation to the threshold ratio used in the current generation is incremented by one (Lines 14-15). HMA terminates when a stopping condition is reached, which is typically a lower bound cutoff or a maximum number of generations.

3.2 Population initialization

To generate one initial solution of the population, HMA uses a randomized path-scanning heuristic (RPSH) to construct a solution which is then further improved by the local refinement procedure described in Section 4. RPSH is adapted from the well-known path-scanning heuristic [17] by randomizing its five arc selecting rules. Specifically, RPSH builds one route at a time in a step-by-step way, each route starting at the depot vertex. At each step, RPSH identifies a set A of arcs (belonging to a set of unserved tasks) that are closest to the end of the current route and satisfy the vehicle capacity constraint. If A is empty, RPSH completes the current route by following a shortest deadheading path to the depot vertex and starts a new route. Otherwise RPSH randomly selects one arc from A and extends the current route with the selected arc. The selected arc as well as its inverse arc are marked served. This process continues until all tasks are served.

The solution constructed by RPSH is further improved by the local refinement procedure of Section 4. The improved solution is inserted to the population if it is unique in terms of solution cost relative to the existing solutions, or discarded otherwise. The population initialization procedure stops when the population is filled with $Psize$ (population size) different individuals or when a maximum of $3 * Psize$ trials is reached. The latter case helps to fill the population with $Psize$ distinct individuals. If ever only $k < Psize$ distinct solutions are obtained after $3 * Psize$ trials, we set the population size to k.

3.3 Route-based crossover operator for CARP

At each of its generations, HMA applies a crossover operator to create an offspring solution by recombining two parent solutions randomly selected from the population. It has been commonly recognized that the success of memetic algorithms relies greatly on the recombination operator which should be adapted to the problem at hand and be able to transfer meaningful properties (building

blocks) from parents to offspring [18]. This idea is closely related to the idea of using structured combinations and vocabulary building [16].

By considering that the solution of CARP is composed of a set of routes, it is a natural idea to manipulate routes of tasks rather than individual tasks. In this regard, the route-based crossover (RBX) operator used for the vehicle routing problem (VRP) [31] seems attractive for CARP. However, given that CARP is quite different from the VRP, RBX must be properly adapted in our context within HMA. Given two parent solutions $S^1 = \{R_1^1, R_2^1, ..., R_{m_1}^1\}$ with m_1 routes and $S^2 = \{R_1^2, R_2^2, ..., R_{m_2}^2\}$ with m_2 routes, our RBX crossover basically copies S^1 to an offspring solution S^0 and replaces a route of S^0 with a route from S^2, and then repairs S^0 to establish feasibility if needed. The RBX crossover procedure consists of three main steps:

- **Step 1:** Copy S^1 to an offspring solution $S^0 = \{R_1^0, R_2^0, ..., R_{m_1}^0\}$ and replace a route of S^0 with a route from S^2. Generate two random integer values a ($a \in [1, m_1]$) and b ($b \in [1, m_2]$); Replace the route R_a^0 of solution S^0 with the route R_b^2 of solution S^2, and collect the tasks that are not served in S^0 to a set UT;
- **Step 2:** Remove duplicated tasks by the following rule. Let $S^0(p_i)$ be the task in position p_i, and let $p_i - 1$ be the position before p_i and $p_i + 1$ be the position after p_i. Also let $dist(u, v)$ denote the shortest path distance between vertex $head(u)$ and vertex $tail(v)$. Given a task t_0 which appears twice respectively in position p_1 and p_2, RBX removes the appearance with the largest value of $s(p_i)$ ($i \in \{1, 2\}$), where $s(p_i) = dist(S^0(p_i-1), S^0(p_i)) + dist(S^0(p_i), S^0(p_i + 1)) - dist(S^0(p_i - 1), S^0(p_i + 1))$.
- **Step 3:** Insert the unserved tasks of UT in S^0. Before task insertion, RBX sorts the tasks in set UT in random order. Then for each task t in UT, RBX scans all possible positions of S^0 to insert t. If a position is able to accommodate t while respecting the vehicle capacity, RBX further calculates the saving (change of the total cost) with t inserted. The two arcs of t are both considered for insertion, and the minimum saving is recorded. RBX finally inserts the task to a position which causes the overall least augmentation of the total cost while maintaining the solution feasibility. Ties are broken randomly. This process is repeated until UT becomes empty.

Our proposed RBX operator not only introduces a new route (taken from S^2) into S^0 , but also modifies other existing routes due to deletion of duplicated tasks and insertion of unserved tasks. Clearly, RBX could lead to an offspring solution which is structurally different from its parent solutions. This is a desirable feature which promotes the overall diversity of HMA. Moreover, the quality of the offspring is not much deteriorated due to the use of greedy heuristics in Steps 2 and 3. As such, when the offspring is used as a seeding solution of local refinement, it helps the search to move into a new promising region. RBX can be realized in $O(n^2)$, where n is the number of tasks.

3.4 Population management

In population-based algorithms, one important goal aims to avoid premature convergence of the population. This can be achieved by adopting a carefully designed strategy for population management. In HMA, we use a quality-and-distance strategy (QNDS) for this purpose. QNDS takes into account not only the solution quality, but also the diversity that the solution contributes to the whole population by resorting to a solution distance measure.

We propose in this work to adapt for the first time the Hamming distance in the context of CARP and use it as our distance measure. Any pair of consecutive tasks $(S(i), S(i+1))$ of a solution S is linked by a shortest path (a path with minimum deadheading cost) between $head(S(i))$ and $tail(S(i+1))$, called *deadheading-link* hereafter. Thus, solution S has $n+m$ deadheading-links where n is the number of tasks and m is the number of routes. Let $V_R \subset V$ be a set of vertices that belong to the required edges, let $V_R^{'} = V_R \cup \{v_d\}$ be a set containing both the vertices of V_R and the depot vertex v_d, let $\Pi = \{(u,v)|u,v \in V_R^{'}\}$ be the set of all possible deadheading-links. Given two solutions S^i with m_i routes and S^j with m_j routes, their Hamming distance $D_{i,j}$ is defined as the number of different deadheading-links between S^i and S^j:

$$D_{i,j} = (n+m) - \sum_{(u,v)\in\Pi} x_{uv} \tag{3}$$

where $m = min\{m_i, m_j\}$,

$$x_{uv} = \begin{cases} 1, & \text{if } \ (u,v) \ is \ a \ deadheading-link \ of \ both \ S^i \ and \ S^j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

Given a population $POP = \{S^1, S^2, ..., S^{Psize}\}$ of size $Psize$ and the distance $D_{i,j}$ between any two individuals S^i and S^j $(i \neq j \in [1, Psize])$, the average distance between S^i and any other individual in POP is given by:

$$AD_{i,POP} = (\sum_{S^j \in POP, j \neq i} D_{i,j})/(Psize - 1) \tag{5}$$

QNDS evaluates each solution in the population using the following quality-and-distance fitness (QDF for short) function:

$$QDF(S^i) = \alpha * OR(f(S^i)) + (1-\alpha) * DR(AD_{i,POP}) \tag{6}$$

where $OR(f(S^i))$ and $DR(AD_{i,POP})$ represent respectively the rank of solution S^i with respect to its objective value and the average distance to the population (objective value is ranked in ascending order while average distance is ranked in descending order), and α is a parameter. We require the

value of α to be higher than 0.5 to ensure that the best individual in terms of objective value will never be removed from the population, which formalizes the elitism property of QNDS.

Given an offspring S^0 (which has undergone both crossover and local refinement), QNDS first inserts S^0 into POP, evaluates the QDF value of each individual and finally removes from POP the solution S^w with the largest QDF value.

4 Local refinement procedure of HMA

The local refinement procedure is another key component of our HMA algorithm and plays en essential role in enforcing intensification which ensures the high performance of HMA. Our local refinement procedure involves two sub-procedures, i.e., a randomized tabu thresholding procedure which explores only the feasible region, and an infeasible descent procedure which visits both feasible and infeasible regions. Both sub-procedures are based on a set of move operators which are explained below. The implementation of the two sub-procedures are also described.

4.1 Move operators

Our local refinement procedure employs six move operators, including five traditional small-step-size operators: inversion, single insertion, double insertion, swap, two-opt; as well as a large-step-size operator called merge-split recently proposed in [34]. These operators are briefly described as follows.

Let u and v be a pair of tasks in the current solution S, tasks x and y be respectively the successor of u and v, $rt(u)$ be the route including task u.

- Inversion (IV): replace the current arc of task u with its reverse arc in S;
- Single insertion (SI): displace task u after task v (also before task v if v is the first task of $rt(v)$); both arcs of u are considered when inserting u in the target position, and the one yielding the best solution is selected;
- Double insertion (DI): displace a sequence (u, x) after task v (also before task v if v is the first task of $rt(v)$); similar to SI, both directions are considered for each task and the resulting best move is chosen;
- Swap (SW): exchange task u and task v; similar to SI, both directions are considered for each task to be swapped and the resulting best move is chosen;
- Two-opt (TO): two cases exist for this move operator: 1) if $rt(u) = rt(v)$,

reverse the direction of the sequence (x, v); 2) if $rt(u) \neq rt(v)$, cut the link between (u, x) and (v, y) , and establish a link between (u, y) and (v, x);

- Merge-split (MS): this operator obtains an unordered list of tasks by merging multiple routes of the current solution, and sorts the unordered list with the path scanning heuristic [17]. It then optimally splits the ordered list into new routes using the Ulusoy's splitting procedure [35]. Each application of this operator results in five new solutions and the best one is chosen. Interested readers are referred to [34] for more details.

In the following two subsections, we explain how these operators are used in our two local refinement sub-procedures.

4.2 Randomized tabu thresholding procedure

The proposed randomized tabu thresholding procedure (RTTP) follows the general principle of the Tabu Thresholding (TT) method whose basis was first proposed in [13]. A main ingredient of TT is the candidate list strategy (CLS) which is dedicated to reduce the number of moves to be considered in order to accelerate the neighborhood examination. CLS subdivides the possible moves of the current solution into subsets and executes one move for each subset rather than for the whole neighborhood. CLS, along with the elements of probabilistic tabu search, simulates the tabu mechanism with memory structure. RTTP is a randomized procedure in the sense that it explores multiple neighborhoods in a random order.

4.2.1 Outline of the randomized tabu thresholding procedure

The randomized tabu thresholding procedure basically alternates between a *Mixed phase* and an *Improving phase* where for both phases, five traditional move operators are employed: inversion, single insertion, double insertion, swap and two-opt. Algorithm 2 sketches the outline of the RTTP procedure for CARP. RTTP starts by initializing a set of global variables with an initial solution S_0 taken from an external input. RTTP then enters the main loop where *Mixed phase* and *Improving phase* alternate.

In the *Mixed phase*, for any move operator o and for a given task i, RTTP examines the candidate list MOVE_CL(i, S, o) in random order and accepts the first improving feasible move if any, or the best admissible feasible move otherwise. The admissible feasible move satisfies a quality threshold TV, i.e., $f(S') \leq TV$ where S' is the neighboring solution generated by the accepted move. TV is calculated as: $TV = (1 + r) * f_p$, where f_p is the current best local optimum objective value, and r is a threshold ratio. With this quality threshold, deteriorating solutions are allowed in order to diversify the search.

Solution cycling is prevented through the complete reshuffling of the order in which candidate lists are examined before each neighborhood examination. An iteration of the *Mixed phase* is based on the examination of the complete neighborhoods of all move operators. The *Mixed phase* is repeated for T iterations. T is called a tabu timing parameter, which is analogous to the tabu tenure when an explicit tabu list is used. T is randomly selected among the values of a given set St.

Algorithm 2: Outline of the RTTP for CARP

Data: P - a $CARP$ instance, St - a set of tabu timing values, r - threshold ratio, W - the number of non-improving attractors visited, S_0 - an initial solution;
Result: the best solution S^* found so far;

```
1  O ← {IV, SI, DI, SW, TO} ;                    /* O contains five move operators */
2  S* ← S0 ;                                     /* S* records the global best solution */
3  S ← S0 ;                                      /* S records the current solution */
4  fp ← f(S*); ;   /* fp records the best local optimum objective value */
5  w ← 0 ;   /* set counter for consecutive non-improving local optima */
6  while w < W do
7      T ← random_select(St) ;
       // Mixed Phase
8      for k ← 1 to T do
9          Randomly shuffle all operators in O ;
10         for each o ∈ O do
11             Randomly shuffle tasks of S in M ;
12             for each i ∈ M do
13                 (S, S*) ← Apply operator o to task i by searching
                   MOVE_CL(i, S, o) and accepting a move according to the quality
                   threshold;
       // Improving Phase
14     while Improving moves can be found do
15         Randomly shuffle all operators in O ;
16         for each o ∈ O do
17             Randomly shuffle tasks of S in M ;
18             for each i ∈ M do
19                 (S, S*) ← Apply operator o to task j by searching
                   MOVE_CL(i, S, o) and accepting the first met improving move;
20     if f(S) > fp then
21         fp ← f(S); w ← 0 ;
22     else
23         w ← w + 1 ;
```

In the *Improving phase*, RTTP always seeks an improving move among the feasible moves within each candidate list MOVE_CL(i, S, o). If no improving move is found in a given candidate list, RTTP skips to the next candidate list. This phase is iterated until no improving move can be found in any candidate list.

If the local optimum reached in the *Improving phase* has a better objective value than the recorded best objective value f_p, the algorithm updates f_p and

resumes a new round of $Mixed - Improving$ phases. RTTP terminates when f_p has not been updated for a consecutive W $Mixed - Improving$ phases.

4.2.2 Construct candidate list

When using the five traditional move operators, neighborhoods of the current solution can always be obtained by operating on two distinct tasks. For instance, *insertion* is to insert one task after or before another task; *swap* is to swap one task with another task; *two-opt* is to exchange the subsequent part of a task with that of another one. As such, given a move operator o, a natural choice for the subsets to be used in the candidate list strategy is to define one subset MOVE_SUBSET(i, S, o) for each task i. In order to speed up neighborhood examination, we further use an estimation criterion to discard moves from MOVE_SUBSET(i, S, o) that are unlikely to lead to a promising solution. This estimation criterion is based on a distance measure between two tasks t_1, t_2 which is defined as:

$$D_{task}(t_1, t_2) = (\sum_{a=1}^{2}\sum_{b=1}^{2} D(v_a(t_1), v_b(t_2)))/4 \tag{7}$$

where $D(v_a(t_1), v_b(t_2))$ is the traversing distance between t_1's ath end node $v_a(t_1)$ and t_2's bth end node $v_b(t_2)$. (This distance measure was first used in [26] to define the distance between two routes.) The candidate move list associated to task i (MOVE_CL(i, S, o)) is restricted to contain *Csize* most promising moves such that for each move which is associated with two tasks $(i,\ t)$, t is a member of i's *Csize* closest neighboring tasks according to the distance measure of formula 7.

4.3 Infeasible descent procedure

For a constrained optimization problem like CARP, it is known that allowing a controlled exploration of infeasible solutions may facilitate transitions between structurally different solutions and help discover high-quality solutions that are difficult to locate if the search is limited to the feasible region. This observation is highlighted by discoveries made with the strategic oscillation approach (see, e.g., [15]) which alternates between phases of infeasible descent and phases of improving feasible search. To further intensify the search, we employ in our local refinement procedure, as a complement to RTTP, an infeasible descent procedure (IDP) which allows visiting infeasible solutions. IDP is a best-improvement descent procedure based on three traditional move operators, i.e., single insertion, double insertion, swap, as well as a large-step-size merge-split operator that was recently proposed and proved to be effective for CARP [34]. We use the merge-split operator in the way as suggested in [34].

IDP basically involves two different stages. In the first stage, IDP examines the complete neighborhoods induced by the SI, DI and SW operators and chooses the best move to perform if it improves the current solution. When no improvement can be found in the first stage, IDP switches to the second stage where it examines the neighboring solutions generated by the MS operator. Since MS is computationally expensive, IDP restricts the examination to a maximum of 100 neighboring solutions which are randomly sampled from the C_m^2 possibilities where m is the number of routes. If $C_m^2 \leq 100$, all neighboring solutions will be examined. Still, the best improving move is performed until no improvement is reported in this stage. If any improvement is found in the second stage, IDP switches back to the first stage to explore the new local region and terminates the algorithm when this stage is finished; otherwise, IDP terminates at the end of the second stage. As in many previous CARP algorithms which allow intermediate infeasible solutions [2, 4, 19, 34], we evaluate the solution quality generated in the search process of IDP by adding a penalty item to the original cost:

$$\phi(S) = f(S) + \beta * EX(S) \tag{8}$$

where $EX(S)$ is the total excess demand of S and β is a self-adjusting penalty parameter. β is halved (doubled) if feasible (infeasible) solutions have been achieved for five consecutive iterations, and its initiating value is set to:

$$\beta = f(S)/(2 * Q) \tag{9}$$

where Q is the vehicle capacity. One notices that we don't consider the violation of S in Eq. 9. This is because we always ensure that IDP starts the search from a feasible solution.

4.4 Combination of RTTP and IDP

After presenting the implementation of RTTP and IDP, the order of combining them in the local refinement procedure remains an issue to be addressed. RTTP is the most important component of our HMA algorithm which compared to IDP, makes more contribution to the high performance of HMA, but also consumes more computing time (see the analysis in Sect. 6.1). IDP is a very simple descent procedure which, when used alone, is not expected to identify very high quality solutions (see the analysis in Sect. 6.1). However, the search ability of RTTP and IDP can be mutually strengthened when they are combined. Indeed, it is beneficial to put IDP either before or after RTTP. When IDP is placed before RTTP, the best feasible solution found by IDP can be considered as a good starting point for RTTP. This is because the performance of neighborhood search algorithms may highly depend on the initial solution and a high-quality initial solution could help discover

still better solutions. When IDP is put after RTTP, the property of tunneling through infeasible regions and the large-step-size MS operator of IDP may help to further improve the high quality solution provided by RTTP. For the above reasons, both orders (i.e., RTTP→IDP and IDP→RTTP) are allowed in our HMA algorithm. The order is randomly determined before the local refinement procedure is carried out.

5 Computational experiments

To evaluate the efficacy of the proposed HMA algorithm, we carry out extensive experiments on a large number of well-known CARP benchmark instances, and compare the results [1] with those of the state-of-the-art algorithms as well as the best known solutions ever reported in the literature.

HMA was coded in C++ and compiled by GNU g++ 4.1.2 with the '-O3' option. The experiments were conducted on a computer with an AMD Opteron 4184 processor (2.8GHz and 2GB RAM) running Ubuntu 12.04. When solving the DIMACS machine benchmarks [2] without compilation optimization flag, the run time on our machine is 0.40, 2.50 and 9.55 seconds respectively for instances r300.5, r400.5 and r500.5.

5.1 Experimental setup

Our HMA algorithm was evaluated on a total of 191 benchmark graphs with 7 to 255 vertices and 11 to 375 edges. These instances are very popular and widely used in the CARP literature. They cover both random instances and real-life applications, and are typically classified into eight sets:

- ***gdb***: 23 instances randomly generated by DeArmon [9], with 7-27 nodes and 11-55 required edges.
- ***val***: 34 instances derived from 10 randomly generated graphs proposed by Benavent *et al.* [1], with 25-50 nodes and 34-97 required edges.
- ***egl***: 24 instances proposed by Eglese [11], which originate from the data of a winter gritting application in Lancashire (UK), with 77-140 nodes and 98-190 edges that include 51-190 required edges.
- ***C***: 25 instances generated by Beullens *et al.* [2] based on the intercity road network in Flanders, with 32-97 nodes and 42-140 edges that include 32-107 required edges.

[1] Our best solution certificates are available at: http://www.info.univ-angers.fr/pub/hao/CARPResults.zip

[2] dfmax: ftp://dimacs.rutgers.edu/pub/dsj/clique/

- ***D***: 25 instances modified from the instances of set *C* by doubling the vehicle capacity for each instance.
- ***E***: 25 instances, also generated by Beullens *et al.* [2] based on the intercity road network in Flanders, with 26-97 nodes and 35-142 edges that include 28-107 required edges.
- ***F***: 25 instances modified from the instances of set *E* by doubling the vehicle capacity for each instance.
- ***EGL-G***: 10 large-sized CARP instances, which like the set *egl*, were also generated based on the road network of Lancashire (UK) [4], each having 255 nodes and 375 edges with 374 to 375 required edges.

Following the common practice in the literature, we compare the results produced by our HMA algorithm on these benchmarks to those of the following eight state-of-the-art algorithms:

1 A guided local search (GLS) algorithm proposed by Beullens *et al.* [2], who reported results on the instance set *gdb*, *val* and *C-F*.
2 A deterministic tabu search algorithm (TSA) proposed by Brando & Eglese [4], who reported results on all eight instance sets. Two sets of results ("TSA1" and "TSA2") were reported, and the one ("TSA2") yielding better performance will be considered for comparative study for all instance sets except for EGL-G where only results of "TSA1" were reported.
3 A variable neighbourhood search (VNS) algorithm proposed by Polacek *et al.* [30], who reported results on set *val* and *egl*. Two sets of results ("993 MHz" and "3.6 GHz") were reported, and the one ("3.6 GHz") yielding better performance will be considered for comparative study.
4 A memetic algorithm with extended neighbourhood search (MAENS) proposed by Tang *et al.* [34], who reported results on set *gdb*, *val*, *egl* and *C-F*.
5 An improved ant colony optimization based algorithm (Ant-CARP) proposed by Santos *et al.* [32], who reported results on set *gdb*, *val*, *egl* and *C-F*. Two sets of results ("Ant-CARP_6" and "Ant-CARP_12") were reported, and the one ("Ant-CARP_12") yielding overall better performance will be considered for comparative study. Hereafter, we use "Ant_12" to represent "Ant-CARP_12", and use its median results for comparison when we study the average performance of the reference algorithms since their average results are not available.
6 A GRASP with evolutionary path relinking (GRASP) proposed by Usberti *et al.* [36], who reported results on the set *gdb*, *val* and *egl*.
7 An iterated variable neighbourhood descent algorithm (ILS-RVND) proposed by Martinelli *et al.* [25]. We reference their reported results of set *EGL-G*.
8 A cooperative co-evolution algorithm with route distance grouping (RDG-MAENS) proposed by Mei *et al.* [26]. RDG-MAENS was specifically designed for large-sized CARP instances, and thus we reference their results

Table 1
Scaling factors for computers used in the reference algorithms. Our computer (AMD Opteron 4184) is used as the basis.

Algorithm	Reference	Processor type	Frequency (GHz)	Factor
HMA	-	AMD Opteron 4184	2.8	1.0
GLS	[2]	Pentium II	0.5	0.18
VNS	[30]	Pentium IV	3.6	1.29
TSA2	[4]	Pentium Mobile	1.4	0.50
MAENS	[34]	Intel Xeon E5335	2.0	0.71
Ant_12	[32]	Pentium III	1.0	0.36
GRASP	[36]	Intel Core 2	3.0	1.07
ILS-RVNS	[25]	Intel Core i5	3.2	1.14
RDG-MAENS	[26]	Intel Core i7-2600	3.4	1.21

on set *EGL-G*.

These reference algorithms were tested on different computers with a CPU frequency ranging from 500 MHz to 3.6 GHz. To make a relatively fair comparison of the runtime, all CPU times reported in the reference papers are scaled here into the equivalent AMD Opteron 4184 2.8 GHz run times. Like in previous CARP literature [25, 26, 32, 34], our time conversion is based on the assumption that the CPU speed is approximately linearly proportional to the CPU frequency. We provide in Table 1 the CPU type and its frequency of each reference algorithm, as well as its resulting scaling factors. This time conversion is only made for indicative purposes, since the computing time of each algorithm is not only influenced by the processor, but also by some inaccessible factors such as the operating systems, compilers and coding skills of the programmer. Nevertheless, we show in the following experiments, the outcomes provide interesting information about the performance of the proposed algorithm relative to the best performing algorithms.

5.2 Parameter tuning

The HMA algorithm relies on a set of correlated parameters. To achieve a reasonable tuning of the parameters, we adopt the Iterated F-race (IFR) method [5], which allows an automatic parameter configuration, using the IFR algorithm that is implemented and integrated in the irace package [23]. Table 2 summarizes the parameters of our HMA algorithm, along with the range of values that were determined by preliminary experiments. Among these parameters, four of them (Psize, α, W, Csize) need to be tuned and the other two parameters (threshold ratio r and tabu timing parameter T) are adaptively or randomly chosen among the values in the given sets (Sr and St) during the search process. We set the tuning budget to 1000 runs of HMA and each run is given 100 generations. We restrict the training set to contain 8 challenging instances taken from *val*, *egl*, *C*, *E* and *EGL-G* sets: val-10D, egl-e3-B, egl-s3-C, C11, E12, E15, EGL-G1-B, EGL-G2-B. The final choices of the parameter

Table 2
Parameter tuning results

Parameter	Description	Range	Final value
Psize	population size	[6,16]	10
α	parameter of QDF in pool updating	[0.51,0.90]	0.60
St	tabu timing parameter values in RTTP	[28,33]	-
W	maximum number of attractors in RTTP	[5,20]	10
Sr	threshold ratios in RTTP for classical sets	{0.003,0.004,0.005,0.006}	-
	for EGL-G set	{0.0001,0.0005,0.0010,0.0015}	
Csize	candidate list size	[6,20]	12

values are presented in Table 2 and they are used in all experiments in the following sections unless otherwise mentioned.

5.3 Comparative results on 7 classical instance sets

We first assess HMA on the 7 most commonly used instance sets (181 instances): gdb, val, egl, C, D, E, F. It is compared to 5 current state-of-the-art algorithms: GLS [2], TSA2 [4], VNS [30], Ant-CARP [32], GRASP [36], and MAENS [34]. To give a general picture of the performance of each compared algorithm, we summarize in Table 3, for each instance set and for each algorithm, the number of best results that match or improve on the best known results (#Best), the number of average results that match or improve on the best known results (#BestAvg), the average gap between the average results and the best known results in percentage (AvgGap, the gap is calculated as $(f_{avg}\text{-}f_{bk})\times 100/f_{bk}$ where f_{avg} is the average solution value obtained by the algorithm and f_{bk} is the best known solution value reported in the literature), and the average of the instance computing time in seconds (AvgTime). When we count #Best, we refer to the current best known results (BKRs) which are compiled from the "best results" reported in all previous CARP literature. These "best results" could be those obtained by a single algorithm with various parameter settings (e.g., [22, 30, 32]) or even with a specific setting tuned for each instance (e.g., TSA_{best} in [4]). Finally, to complement these summarized results, Appendix A (Tables A.1–A.7) reports, for each of the 181 CARP instance, the detailed results of our HMA algorithm as well as the average results of the reference algorithms. These tables permit a thorough assessment of all compared algorithms.

Note that some results were obtained from a single run of the algorithms (GLS and TSA) whereas other results came from multiple runs (VNS, GRASP, Ant-CARP, MAENS, HMA1 and HMA2). Clearly, #Best favors multiple-run results. To make a fair comparison, we refer to average statistics (#BestAvg, AvgGap, AvgTime) when we compare single-run results with multiple-run results.

For each instance, our HMA algorithm was run 30 times under two different

stop criteria: 500 generations and 2000 generations. To ease presentation, we denote HMA with 500 generations as HMA_1, and HMA with 2000 generations as HMA_2. Studying the outcomes of these two termination criteria affords insights into how HMA behaves when more computing time is available.

From Table 3, we can see that HMA_1 shows a remarkable performance on all 7 tested instance sets compared to the multiple-run reference algorithms. Indeed, it attains the largest number of best known results for all 7 data sets and the lowest average gap to the best known results for 6 out of 7 sets. Compared to Ant_12 and MAENS which, like HMA_1, are both population-based algorithms, HMA_1 clearly shows its dominance in terms of both best results and average results. For set D, HMA_1 is the only algorithm which is able to find all BKRs. Additionally, HMA_1 obtains improved best known results on three well-studied instances from set *egl*. By increasing the HMA_1 termination criterion of 500 generations to 2000 generations, HMA_2 achieves a still better performance, always obtaining equal or better results in terms of both #Best and AvgGap. In particular, for set egl, HMA_2 discovers 6 new BKRs and matches 6 more BKRs, leading to #Best = 23 which is significantly larger than those obtained by the reference algorithms. HMA_2 is able to achieve overall 180 current or new BKRs out of 181 instances with one standard parameter setting, while the previous BKRs are compiled from many previous articles, among which some were obtained with parameters specifically tuned for individual instance.

Now we turn to compare our HMA algorithm to the single-run reference algorithms. As mentioned before, we should look at average statistics when comparing multiple-run algorithms to single-run algorithms. According to two average indicators, namely #BestAvg and AvgGap, GLS is clearly the best performing single-run algorithm among all 6 reference algorithms (including single-run algorithms and multiple-run ones). Still, compared to GLS, our HMA algorithm remains competitive on 6 instance sets (i.e., gdb, val and C-F). Indeed, when the short time limit (500 generations) is applied, HMA_1 performs better in items of AvgGap (by achieving an equal or lower AvgGap for more instance sets: 5 vs. 3), but worse in terms of #BestAvg (by attaining an equal or higher #BestAvg for less instance sets: 2 vs. 6). On set D, both HMA_1 and HMA_2 are dominated by GLS in terms of both indicators. Finally, one observes that when given more computing time, HMA_2 is able to further improve its results.

To validate the above observations, we apply a Wilcoxon test with a significance factor of 0.05 for a pairwise comparison of the average performance between HMA_1 and TSA2, Ant_12 as well as MAENS, which are three approaches that have been tested on all 181 instances. The resulting p-values of 2.15E-10, 9.31E-10 and 2.20E-16 confirm that the results of HMA_1 are significantly better than those of these current best performing algorithms. This conclusion remains valid for HMA_2 since it always performs better than

Table 3
Comparative statistical results on the 7 classical instance sets: gdb, val, egl, C-F. The best result of each row is indicated in bold. The results of HMA_2 are starred if they improve on the results of HMA_1.

		GLS	TSA2	VNS	GRASP	Ant_12	MAENS	HMA_1	HMA_2
		1 run	1 run	10 runs	15 runs	10 runs	30 runs	30 runs	30 runs
gdb	#Best	**23**	21	-	**23**	22	**23**	**23**	**23**
	#BestAvg	**23**	21	-	19	21	22	**23**	**23**
	AvgGap	**0.00**	0.07	-	0.11	0.10	0.01	**0.00**	**0.00**
	AvgTime	**0.32**	1.23	-	5.47	1.21	4.47	1.19	1.19
val	#Best	30	29	**32**	30	29	31	**32**	**34***
	#BestAvg	**30**	29	26	0	26	0	29	**30**
	AvgGap	0.05	0.14	0.09	0.12	0.17	0.18	**0.03**	**0.02***
	AvgTime	14.64	10.09	56.70	65.85	**9.11**	48.35	11.23	26.71
egl	#Best	-	4	11	12	11	12	**15**	**23***
	#BestAvg	-	4	4	3	**9**	2	**9**	**14***
	AvgGap	-	0.75	0.56	0.50	0.58	0.59	**0.13**	**0.07***
	AvgTime	-	**145.68**	649.17	854.48	181.09	498.49	198.83	646.03
C	#Best	20	18	-	-	22	23	**24**	**25***
	#BestAvg	20	18	-	-	16	3	18	**23**
	AvgGap	0.12	0.14	-	-	0.52	0.98	**0.06**	**0.02***
	AvgTime	42.49	42.92	-	-	40.76	165.50	**36.88**	54.22
D	#Best	24	17	-	-	23	23	**25**	**25**
	#BestAvg	**24**	17	-	-	20	10	20	23
	AvgGap	**0.04**	0.66	-	-	0.34	0.79	0.17	0.10
	AvgTime	17.36	19.20	-	-	51.88	219.53	**14.65**	35.13
E	#Best	19	17	-	-	20	20	**23**	**25***
	#BestAvg	19	17	-	-	16	5	16	**20**
	AvgGap	0.20	0.39	-	-	0.83	1.44	**0.19**	**0.08***
	AvgTime	40.65	46.00	-	-	**40.54**	160.89	42.62	116.54
F	#Best	25	15	-	-	22	**25**	**25**	**25**
	#BestAvg	**25**	15	-	-	20	12	**25**	**25**
	AvgGap	**0.00**	0.90	-	-	0.77	1.01	**0.00**	**0.00**
	AvgTime	10.67	21.09	-	-	52.17	166.85	**8.44**	**8.44**

HMA_1.

When it comes to computational time ('AvgTime' in Table 3), our HMA algorithm also remains competitive. Recall that the indicated time for the reference algorithms are scaled according to our computer and the average time of a multiple-run algorithm can be compared to the time of a single-run algorithm. Table 3 shows that HMA_1 is in overall not slower than any of the reference algorithms. Compared to the fast GLS, TSA2 and Ant_12 algorithms, HMA_1 generally requires comparable computing time. Compared to the remaining reference algorithms (i.e., VNS, GRASP and MAENS), HMA_1 is clearly more efficient. By extending the stop condition to 2000 generations, HMA_2, which finds improved solutions, consumes more computing time than HMA_1 as expected.

Finally, the "best results" reported by previous CARP studies were often achieved by executing tests involving multiple parameter settings to show the extreme performance of the associated algorithms. Following this practice, we

report in Appendix A some new best known results discovered by our HMA algorithm with parameter settings other than the standard one given in Table 2. The form of HMA using these additional parameter settings, which we call HMA*, further attains two new BKRs (for S4-A, S4-B) and matches the BKR for S3-C, which finally makes HMA* consistently match or improve on *all* 181 BKRs.

5.4 Comparative results on the EGL-G set

To test the scalability of HMA, we carried out experiments on the EGL-G set containing 10 large scale CARP (LSCARP) instances. As stated in [26], solving LSCARP is much more challenging than solving small-sized or medium-sized instances since the solution space increases exponentially as the problem size increases. Compared to the classical instance sets which involve instances having at most 190 required edges, all instances in EGL-G have more than 347 required edges. Such a size, as was shown in previous studies [4, 25], is large enough to pose a scalability challenge to the existing CARP algorithms. For this reason, a dedicated algorithm called RDG-MAENS [26] has been proposed specifically for solving LSCARP instances. In this section, we evaluate the capacity of our HMA algorithm to solve these 10 LSCARP instances by comparing its performance to those of the current best performing CARP algorithms including RDG-MAENS.

As before, HMA was executed 30 runs to solve each instance under two termination criteria: 500 generations (HMA_1) and 2000 generations (HMA_2). We also report the results obtained by HMA with various other parameter settings (HMA*). Table 4 summarizes our results on the EGL-G set, along with those of the current best performing algorithms: TSA1, ILS, MAENS, RDG-MAENS. In [26], the authors report the results of RDG-MAENS for 6 parameter combinations of (g,α) where $g = 2$ and 3, $\alpha = 1$, 5 and 10. We include the results of the best version ($g = 2$, $\alpha = 10$) for our comparative study. Table 4 lists the average results of each algorithm, the solution time of HMA, the best lower bounds (LB), the best known results (BKR), and the best results of HMA. The last two rows show, for each algorithm, the average of the average gaps to the BKR ($AvgGap$), and the "scaled" average of the average solution time ($AvgTime$).

Table 4 discloses that HMA_1 dominates all reference algorithms. In terms of average results, HMA_1 is much better than any of the reference algorithms for all 10 instances. The very small $AvgGap$ value of -0.02% of HMA_1 indicates that HMA_1 is on average better than the previous BKRs, and compares favorably to the $AvgGap$ value of more than 0.89% for the other approaches. In terms of best results, HMA_1 discovers an improved solution relative to the

Table 4
Comparative results of our HMA algorithm with 4 state-of-the-art algorithms on the 10 instances of EGL-G set. The best average results and best results are in bold. The best result of HMA* is starred if it improves on the BKR.

INST	($\|N\|, \|E_R\|$)	Average results								Best results				
		TSA1	ILS	MAENS	RDG	HMA_1	Time	HMA_2	Time	LB	BKR	HMA_1	HMA_2	HMA*
		1 run	10 runs	30 runs	30 runs	30 runs		30 runs						
G1-A	(255,347)	1049708	1010937.40	1009302	1007619	992823.00	1221.45	**992300.33**	3209.16	976907	998777	992337	**992045**	992045*
G1-B	(255,347)	1140692	1137141.50	1128114	1122863	1117198.77	1354.57	**1114992.53**	3961.81	1093884	1111971	1114125	1113003	1112245
G1-C	(255,347)	1282270	1266576.80	1255709	1250174	1240200.13	1453.32	**1237543.93**	4288.60	1212151	1241762	1235062	**1233536**	1231335*
G1-D	(255,347)	1420126	1406929.00	1390034	1386120	1374555.27	1411.71	**1371488.53**	3768.35	1341918	1371443	1370331	**1365259**	1365259*
G1-E	(255,347)	1583133	1554220.20	1535511	1525629	1516790.63	1338.02	**1513731.10**	3496.35	1482176	1512584	1511684	**1506179**	1503871*
G2-A	(255,375)	1129229	1118363.00	1109376	1104944	1095027.80	1816.93	**1091999.37**	4566.27	1069536	1094912	1090595	**1088040**	1088040*
G2-B	(255,375)	1255907	1233720.50	1225361	1221429	1206313.77	1749.72	**1201944.50**	4972.88	1185221	1208326	1200817	**1199877**	1199272*
G2-C	(255,375)	1418145	1374479.70	1358398	1355548	1338918.87	1646.83	**1337615.83**	3913.69	1311339	1341519	1334130	**1332791**	1331646*
G2-D	(255,375)	1516103	1515119.30	1500415	1492063	1480175.70	1580.93	**1477007.10**	4464.64	1446680	1481181	1473099	**1471107**	1470059*
G2-E	(255,375)	1701681	1658378.10	1641260	1629002	1618924.97	1655.15	**1615848.83**	4407.06	1581459	1618899	1611364	**1606807**	1606079*
AvgGap		3.94	2.22	1.32	0.89	-0.02		-0.22		-	-	-	-	-
AvgTime		510.55	1163.71	2437.30	1633.86	1522.86		4104.88		-	-	-	-	-

BKRs for 9 out of 10 instances (90%). As the current best and highly specialized algorithm, the best version of the reference algorithm RDG-MAENS is outperformed by HMA_1. Moreover, the average computational time of HMA_1 is comparable to that of the reference algorithms. HMA_1 requires much less time to find substantially better results than MAENS. Compared to the best version of RDG-MAENS, HMA_1 is also on average faster (1522.86 vs. 1633.86 seconds). The fact that HMA_2 further significantly improves on HMA_1 demonstrates that HMA can reach better performance when more computational time is allowed. By testing several other parameter settings, HMA* further discovers 6 improved best known results.

A Wilcoxon test is finally applied to a pairwise comparison of the average performance between HMA_1 and each of the four reference methods, which always results in a p-value of 0.001953 (<0.05) for all tested pairs, indicating the superiority of our method relative to the compared approaches.

6 Analysis

In this section, we present additional experimental results to analyze the performance of each algorithmic component of the proposed HMA algorithm, in order to understand their contribution to the overall performance of HMA and how they should be combined in the proposed algorithm. These experiments were performed on the *egl* set which contains a number of challenging instances of medium size, and helps to better distinguish the performance of the algorithm variants to be considered.

Table 5
Analysis of the components of the local refinement procedure using a single solution approach.

	IDP	TTP	RTTP	I+R	RST	HMA_1
BestGap	4.63	0.69	0.65	0.55	0.14	0.01
AvgGap	9.30	1.80	1.79	1.80	0.29	0.13
AvgTime	0.07	0.98	0.98	0.99	213.02	198.83

6.1 Analysis of algorithmic components

To provide insight of the performance of the local refinement procedure and the crossover operator, we test in this experiment several algorithm variants based on a single solution rather than a population of solutions. The "IDP" version is a very simple algorithm having only solution initialization (random path scanning) and IDP as its two components. Similar to "IDP", the "RTTP" version includes only the solution initialization and the RTTP procedure. The "TTP" version is exactly the same as "RTTP" except that the move operators are used in a fixed order as they are presented in Section 4.1. The "I+R" version combines IDP and RTTP in random order. The "RST" version is a random restart algorithm that simply starts "I+R" for 500 times. We also include the results of HMA_1 (500 generations) for comparative purposes. For each algorithm variant, we report in Table 5 the average of the best gap to the BKR in percentage (BestGap), and the average of the average gap to the BKR in percentage (AvgGap), as well as the average computing time in seconds (AvgTime).

Table 5 shows that TTP performs much better than IDP. RTTP further improves on TTP which shows the effectiveness of the random use of move operators. The results of "I+R" indicates that though RTTP is a very crucial component, its performance can be still ameliorated by a random collaboration with IDP. The comparative results of RST and HMA_1 clearly shows the relevance of the crossover operator and the population-based framework.

6.2 Analysis under the population-based framework

Given that both the local refinement procedures and the crossover operator are effective, this part of analysis investigates how they should be combined to achieve the best performance. For this purpose, we propose another set of algorithm variants based on a population of solutions. The "XO+I" version is obtained by removing the RTTP procedure from HMA, "XO+R" version by removing IDP from HMA. "XO+I→R" version works all the same as HMA, except that the order of using IDP and RTTP is fixed to I→R. Similarly, "XO+R→I" uses the order R→I. Table 6 summarizes the results of these

Table 6
Analysis of different combinations of the crossover operator and the local refinement procedure using a population of solutions.

	XO+I	XO+R	XO+I→R	XO+R→I	HMA_1
BestGap	0.72	0.02	0.02	0.04	0.01
AvgGap	1.59	0.16	0.16	0.15	0.13
AvgTime	6.77	254.11	158.37	244.20	198.83

Table 7
Analysis of the quality-and-distance pool updating strategy.

	HMA_{PW1}	HMA_1	HMA_{PW2}	HMA_2
BestGap	0.02	0.01	0.00	-0.04
AvgGap	0.15	0.13	0.10	0.07
AvgTime	188.15	198.83	437.34	646.03

algorithm variants, along with those of HMA_1. Without surprise, using only IDP in the local refinement procedure shows a rather poor performance, while employing RTTP in the local refinement procedure leads to a much better performance. Compared to "XO+R", including an additional IDP in the local refinement procedure never leads to a definitely better performance if the order of using IDP and RTTP is fixed. However, if IDP and RTTP are used in a random order as in HMA, a better performance can be observed in terms of both BestGap and AvgGap.

6.3 Analysis on the pool updating strategy

The third part of the analysis investigates the effectiveness of our pool updating strategy, which uses the hamming distance to control the diversity of the population. We therefore compare the adopted strategy to a traditional "pool worst" strategy which simply replaces the worst solution in terms of fitness in the population, leading to an algorithm variant denoted as HMA_{PW}. HMA_{PW} was tested under two termination criteria: 500 generations (HMA_{PW1}) and 2000 generations (HMA_{PW2}), and the outcomes are compared to those of HMA_1 and HMA_2. The computational results are summarized in Table 7, from which we can clearly see that HMA_1 performs better than HMA_{PW1}, and the superiority of HMA_2 relative to HMA_{PW2} enlarges when more computing time is allowed. This experiment confirms the usefulness of the diversity control mechanism used in our HMA.

7 Conclusions

The capacitated arc routing problem (CARP) is of great practical interest and represents a significant computational challenge due to its NP-hardness. We developed a new hybrid metaheuristic approach (HMA) for effectively solving CARP, which employs a randomized tabu thresholding procedure (RTTP) coupled with an infeasible descent procedure to explore both feasible and infeasible regions. HMA relies on a specialized route-based crossover operator to generate diversified and promising new solutions. Thanks to its quality-and-distance based pool updating strategy, HMA prevents the search process from premature convergence.

The proposed approach demonstrates an excellent performance over the eight sets of 191 popular CARP benchmarks. Specifically, on the 7 sets of 181 classical instances, HMA with a standard parameter setting outperforms the current best performing algorithms, in terms of both solution quality and computational efficiency. HMA further improves its own performance when more computing time is available (to run 2000 generations), attaining the best known results for all 181 cases including 6 improved new best results. HMA also proves to be scalable to handle the last set of 10 large-sized instances, by obtaining 9 new best results, dominating the current state-of-the-art algorithms including the approaches which were specially designed for the large-sized CARP instances. We additionally conducted experiments to analyze the contribution of the two sub-procedures for local refinement, the relevance of the route-based crossover operator (and thus the population-based framework), the strategy for combining crossover with the local optimization procedure, as well as the quality-and-distance pool updating strategy.

Finally, we observe that the proposed method can be adapted to handle other CARP variants with slight modifications of the route-based crossover operator and of the local refinement procedure to accommodate additional constraints.

Acknowledgment

We are grateful to the anonymous referees for their valuable suggestions and comments which helped us to improve the paper. The work is partially supported by the PGMO (2014-0024H) project from the Jacques Hadamard Mathematical Foundation and the National Natural Science Foundation of China (Grants 61473301, 71201171, 71501179). Support for Yuning Chen from the China Scholarship Council is also acknowledged.

References

[1] Benavent E., Campos V., Corberan E., Mota E. The capacitated arc routing problem: lower bounds. Networks 1992; 22(4) 669-690.

[2] Beullens P., Muyldermans L., Cattrysse D., Van Oudheusden D. A guided local search heuristic for the capacitated arc routing problem. European Journal of Operational Research 2003; 147(3): 629-643.

[3] Baldacci R., Maniezzo V. Exact methods based on noderouting formulations for undirected arcrouting problems. Networks 2006; 47(1): 52-60.

[4] Brandão J., Eglese R. A deterministic tabu search algorithm for the capacitated arc routing problem. Computers & Operations Research 2008; 35(4): 1112-1126.

[5] Birattari M., Yuan Z., Balaprakash P., Stützle T. F-Race and iterated F-Race: An overview. Experimental methods for the analysis of optimization algorithms 2010; Berlin, Germany: Springer, pp. 311-336.

[6] Bartolini E., Cordeau J.F., Laporte G. Improved lower bounds and exact algorithm for the capacitated arc routing problem. Mathematical Programming 2013; 137(1-2): 409-452.

[7] Bode C., Irnich S. Cut-first branch-and-price-second for the capacitated arc-routing problem. Operations research 2012; 60(5): 1167-1182.

[8] Corberán Á., Laporte G. Arc routing: problems, methods, and applications. MOS-SIAM Series on Optimization, Vol. 20. SIAM, 2015.

[9] DeArmon J. S. A comparison of heuristics for the capacitated Chinese postman problems. Master thesis, University of Maryland, College Park, MD, 1981.

[10] Dror, M. Arc Routing: Theory, Solutions and Applications. Kluwer Academic Publishers 2000.

[11] Eglese R.W. Routing Winter Gritting Vehicles. Discrete Applied Mathematics 1994; 48(3): 231-244.

[12] Golden B. L. and Wong R. T. Capacitated arc routing problems. Networks 1981; 11(3): 305-315.

[13] Glover F. Tabu thresholding: Improved search by nonmonotonic trajectories. ORSA Journal on Computing 1995; 7(4): 426-442.

[14] Glover F., Kochenberger G. (Eds.). Handbook of Metaheuristics, Kluwer, Norwell, Massachusetts, USA. 2003.

[15] Glover F., Hao J.K. The case for strategic oscillation. Annals of Operations Research 2011; 183(1): 163-173.

[16] Glover F, Laguna M. Tabu search. Kluwer Academic Publishers 1997.

[17] Golden B. L., DeArmon J. S., Baker E. K. Computational experiments with algorithms for a class of routing problems. Computers & Operations Research 1983; 10(1): 47-59.

[18] Hao J.K. Memetic algorithms in discrete optimization. In Neri F., Cotta C., Moscato P.(Eds.) Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Chapter 6, pages 7394, 2012.

[19] Hertz A., Laporte G., Mittaz M. A tabu search heuristic for the capacitated arc routing problem. Operations research 2000; 48(1): 129-135.

[20] Hertz A., Mittaz M. A variable neighborhood descent algorithm for the undirected capacitated arc routing problem. Transportation science 2001; 35(4): 425-434.

[21] Kirkpatrick S., Vecchi M. P. Optimization by simmulated annealing. Science 1983; 220(4598), 671-680.

[22] Lacomme P., Prins C., Ramdane-Cherif W. Competitive memetic algorithms for arc routing problems. Annals of Operations Research 2004; 131(1-4): 159-185.

[23] López-Ibáñez M., Dubois-Lacoste J., Stützle T., Birattari M. The irace package, iterated race for automatic algorithm configuration. Technical Report TR/IRIDIA/2011-004, IRIDIA, Universit Libre de Bruxelles, Belgium.

[24] Mei Y., Tang K., Yao X. A global repair operator for capacitated arc routing problem. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 2009; 39(3): 723-734.

[25] Martinelli R., Poggi M., Subramanian A. Improved Bounds for Large Scale Capacitated Arc Routing Problem. Computers & Operations Research 2013; 40(8): 2145-2160.

[26] Mei Y., Li X., Yao X. Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. IEEE Transactions on Evolutionary Computation 2014; 18(3): 435-449.

[27] Moscato P. On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms. Caltech concurrent computation program, C3P Report, 826, 1989.

[28] Moscato P., Cotta C. A Gentle Introduction to Memetic Algorithms. In Glover F. and Kochenberger G. A. (Eds.), Handbook of Metaheuristics. Kluwer, Norwell, Massachusetts, USA, 2003.

[29] Neri F., Cotta C., Moscato P. (Eds.) Handbook of Memetic Algorithms. Studies in Computational Intelligence 379, Springer, 2011.

[30] Polacek M., Doerner K.F., Hartl R.F., Maniezzo V. A variable neighborhood search for the capacitated arc routing problem with intermediate facilities. Journal of Heuristics 2008, 14(5), 405-423.

[31] Potvin J. Y., Bengio S. The vehicle routing problem with time windows part II: genetic search. INFORMS journal on Computing 1996; 8(2): 165-172.

[32] Santos L., Coutinho-Rodrigues J., Current J. R. An improved ant colony optimization based algorithm for the capacitated arc routing problem. Transportation Research Part B: Methodological 2010; 44(2): 246-266.

[33] Stern H. I., Dror M. Routing electric meter readers. Computers & Operations Research 1979; 6(4): 209-223.

[34] Tang K., Mei Y., Yao X. Memetic algorithm with extended neighborhood search for capacitated arc routing problems. IEEE Transactions on Evolutionary Computation 2009; 13(5): 1151-1166.

[35] Ulusoy, G. The fleet size and mix problem for capacitated arc routing. European Journal of Operational Research 1985; 22(3): 329-337.

Table A.1
Detailed comparative results of HMA algorithm with five state-of-the-art algorithms on the 23 instances of Set gdb. The average results of all algorithms and the best results of HMA_1 (500 generations) are highlighted in bold if they match the BKRs.

INST	($\|N\|,\|E\|$)	Average (median) results							Best results	
		GLS	TSA2	Ant_12	GRASP	MAENS	HMA_1	Time	BKR	HMA_1
		1 run	1 run	10 runs	15 runs	30 runs	30 runs			
1	(12,22)	**316**	**316**	**316**	**316.00**	**316.00**	**316.00**	0.09	316	**316**
2	(12,26)	**339**	**339**	**339**	**339.00**	**339.00**	**339.00**	0.16	339	**339**
3	(12,22)	**275**	**275**	**275**	**275.00**	**275.00**	**275.00**	0.09	275	**275**
4	(11,19)	**287**	**287**	**287**	**287.00**	**287.00**	**287.00**	0.08	287	**287**
5	(13,26)	**377**	**377**	**377**	**377.00**	**377.00**	**377.00**	0.13	377	**377**
6	(12,22)	**298**	**298**	**298**	**298.00**	**298.00**	**298.00**	0.13	298	**298**
7	(12,22)	**325**	**325**	**325**	**325.00**	**325.00**	**325.00**	0.09	325	**325**
8	(27,46)	**348**	**348**	**348**	349.50	348.70	**348.00**	14.01	348	**348**
9	(27,51)	**303**	**303**	**303**	303.60	**303.00**	**303.00**	3.20	303	**303**
10	(12,25)	**275**	**275**	**275**	**275.00**	**275.00**	**275.00**	0.12	275	**275**
11	(22,45)	**395**	**395**	**395**	**395.00**	**395.00**	**395.00**	0.34	395	**395**
12	(13,23)	**458**	**458**	**458**	**458.00**	**458.00**	**458.00**	0.24	458	**458**
13	(10,28)	**536**	540	544	542.60	**536.00**	**536.00**	6.59	536	**536**
14	(7,21)	**100**	**100**	**100**	**100.00**	**100.00**	**100.00**	0.11	100	**100**
15	(7,21)	**58**	**58**	**58**	**58.00**	**58.00**	**58.00**	0.10	58	**58**
16	(8,28)	**127**	**127**	**127**	**127.00**	**127.00**	**127.00**	0.14	127	**127**
17	(8,28)	**91**	**91**	**91**	**91.00**	**91.00**	**91.00**	0.13	91	**91**
18	(9,36)	**164**	**164**	**164**	**164.00**	**164.00**	**164.00**	0.16	164	**164**
19	(8,11)	**55**	**55**	**55**	**55.00**	**55.00**	**55.00**	0.05	55	**55**
20	(11,22)	**121**	**121**	**121**	**121.00**	**121.00**	**121.00**	0.12	121	**121**
21	(11,33)	**156**	**156**	**156**	**156.00**	**156.00**	**156.00**	0.19	156	**156**
22	(11,44)	**200**	**200**	**200**	**200.00**	**200.00**	**200.00**	0.33	200	**200**
23	(11,55)	**233**	235	235	234.70	**233.00**	**233.00**	0.71	233	**233**
AvgGap		0.00	0.07	0.10	0.11	0.01	0.00		-	-
AvgTime (scaled)		0.32	1.23	1.21	5.47	4.47	1.19		-	-

[36] Usberti F.L., Paulo M.F., André L.M.F. GRASP with evolutionary path-relinking for the capacitated arc routing problem. Computers & Operations Research 2013; 40(12): 3206-3217.

A Appendix

This appendix shows the detailed results of our HMA algorithm on the 181 conventional CARP instances (Table A.1–A.7). We also include in Table A.3 the new best known results (BKRs) discovered by HMA on *egl* instances with various parameter settings (indicated by HMA^*) other than the standard setting given in Table 2.

Table A.2
Detailed comparative results of HMA algorithm with 6 state-of-the-art algorithms on the 34 instances of Set val. The average results of all algorithms and the best results of HMA_1 (500 generations) and HMA_2 (2000 generations) are highlighted in bold if they match the BKRs. The optimal BKRs are underlined.

INST	($\|N\|$,$\|E\|$)	Average (median) results										Best results		
		GLS	TSA2	VNS	Ant_12	GRASP	MAENS	HMA_1	Time	HMA_2	Time	BKR	HMA_1	HMA_2
		1 run	1 run	10 runs	10 runs	15 runs	30 runs	30 runs		30 runs				
1A	(24,39)	**173**	**173**	**173.00**	**173**	**173.00**	**173.00**	**173.00**	0.28	**173.00**	0.28	173	**173**	**173**
1B	(24,39)	**173**	**173**	**173.00**	**173**	**173.00**	**173.00**	**173.00**	0.33	**173.00**	0.33	173	**173**	**173**
1C	(24,39)	**245**	**245**	**245.00**	**245**	**245.00**	**245.00**	**245.00**	0.43	**245.00**	0.43	245	**245**	**245**
2A	(24,34)	**227**	**227**	**227.00**	**227**	**227.00**	**227.00**	**227.00**	0.16	**227.00**	0.16	227	**227**	**227**
2B	(24,34)	**259**	**259**	**259.00**	**259**	**259.00**	**259.00**	**259.00**	6.37	**259.00**	6.37	259	**259**	**259**
2C	(24,34)	**457**	**457**	**457.00**	**457**	457.30	457.20	457.17	10.18	**457.00**	12.90	457	**457**	**457**
3A	(24,35)	**81**	**81**	**81.00**	**81**	**81.00**	**81.00**	**81.00**	0.17	**81.00**	0.17	81	**81**	**81**
3B	(24,35)	**87**	**87**	**87.00**	**87**	**87.00**	**87.00**	**87.00**	0.25	**87.00**	0.25	87	**87**	**87**
3C	(24,35)	**138**	**138**	**138.00**	**138**	**138.00**	**138.00**	**138.00**	0.31	**138.00**	0.31	138	**138**	**138**
4A	(41,69)	**400**	**400**	**400.00**	**400**	**400.00**	**400.00**	**400.00**	0.43	**400.00**	0.43	400	**400**	**400**
4B	(41,69)	**412**	**412**	**412.00**	**412**	**412.00**	**412.00**	**412.00**	0.44	**412.00**	0.44	412	**412**	**412**
4C	(41,69)	**428**	**428**	**428.00**	**428**	430.30	431.10	**428.00**	1.22	**428.00**	1.22	428	**428**	**428**
4D	(41,69)	530	530	531.20	530	531.00	532.90	530.00	6.39	529.87	17.57	528	530	**528**
5A	(34,65)	**423**	**423**	**423.00**	**423**	**423.00**	**423.00**	**423.00**	0.49	**423.00**	0.49	423	**423**	**423**
5B	(34,65)	**446**	**446**	**446.00**	**446**	**446.00**	**446.00**	**446.00**	0.41	**446.00**	0.41	446	**446**	**446**
5C	(34,65)	**474**	**474**	**474.00**	**474**	**474.00**	**474.00**	**474.00**	0.48	**474.00**	0.48	474	**474**	**474**
5D	(34,65)	579	583	579.80	583	584.50	582.90	576.33	59.65	575.60	210.21	575	**575**	**575**
6A	(31,50)	**223**	**223**	**223.00**	**223**	**223.00**	**223.00**	**223.00**	0.25	**223.00**	0.25	223	**223**	**223**
6B	(31,50)	**233**	**233**	**233.00**	**233**	**233.00**	**233.00**	**233.00**	0.82	**233.00**	0.82	233	**233**	**233**
6C	(31,50)	**317**	**317**	**317.00**	**317**	**317.00**	**317.00**	**317.00**	0.54	**317.00**	0.54	317	**317**	**317**
7A	(40,66)	**279**	**279**	**279.00**	**279**	**279.00**	**279.00**	**279.00**	0.41	**279.00**	0.41	279	**279**	**279**
7B	(40,66)	**283**	**283**	**283.00**	**283**	**283.00**	**283.00**	**283.00**	0.37	**283.00**	0.37	283	**283**	**283**
7C	(40,66)	**334**	**334**	**334.00**	**334**	**334.00**	**334.00**	**334.00**	0.48	**334.00**	0.48	334	**334**	**334**
8A	(30,63)	**386**	**386**	**386.00**	**386**	**386.00**	**386.00**	**386.00**	0.39	**386.00**	0.39	386	**386**	**386**
8B	(30,63)	**395**	**395**	**395.00**	**395**	**395.00**	**395.00**	**395.00**	0.43	**395.00**	0.43	395	**395**	**395**
8C	(30,63)	**521**	529	522.00	527	526.50	525.90	**521.00**	58.00	**521.00**	58.00	521	**521**	**521**
9A	(50,92)	**323**	**323**	**323.00**	**323**	**323.00**	**323.00**	**323.00**	1.09	**323.00**	1.09	323	**323**	**323**
9B	(50,92)	**326**	**326**	**326.00**	**326**	**326.00**	**326.00**	**326.00**	0.83	**326.00**	0.83	326	**326**	**326**
9C	(50,92)	**332**	**332**	**332.00**	**332**	**332.00**	**332.00**	**332.00**	0.88	**332.00**	0.88	332	**332**	**332**
9D	(50,92)	391	391	390.80	391	392.10	391.00	390.03	112.24	389.50	391.30	389	**389**	**389**
10A	(50,97)	**428**	**428**	428.40	**428**	**428.00**	**428.00**	**428.00**	1.52	**428.00**	1.52	428	**428**	**428**
10B	(50,97)	**436**	**436**	436.60	**436**	**436.00**	**436.00**	**436.00**	2.45	**436.00**	2.45	436	**436**	**436**
10C	(50,97)	**446**	**446**	447.00	**446**	446.20	**446.00**	**446.00**	1.78	**446.00**	1.78	446	**446**	**446**
10D	(50,97)	526	530	526.90	528	530.60	533.60	526.37	111.37	526.00	194.26	525	526	**525**
AvgGap		0.05	0.14	0.09	0.12	0.17	0.18	0.03		0.02		-	-	-
AvgTime (scaled)		14.64	10.09	56.70	9.11	65.85	48.35	11.23		26.71		-	-	-

Table A.3
Detailed comparative results of HMA algorithm with five state-of-the-art algorithms on the 24 instances of Set egl. The average results of all algorithms and the best results of HMA_1 (500 generations) and HMA_2 (2000 generations) are highlighted in bold if they match the BKRs. The best results of HMA* are starred if they are new BKRs. The optimal BKRs are underlined.

INST	($\|N\|,\|E\|$)	Average (median) results									Best results				
		TSA2	VNS	Ant_12	GRASP	MAENS	HMA_1	Time	HMA_2	Time	LB	BKR	HMA_1	HMA_2	HMA*
		1 run	10 runs	10 runs	15 runs	30 runs	30 runs		30 runs						
E1-A	(77,98)	**3548**	**3548.00**	**3548**	**3548.00**	**3548.00**	**3548.00**	0.58	**3548.00**	0.58	3548	3548	**3548**	**3548**	3548
E1-B	(77,98)	4533	4522.20	4539	4508.60	4516.50	**4498.00**	9.62	**4498.00**	9.62	4498	4498	**4498**	**4498**	4498
E1-C	(77,98)	**5595**	5608.00	**5595**	5615.30	5601.60	**5595.00**	8.71	**5595.00**	8.71	5595	5595	**5595**	**5595**	5595
E2-A	(77,98)	**5018**	5023.80	**5018**	**5018.00**	**5018.00**	**5018.00**	24.08	**5018.00**	24.08	5018	5018	**5018**	**5018**	5018
E2-B	(77,98)	6343	6335.40	6344	6330.70	6341.40	6319.77	44.29	**6317.00**	104.93	6317	6317	**6317**	**6317**	6317
E2-C	(77,98)	8347	8355.90	**8335**	8335.80	8355.70	**8335.00**	28.12	**8335.00**	28.12	8335	8335	**8335**	**8335**	8335
E3-A	(77,98)	5902	**5898.00**	**5898**	**5898.00**	5898.80	**5898.00**	5.35	**5898.00**	5.35	5898	5898	**5898**	**5898**	5898
E3-B	(77,98)	7816	7806.40	7787	7787.30	7802.90	7777.00	76.19	7776.93	98.19	7744	7775	7777	**7775**	7775
E3-C	(77,98)	10309	10322.30	**10292**	10296.50	10321.90	10294.43	133.00	**10292.00**	177.41	10244	10292	**10292**	**10292**	10292
E4-A	(77,98)	6473	6459.40	6464	6461.10	6475.20	6461.17	82.61	6450.57	675.30	6408	6444	6446	**6444**	6444
E4-B	(77,98)	9063	9016.30	9047	9037.10	9023.00	8991.50	143.63	8987.00	454.17	8935	8961	8972	**8961**	8961
E4-C	(77,98)	11627	11750.10	11645	11670.00	11645.80	11582.40	203.43	11563.97	511.24	11512	11529	11545	**11529**	11529
S1-A	(140,190)	5072	**5018.00**	**5018**	5038.90	5039.80	**5018.00**	11.98	**5018.00**	11.98	5018	5018	**5018**	**5018**	5018
S1-B	(140,190)	**6388**	**6388.00**	**6388**	6388.40	6433.40	**6388.00**	36.29	**6388.00**	36.29	6388	6388	**6388**	**6388**	6388
S1-C	(140,190)	8535	8518.20	**8518**	8521.50	8518.30	**8518.00**	21.59	**8518.00**	21.59	8518	8518	**8518**	**8518**	8518
S2-A	(140,190)	10038	9997.90	9974	9980.50	9959.20	9907.23	390.84	9891.33	1887.59	9825	9884	9888	**9874**	9874*
S2-B	(140,190)	13178	13176.00	13283	13240.60	13231.60	13149.97	361.66	13153.70	597.74	13017	13099	13111	**13078**	13078*
S2-C	(140,190)	16505	16551.60	16558	16539.90	16509.80	16465.30	294.90	16439.67	1002.06	16425	16425	16430	**16425**	16425
S3-A	(140,190)	10451	10291.20	10306	10276.10	10312.70	10228.10	463.81	10217.83	1719.03	10165	10220	**10220**	**10201**	10201*
S3-B	(140,190)	13981	13829.20	13890	13860.70	13876.60	13739.10	466.39	13705.67	1397.05	13648	13682	13688	**13682**	13682
S3-C	(140,190)	17346	17327.90	17304	17277.70	17305.80	17248.60	345.12	17217.63	1338.38	17188	17188	17196	17189	17188
S4-A	(140,190)	12462	12440.40	12439	12406.50	12419.20	12269.73	603.98	12258.30	2035.96	12153	12268	**12254**	**12244**	12216*
S4-B	(140,190)	16490	16410.30	16502	16432.00	16441.20	16264.70	611.16	16243.70	1663.58	16113	16260	**16222**	**16216**	16201*
S4-C	(140,190)	20733	20731.50	20731	20660.50	20767.20	20545.53	404.61	20518.37	1695.86	20430	20481	**20479**	**20476**	20476*
AvgGap		0.75	0.56	0.58	0.50	0.59	0.13		0.07		-	-	-	-	-
AvgTime (scaled)		145.68	649.17	181.09	854.48	498.49	198.83		646.03		-	-	-	-	-

Table A.4
Detailed comparative results of HMA algorithm with 4 state-of-the-art algorithms on the 25 instances of Set C. The average results of all algorithms and the best results of HMA_1 (500 generations) and HMA_2 (2000 generations) are highlighted in bold if they match the BKRs. The optimal BKRs are underlined.

INST	($\|N\|$, $\|E\|$)	Average (median) results								Best results			
		GLS	TSA2	Ant_12	MAENS	HMA_1	Time	HMA_2	Time	LB	BKR	HMA_1	HMA_2
		1 run	1 run	10 runs	30 runs	30 runs		30 runs					
C1	(69,98)	**1660**	**1660**	1705	1707.00	1661.00	117.04	**1660.00**	160.51	1660	1660	**1660**	**1660**
C2	(48,66)	**1095**	**1095**	**1095**	1095.70	**1095.00**	9.10	**1095.00**	9.10	1095	1095	**1095**	**1095**
C3	(46,64)	**925**	**925**	**925**	927.80	930.33	30.94	926.33	163.67	925	925	**925**	**925**
C4	(60,84)	**1340**	**1340**	**1340**	1342.70	**1340.00**	5.20	**1340.00**	5.20	1340	1340	**1340**	**1340**
C5	(56,79)	2475	**2470**	2540	2522.30	2470.33	67.19	**2470.00**	84.00	2470	2470	**2470**	**2470**
C6	(38,55)	**895**	**895**	**895**	907.50	**895.00**	13.78	**895.00**	13.78	895	895	**895**	**895**
C7	(54,70)	**1795**	**1795**	**1795**	**1795.00**	**1795.00**	0.61	**1795.00**	0.61	1795	1795	**1795**	**1795**
C8	(66,88)	**1730**	**1730**	**1730**	1732.30	**1730.00**	14.20	**1730.00**	14.20	1730	1730	**1730**	**1730**
C9	(76,117)	1825	1830	1860	1852.80	**1820.00**	74.70	**1820.00**	74.70	1805	1820	**1820**	**1820**
C10	(60,82)	2290	**2270**	2305	2317.80	**2270.00**	24.17	**2270.00**	24.17	2270	2270	**2270**	**2270**
C11	(83,118)	1815	1815	1820	1853.70	1815.00	37.19	1813.67	121.19	1790	1805	1815	**1805**
C12	(62,88)	**1610**	**1610**	**1610**	**1610.00**	**1610.00**	1.30	**1610.00**	1.30	1605	1610	**1610**	**1610**
C13	(40,60)	**1110**	**1110**	**1110**	1122.00	**1110.00**	8.20	**1110.00**	8.20	1110	1110	**1110**	**1110**
C14	(58,79)	**1680**	**1680**	**1680**	1687.30	**1680.00**	15.88	**1680.00**	15.88	1680	1680	**1680**	**1680**
C15	(97,140)	**1860**	1865	1880	1896.50	**1860.00**	82.70	**1860.00**	82.70	1840	1860	**1860**	**1860**
C16	(32,42)	**585**	**585**	**585**	585.20	**585.00**	9.56	**585.00**	9.56	585	585	**585**	**585**
C17	(43,56)	**1610**	**1610**	**1610**	1618.30	**1610.00**	5.00	**1610.00**	5.00	1610	1610	**1610**	**1610**
C18	(93,133)	2410	2415	2390	2411.70	**2385.00**	160.09	**2385.00**	160.09	2345	2385	**2385**	**2385**
C19	(62,84)	**1395**	1400	1400	1425.70	1396.33	53.91	**1395.00**	115.09	1395	1395	**1395**	**1395**
C20	(45,64)	**665**	**665**	**665**	668.50	**665.00**	0.64	**665.00**	0.64	665	665	**665**	**665**
C21	(60,84)	**1725**	**1725**	**1725**	1725.20	**1725.00**	1.52	**1725.00**	1.52	1725	1725	**1725**	**1725**
C22	(56,76)	**1070**	**1070**	**1070**	**1070.00**	**1070.00**	0.30	**1070.00**	0.30	1070	1070	**1070**	**1070**
C23	(78,109)	**1690**	1700	1710	1724.30	1690.33	66.86	**1690.00**	76.30	1680	1690	**1690**	**1690**
C24	(77,115)	**1360**	**1360**	**1360**	1368.50	1361.00	104.42	**1360.00**	190.24	1360	1360	**1360**	**1360**
C25	(37,50)	**905**	**905**	**905**	907.00	**905.00**	17.47	**905.00**	17.47	905	905	**905**	**905**
AvgGap		0.12	0.14	0.52	0.98	0.06		0.02		-	-	-	-
AvgTime (scaled)		42.49	42.92	40.76	165.50	36.88		54.22		-	-	-	-

Table A.5
Detailed comparative results of HMA algorithm with 4 state-of-the-art algorithms on the 25 instances of Set D. The average results of all algorithms and the best results of HMA_1 (500 generations) and HMA_2 (2000 generations) are highlighted in bold if they match the BKRs. The optimal BKRs are underlined.

INST	($\|N\|$, $\|E\|$)	Average (median) results								Best results			
		GLS	TSA2	Ant_12	MAENS	HMA_1	Time	HMA_2	Time	LB	BKR	HMA_1	HMA_2
		1 run	1 run	10 runs	30 runs	30 runs		30 runs					
D1	(69,98)	**725**	740	745	745.00	743.83	14.09	738.83	145.70	725	725	**725**	**725**
D2	(48,66)	**480**	**480**	**480**	**480.00**	**480.00**	0.34	**480.00**	0.34	480	480	**480**	**480**
D3	(46,64)	**415**	**415**	**415**	415.20	**415.00**	0.24	**415.00**	0.24	415	415	**415**	**415**
D4	(60,84)	**615**	**615**	**615**	616.00	**615.00**	1.25	**615.00**	1.25	615	615	**615**	**615**
D5	(56,79)	**1040**	**1040**	**1040**	**1040.00**	**1040.00**	0.76	**1040.00**	0.76	1040	1040	**1040**	**1040**
D6	(38,55)	**485**	**485**	**485**	493.00	**485.00**	0.43	**485.00**	0.43	485	485	**485**	**485**
D7	(54,70)	**835**	**835**	855	847.30	838.67	53.72	**835.00**	145.24	835	835	**835**	**835**
D8	(66,88)	**685**	**685**	**685**	704.20	**685.00**	6.98	**685.00**	6.98	685	685	**685**	**685**
D9	(76,117)	**680**	**680**	**680**	**680.00**	**680.00**	0.84	**680.00**	0.84	680	680	**680**	**680**
D10	(60,82)	**910**	**910**	**910**	**910.00**	**910.00**	0.26	**910.00**	0.26	910	910	**910**	**910**
D11	(83,118)	930	960	935	935.20	930.00	38.38	925.33	286.78	920	920	**920**	**920**
D12	(62,88)	**680**	**680**	**680**	**680.00**	**680.00**	4.94	**680.00**	4.94	680	680	**680**	**680**
D13	(40,60)	**690**	695	**690**	691.00	**690.00**	6.70	**690.00**	6.70	690	690	**690**	**690**
D14	(58,79)	**930**	940	**930**	931.00	**930.00**	0.40	**930.00**	0.40	930	930	**930**	**930**
D15	(97,140)	**910**	950	920	919.00	910.50	115.49	**910.00**	141.87	910	910	**910**	**910**
D16	(32,42)	**170**	**170**	**170**	**170.00**	**170.00**	0.13	**170.00**	0.13	170	170	**170**	**170**
D17	(43,56)	**675**	**675**	**675**	**675.00**	**675.00**	0.17	**675.00**	0.17	675	675	**675**	**675**
D18	(93,133)	**930**	**930**	**930**	934.20	**930.00**	3.69	**930.00**	3.69	930	930	**930**	**930**
D19	(62,84)	**680**	690	**680**	**680.00**	**680.00**	0.64	**680.00**	0.64	680	680	**680**	**680**
D20	(45,64)	**415**	**415**	**415**	415.20	**415.00**	0.32	**415.00**	0.32	415	415	**415**	**415**
D21	(60,84)	**805**	825	810	834.20	805.17	79.37	**805.00**	93.50	760	805	**805**	**805**
D22	(56,76)	**690**	**690**	**690**	**690.00**	**690.00**	0.24	**690.00**	0.24	690	690	**690**	**690**
D23	(78,109)	**735**	**735**	**735**	748.20	**735.00**	33.69	**735.00**	33.69	735	735	**735**	**735**
D24	(77,115)	**670**	**670**	**670**	683.50	**670.00**	2.88	**670.00**	2.88	665	670	**670**	**670**
D25	(37,50)	**410**	**410**	**410**	**410.00**	**410.00**	0.19	**410.00**	0.19	410	410	**410**	**410**
AvgGap		0.04	0.66	0.34	0.79	0.17		0.10		-	-	-	-
AvgTime (scaled)		17.36	19.20	51.88	219.53	14.65		35.13		-	-	-	-

Table A.6
Detailed comparative results of HMA algorithm with 4 state-of-the-art algorithms on the 25 instances of Set E. The average results of all algorithms and the best results of HMA_1 (500 generations) and HMA_2 (2000 generations) are highlighted in bold if they match the BKRs. The optimal BKRs are underlined.

INST	($\|N\|$,$\|E\|$)	Average (median) results								Best results			
		GLS	TSA2	Ant_12	MAENS	HMA_1	Time	HMA_2	Time	LB	BKR	HMA_1	HMA_2
		1 run	1 run	10 runs	30 runs	30 runs		30 runs					
E1	(73,105)	1940	**1935**	1945	1967.80	1936.83	102.65	**1935.00**	160.72	1925	1935	**1935**	**1935**
E2	(58,81)	**1610**	**1610**	**1610**	1615.50	**1610.00**	11.38	**1610.00**	11.38	1610	1610	**1610**	**1610**
E3	(46,61)	**750**	**750**	**750**	752.00	**750.00**	1.33	**750.00**	1.33	750	750	**750**	**750**
E4	(70,99)	**1610**	1615	1675	1684.30	**1610.00**	23.60	**1610.00**	23.60	1610	1610	**1610**	**1610**
E5	(68,94)	2170	**2160**	2220	2228.70	2162.33	85.64	**2160.00**	132.12	2160	2160	**2160**	**2160**
E6	(49,66)	**670**	**670**	**670**	**670.00**	**670.00**	0.24	**670.00**	0.24	670	670	**670**	**670**
E7	(73,94)	**1900**	**1900**	**1900**	**1900.00**	**1900.00**	0.63	**1900.00**	0.63	1900	1900	**1900**	**1900**
E8	(74,98)	**2150**	2155	**2150**	2150.50	**2150.00**	2.11	**2150.00**	2.11	2150	2150	**2150**	**2150**
E9	(93,141)	2250	2300	2295	2327.70	2247.50	170.23	2228.67	498.99	2220	2225	**2225**	**2225**
E10	(56,76)	**1690**	**1690**	**1690**	1691.50	**1690.00**	1.44	**1690.00**	1.44	1690	1690	**1690**	**1690**
E11	(80,113)	1850	1855	1860	1932.00	1846.67	96.44	1840.00	460.01	1830	1830	**1830**	**1830**
E12	(74,103)	1710	1730	1760	1764.30	1723.83	124.02	1707.00	341.32	1695	1695	1700	**1695**
E13	(49,73)	**1325**	**1325**	**1325**	1335.30	**1325.00**	7.97	**1325.00**	7.97	1325	1325	**1325**	**1325**
E14	(53,72)	**1810**	**1810**	**1810**	1817.00	1810.67	46.57	**1810.00**	51.25	1810	1810	**1810**	**1810**
E15	(85,126)	1610	1610	1610	1617.80	1601.00	149.37	1599.00	723.39	1590	1590	1600	**1590**
E16	(60,80)	**1825**	**1825**	**1825**	**1825.00**	**1825.00**	39.74	**1825.00**	39.74	1825	1825	**1825**	**1825**
E17	(38,50)	**1290**	**1290**	**1290**	1294.30	1291.00	18.71	**1290.00**	26.35	1290	1290	**1290**	**1290**
E18	(78,110)	**1610**	**1610**	**1610**	1612.30	**1610.00**	1.41	**1610.00**	1.41	1610	1610	**1610**	**1610**
E19	(77,103)	**1435**	**1435**	**1435**	1437.00	**1435.00**	3.54	**1435.00**	3.54	1435	1435	**1435**	**1435**
E20	(56,80)	**990**	**990**	**990**	**990.00**	**990.00**	0.64	**990.00**	0.64	990	990	**990**	**990**
E21	(57,82)	**1705**	**1705**	1760	1755.50	**1705.00**	51.59	**1705.00**	51.59	1705	1705	**1705**	**1705**
E22	(54,73)	**1185**	**1185**	**1185**	1187.50	**1185.00**	1.42	**1185.00**	1.42	1185	1185	**1185**	**1185**
E23	(93,130)	**1430**	1445	1435	1469.00	1432.33	88.98	1430.33	336.54	1430	1430	**1430**	**1430**
E24	(97,142)	**1785**	**1785**	**1785**	1822.20	**1785.00**	35.38	**1785.00**	35.38	1785	1785	**1785**	**1785**
E25	(26,35)	**655**	**655**	**655**	**655.00**	**655.00**	0.45	**655.00**	0.45	655	655	**655**	**655**
AvgGap		0.20	0.39	0.83	1.44	0.19		0.08		-	-	-	-
AvgTime (scaled)		40.65	46.00	40.54	160.89	42.62		116.54		-	-	-	

Table A.7
Detailed comparative results of HMA algorithm with 4 state-of-the-art algorithms on the 25 instances of Set F. The average results of all algorithms and the best results of HMA_1 (500 generations) are highlighted in bold if they match the BKRs. The optimal BKRs are underlined.

INST	($\|N\|$,$\|E\|$)	Average (median) results						Best results		
		GLS	TSA2	Ant_12	MAENS	HMA_1	Time	LB	BKR	HMA_1
		1 run	1 run	10 runs	30 runs	30 runs				
F1	(73,105)	**1065**	1085	**1065**	1071.00	**1065.00**	43.29	1065	1065	**1065**
F2	(58,81)	**920**	**920**	**920**	**920.00**	**920.00**	0.67	920	920	**920**
F3	(46,61)	**400**	**400**	**400**	**400.00**	**400.00**	0.25	400	400	**400**
F4	(70,99)	**940**	960	955	963.50	**940.00**	22.54	940	940	**940**
F5	(68,94)	**1180**	**1180**	**1180**	1180.30	**1180.00**	3.86	1180	1180	**1180**
F6	(49,66)	**490**	**490**	**490**	**490.00**	**490.00**	0.22	490	490	**490**
F7	(73,94)	**1080**	**1080**	**1080**	1090.70	**1080.00**	3.00	1080	1080	**1080**
F8	(74,98)	**1145**	**1145**	**1145**	**1145.00**	**1145.00**	0.41	1145	1145	**1145**
F9	(93,141)	**1145**	1170	1225	1197.80	**1145.00**	11.24	1145	1145	**1145**
F10	(56,76)	**1010**	**1010**	**1010**	**1010.00**	**1010.00**	0.23	1010	1010	**1010**
F11	(80,113)	**1015**	**1015**	1045	1037.50	**1015.00**	6.56	1015	1015	**1015**
F12	(74,103)	**910**	**910**	975	939.50	**910.00**	10.40	910	910	**910**
F13	(49,73)	**835**	**835**	**835**	**835.00**	**835.00**	0.63	835	835	**835**
F14	(53,72)	**1025**	1035	**1025**	1065.50	**1025.00**	9.62	1025	1025	**1025**
F15	(85,126)	**945**	990	**945**	951.70	**945.00**	2.41	945	945	**945**
F16	(60,80)	**775**	**775**	**775**	**775.00**	**775.00**	0.45	775	775	**775**
F17	(38,50)	**605**	630	**605**	**605.00**	**605.00**	0.15	605	605	**605**
F18	(78,110)	**850**	**850**	**850**	861.20	**850.00**	0.85	840	850	**850**
F19	(77,103)	**725**	740	**725**	**725.00**	**725.00**	3.21	715	725	**725**
F20	(56,80)	**610**	**610**	**610**	614.80	**610.00**	6.38	610	610	**610**
F21	(57,82)	**905**	**905**	**905**	**905.00**	**905.00**	2.43	905	905	**905**
F22	(54,73)	**790**	**790**	**790**	**790.00**	**790.00**	0.85	790	790	**790**
F23	(93,130)	**725**	730	730	736.30	**725.00**	27.86	725	725	**725**
F24	(97,142)	**975**	1010	**975**	1001.30	**975.00**	53.28	975	975	**975**
F25	(26,35)	**430**	**430**	**430**	**430.00**	**430.00**	0.12	430	430	**430**
AvgGap		0.00	0.90	0.77	1.01	0.00		-	-	
AvgTime (scaled)		10.67	21.09	52.17	166.85	8.44		-	-	