Engineering Applications of Artificial Intelligence 95 (2020) 103890

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

An effective memetic algorithm for the generalized bike-sharing rebalancing problem

Yongliang Lu [a], Una Benlic [b,c], Qinghua Wu [a,*]

[a] School of Management, Huazhong University of Science and Technology, No. 1037, Luoyu Road, Wuhan, China
[b] Tesco PLC, Lever Building, 85 Clerkenwell Rd, Holborn, London EC1R 5AR, UK
[c] School of Electrical and Automation Engineering, East China Jiaotong University, Nanchang, China

ARTICLE INFO

Keywords:
Heuristics
Bike-sharing rebalancing
Memetic algorithm
Evolutionary local search

ABSTRACT

The generalized bike-sharing rebalancing problem (BRP) entails driving a fleet of capacitated vehicles to rebalance bicycles among bike-sharing system stations at a minimum cost. To solve this NP-hard problem, we present a highly effective memetic algorithm that combines (i) a randomized greedy construction method for initial solution generation, (ii) a route-copy-based crossover operator for solution recombination, and (iii) an effective evolutionary local search for solution improvement integrating an adaptive randomized mutation procedure. Computational experiments on real-world benchmark instances indicate a remarkable performance of the proposed approach with an improvement in the best-known results (new upper bounds) in more than 46% of the cases. In terms of the computational efficiency, the proposed algorithm shows to be nearly two to six times faster when compared to the existing state-of-the-art heuristics. In addition to the generalized BRP, the algorithm can be easily adapted to solve the one-commodity pickup-and-delivery vehicle routing problem with distance constraints, as well as the multi-commodity many-to-many vehicle routing problem with simultaneous pickup and delivery.

1. Introduction

With the increased popularity of bike sharing systems (BSSs) across the world in the recent years, the bike-sharing rebalancing problem (BRP) (Dell'Amico et al., 2014) has received considerable attention in the optimization literature. In BSSs, each station has an inventory with an initial number of bikes and a number of free slots where users can return bikes. Users can rent a bike at any station and return it to a station with vacant slots. Throughout the day, some stations might run out of bikes or free slots to store the returned bikes. Therefore, rebalancing is usually conducted by means of capacitated vehicles based at a central depot to relocate bicycles from a station with an excess to a station with a shortage.

Different variations of BRPs have been investigated in the literature, including single-vehicle rebalancing (Erdoğan et al., 2014; Ho and Szeto, 2014; Benchimol et al., 2011; Li et al., 2016; Chemla et al., 2013), multi-vehicle rebalancing (Lin and Chou, 2012; Raviv et al., 2013; Forma et al., 2015; Dell'Amico et al., 2014; Di Gaspero et al., 2013; Rainer-Harbach et al., 2015; Espegren et al., 2016; Alvarez-Valdes et al., 2016; Bulhões et al., 2018), and dynamic rebalancing (Contardo et al., 2012; Kloimüllner et al., 2014; Zhang et al., 2017). In some studies, additional features have been considered, such as stochastic demands for stations (Dell'Amico et al., 2018), forbidden temporary operations (Bruck et al., 2019), split deliveries (Di Gaspero et al., 2013), multiple types of bikes (Li et al., 2016), etc. According to the classification scheme introduced in Berbeglia et al. (2007), BRP can be modeled as a many-to-many pickup-and-delivery vehicle routing problem. Unlike the classic pickup-and-delivery vehicle routing problem (PDVRP), BRP includes multiple origins and destinations of requests where bikes collected from a pickup station can be supplied to any delivery station. This introduces an additional layer of complexity compared to PDVRP, making it an NP-hard problem.

This work tackles the generalized BRP (Dell'Amico et al., 2014) considering multiple vehicles and the minimizing of the total travel cost. Indeed, the bike rebalancing systems in practice usually involve a fleet of vehicles, while considering the total distance traveled by the repositioning vehicles as the primary problem objective. The problem is formally formulated in Dell'Amico et al. (2014) as follows. Given a complete weighted graph $G = (V, E)$, let $V = \{0, 1, \dots, n\}$ represent a set of vertices in which the first vertex is the depot and the remaining vertices represent the stations. Each edge $(i, j) \in E$ $(i, j \in V)$ is associated with a weight c_{ij} that represents the required travel cost, and satisfies the triangle inequality rule. Each station i has a request q_i, which can be either positive or negative. If $q_i > 0$, then i is a pickup station, where q_i bikes should be collected. If $q_i < 0$, then i

* Corresponding author.
E-mail addresses: luyongliang@hust.edu.cn (Y. Lu), unaben@gmail.com (U. Benlic), qinghuawu1005@gmail.com (Q. Wu).

https://doi.org/10.1016/j.engappai.2020.103890
Received 17 January 2020; Received in revised form 11 May 2020; Accepted 6 August 2020
Available online 20 August 2020

is a delivery station, where $-q_i$ bikes should be supplied. The depot can be considered as a station with $q_0 = 0$. Any bike collected at a pickup station can be supplied to a delivery station or can be returned to the depot. Vehicles can depart from the depot without necessarily leaving it empty, if needed. The aim of the problem is to minimize the total travel cost of driving a fleet of identical vehicles with a limited capacity Q starting from the depot, servicing each station $i \in V \setminus \{0\}$ exactly once, and finishing at the depot, while satisfying the vehicle capacity constraint.

Given the computational challenge of BRP, heuristic and metaheuristic algorithms are very suitable for finding near-optimal solutions to large BRP instances that cannot be solved exactly in an acceptable computation time. In recent years, heuristic and metaheuristic algorithms have received a lot of attention in different areas, including controller tuning in engineering (Nedic et al., 2015; Stojanovic et al., 2016; Stojanovic and Nedic, 2016). In this work, we present the first population-based memetic algorithm (MA) that is able to find high-quality solutions to the generalized BRP.

We summarize the main contributions as follows:

- The proposed MA has two original features. First, MA uses a modified route-copy-based crossover operator for solution recombination. This crossover is specially designed for BRP to guarantee a valid search diversification. Second, MA applies a powerful evolutionary local search (ELS) for solution improvement which integrates: (i) a memory-enhanced variable neighborhood descent with random neighborhood ordering (RVND) and (ii) an adaptive randomized mutation procedure to generate multiple distinct offspring solutions.
- We perform extensive computational assessments on 90 commonly used real-world benchmark instances (Dell'Amico et al., 2014, 2016), which confirm the high competitiveness of the proposed MA approach compared to the two existing state-of-the-art algorithms. In particular, MA is able to report new upper bounds for 42 medium and large instances and match the best-known result for the remaining 48 small instances. For a majority of large instances, it outperforms by a significant margin the two reference methods in terms of the solution quality. Furthermore, the proposed algorithm is nearly two to six times faster in attaining its best solution compared to the state-of-the-art heuristics, leaving it the current most efficient algorithm for the generalized BRP.
- The proposed MA algorithm can be adapted to other similar problems. In our work, we develop two improved versions of MA to solve the one-commodity pickup-and-delivery vehicle routing problem with distance constraints (1-PDVRPD) (Shi et al., 2009) and the multi-commodity many-to-many vehicle routing problem with simultaneous pickup and delivery (M-M-VRPSPD) (Zhang et al., 2019). Extensive experimental tests disclose that the adapted MAs are able to provide competitive results for the two problems.

The remainder of this paper is organized as follows. Section 2 provides a brief summary of different BRP variations, followed by a detailed description of the proposed algorithm in Section 3. Computational results and comparisons with state-of-the-art approaches are given in Section 4. A discussion on the key algorithmic ingredients is provided in Section 5, followed by conclusions and future research directions.

2. Literature review

BRPs have received considerable attention over the past decade. Numerous approaches have been proposed for different variations of BRP, including exact algorithms, approximate algorithms, heuristics, and metaheuristics.

For the case of the static single-vehicle bike repositioning, Erdoğan et al. (2014) proposed a branch-and-cut (B&C) algorithm to solve the single-vehicle BRP with demand intervals, where the number of bicycles at the inventory of each station lies within an interval. Ho and Szeto (2014) proposed an iterated tabu search for the selective single-vehicle BRP, where the constraint in which all the stations must be visited is not strictly imposed. Benchimol et al. (2011) introduced a BRP version for general networks and presented lower-bounding techniques, approximation algorithms, and a polynomial algorithm for the single-vehicle case. Li et al. (2016) introduced a new single-vehicle BRP with multiple types of bikes. The problem is then formulated as a mixed-integer programming model and heuristically solved by means of a hybrid genetic algorithm (GA). Chemla et al. (2013) coped with a new variant of the single-vehicle BRP in which each station can be visited several times and be used as temporary storage. They proposed a B&C algorithm and a tabu search algorithm to solve the problem. Subsequently, an exact separation algorithm (Erdoğan et al., 2015) and an iterated local search (ILS) heuristic algorithm (Cruz et al., 2017) were presented for this same problem.

In the studies on multi-vehicle bike repositioning problems, Lin and Chou (2012) applied an actual path distance optimization method from VRP to deal with a multi-vehicle BRP. Raviv et al. (2013) studied a multi-vehicle static repositioning in a BSS for which the objective is to minimize the total penalty and operational costs. They proposed two mixed-integer linear programming formulations based on different model assumptions that are capable of solving problem instances with up to 60 stations within acceptable optimality gaps using CPLEX (12.3). Forma et al. (2015) proposed a three-step mathematical programming-based heuristic for a static repositioning problem where the objective is to minimize the number of unserved users and the total traveling distance.

Dell'Amico et al. (2014) introduced a generalized BRP with multiple vehicles. Their problem involves determining routes for a fleet of capacitated vehicles with the aim of minimizing the total travel cost of redistributing bikes across BSS stations. Moreover, each station has to be visited exactly once. The authors further proposed four mixed-integer linear programming models and a B&C algorithm to solve symmetric and asymmetric real-world instances with up to 50 vertices. Recently, Dell'Amico et al. (2016) proposed an effective destroy-and-repair (DR) metaheuristic involving a new construction strategy and several local search operators. They apply the proposed approach on a new set of large real-world instances. In addition, the authors tackled a related version of the problem with distance constraints by means of an adaptation of the proposed metaheuristic.

Several variants of the multi-vehicle BRPs with multiple visits, where stations are allowed to be visited more than once, have been considered in the literature (Di Gaspero et al., 2013; Rainer-Harbach et al., 2015; Espegren et al., 2016; Alvarez-Valdes et al., 2016; Bulhões et al., 2018). Di Gaspero et al. (2013) tackled the balancing problem of the Citybike Vienna BSS by means of a hybrid ACO+CP approach that combines a novel constraint programming (CP) model with an ant colony optimization (ACO). Rainer-Harbach et al. (2015) dealt with a static variant of the bike rebalancing problem, where the goal is to minimize the deviation from the target number of bikes, the total loading/unloading cost, and the total duration of all the routes. They investigated the performance of several heuristics, including variable neighborhood descent (VND), greedy randomized adaptive search procedure (GRASP), and variable neighborhood search (VNS). Bulhões et al. (2018) introduced a BRP version with multiple vehicles and visits. Unlike the generalized BRP, vehicles have service time limits and are allowed to visit stations multiple times. The authors proposed a B&C algorithm with an extended network-based mathematical formulation and an ILS metaheuristic algorithm. Alvarez-Valdes et al. (2016) focused on static repositioning systems and proposed several redistribution algorithms based on the estimates of unsatisfied demand at each station.

In addition, Dell'Amico et al. (2018) proposed a BRP with stochastic demands. They developed stochastic programming models that they solve using different approaches, including L-shaped methods and a heuristic algorithm based on the VND local search procedure. Bruck et al. (2019) presented a static BRP with forbidden temporary operations. In this problem, stations can be visited multiple times but cannot be used as depots to either temporarily drop off bikes or to temporarily pick up bikes. The authors presented an integer programming model based on a binary network expansion that they solve using a branch-and-reject algorithm.

Several dynamic versions of BRP have been addressed (Contardo et al., 2012; Kloimüllner et al., 2014; Zhang et al., 2017). Contardo et al. (2012) introduced three mathematical formulations to define a dynamic rebalancing problem, and proposed a solution method based on Dantzig and Benders decompositions for medium and large-size instances. Kloimüllner et al. (2014) extended a static BRP (Rainer-Harbach et al., 2015) to model the dynamic case and proposed an adaption of the VNS and GRASP metaheuristics to tackle the problem. Zhang et al. (2017) proposed a time–space network flow model to simultaneously consider user dissatisfaction forecasting, bicycle repositioning, and vehicle routing. They tackled the model by means of a matheuristic algorithm.

Population-based genetic algorithms (GAs) is a well-known algorithmic framework in combinatorial optimization (Goldberg, 1989) which has been applied with great success to a variety of difficult optimization problems (e.g., Savino et al., 2014b,a). While GAs integrate useful mechanisms to facilitate the exploration of the search space, local search approaches focus on exploiting a limited region of the search space. To enrich the BRP literature, we thus propose a population-based memetic algorithm for the generalized BRP that combines the diversification strength of a GA and the intensification power of a local search strategy.

3. Memetic algorithm for the generalized BRP

3.1. Main scheme

Memetic algorithm (MA) (Neri et al., 2012) is a powerful optimization framework that combines the exploration power of population-based algorithms and the exploitation strength of neighborhood-based local search. MAs have been applied with great success to numerous difficult optimization problems (Hao, 2012; Neri et al., 2012; Neri and Cotta, 2012), including various routing problems, such as the traveling salesman

problem (Lu et al., 2018), the vehicle routing problem (Cattaruzza et al., 2014), the traveling repairman problem (Lu et al., 2019), and general routing with nodes, edges, and arcs (Prins and Bouchenoua, 2005).

Algorithm 1 Memetic algorithm for the generalized BRP

Require: $G = (V, E)$: a BRP instance; p: population size; t_{max}: timeout limit
Ensure: best solution S^* found so far

$P \leftarrow Population\_Initialization(G, p)$ /* Section 3.3 */
$S^* \leftarrow Best(P)$ /* S^* records the best solution found so far */
while $time() < t_{max}$ **do**
  Randomly select S_1 and S_2 from P
  $(o_1, o_2) \leftarrow$ Crossover(S_1, S_2) /* Section 3.5 */
  for each offspring $o \in (o_1, o_2)$ **do**
    $S \leftarrow$ ELS(o) /* Section 3.4 */
    if $f(S) < f(S^*)$ **then**
      $S^* \leftarrow S$
    end if
    $P \leftarrow Population\_Update(P, S)$ /* Section 3.6 */
  end for
end while
Return S^*

The general architecture of our MA for the generalized BRP is summarized in Algorithm 1. Starting with a population of initial solutions (Section 3.3), it performs a series of cycles (also called generations) where each generation alternates between a crossover operator (Section 3.5) and an evolutionary local search (ELS) procedure (Section 3.4). The crossover operator recombines two randomly selected parent solutions from the population to generate two new offspring solutions. The ELS procedure is then used to improve each newly created offspring. Finally, the population is updated in case of improved offspring solutions (Section 3.6). The algorithm terminates when the timeout limit $t_{\max}$ is reached.

To make the structure of the proposed algorithm more understandable, a comprehensive flowchart of the MA procedure is given in Fig. 1. A detailed description of the algorithmic components is provided in the following sections.

3.2. Solution representation and evaluation

Each solution is represented by a set of vehicle routes $\{r_1, r_2, \ldots, r_m\}$, where m is the number of routes and each route r_i ($i \leq m$) is represented as a permutation of vertices (stations) $(v_0, v_1, \ldots, v_i, v_0)$, $v_0 = 0$ defining the order in which the stations are visited by a single vehicle. The evaluation function considered in this work is the total travel distance summed across each route.

3.3. Population initialization

Algorithm 2 RGCP procedure

Require: $G = (V, E)$: a BRP instance; a: size of restricted candidate list (RCL)
Ensure: an initial feasible solution S

Initialize vertex list $U \leftarrow \{v_1, v_2, ..., v_n\}$
$m \leftarrow 1$
Create an empty route r_m starting at the depot vertex v_0
Randomly select a vertex v from U as the second vertex of r_m
Remove vertex v from U
while $U \neq \emptyset$ **do**
  Build a restricted candidate list RCL consisting of the a closest vertices that can be feasibly added at the end of r_m without violating the capacity constraint
  if $RCL \neq \emptyset$ **then**
    Randomly select a vertex v from RCL and add it at the end of r_m
    Remove vertex v from U
  else
    Complete the construction of r_m by connecting the last vertex with the depot vertex v_0
    $m \leftarrow m + 1$
    Create a new empty route r_m that includes only the depot vertex v_0
    Randomly select a vertex v from U as the second vertex of r_m
    Remove vertex v from U
  end if
end while
$S \leftarrow \{r_1, r_2, ..., r_m\}$
return S

To generate an initial feasible solution S composed of multiple vehicle routes, we propose a randomized greedy construction procedure (RGCP) presented in Algorithm 2. RGCP is an adaptation of the well-known nearest neighbor heuristic for TSP. Specifically, RGCP builds one route at a time in a step-by-step way, where each route starts at the depot vertex and the second vertex (station) $i \notin S$ in the route is selected at random. At each step, RGCP builds a restricted candidate list (RCL) corresponding to a set of unserved stations that are the closest to the last station of the current route and that satisfy the vehicle capacity constraint. If RCL is empty, then the construction of the current route is completed and a new route is started. Otherwise, a vertex from RCL is randomly selected and then added at the end of the current route.

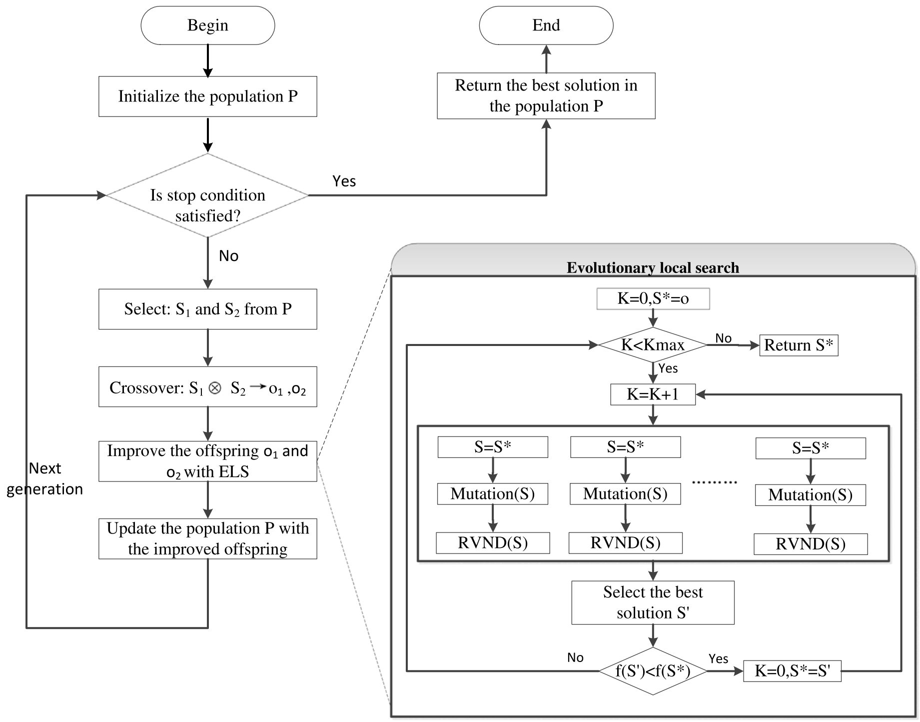


Fig. 1. Flowchart of the proposed memetic algorithm.

This process is repeated until all the stations are included in S. The solution S constructed by RGCP is then further improved by the RVND procedure (Section 3.4.3).

We run the RGCP procedure γ times ($\gamma \geq p$, where p is the population size) to obtain a set of locally optimal solutions and then select the p best solutions to form the initial population. Each solution in the population is further locally improved by the ELS procedure (Section 3.4).

3.4. Evolutionary local search

Evolutionary local search (ELS) approach is an effective metaheuristic proposed by Merz and Wolf (Wolf and Merz, 2007), which extends the classic iterated local search (ILS) (Lourenço et al., 2003). It has so far been successfully applied to several well-studied vehicle routing problems (Duhamel et al., 2011, 2012; Labadie et al., 2011; Zhang et al., 2015).

The main components of an ELS based heuristic are a local improvement procedure and a mutation procedure. Following the general ELS framework, our ELS combines a memory-enhanced variable neighborhood descent based on a random neighborhood ordering (RVND) with an adaptive randomized mutation procedure. The distinguishing features of the proposed ELS algorithm are a memory-based strategy used as an early termination for variable neighborhood descent and a probabilistic mechanism to adaptively adjust the mutation strength. The proposed ELS records information on the recently visited local optima in a short-term memory. The RVND search terminates immediately if an improved neighboring solution has previously been recorded in the memory so as to avoid unnecessary computing efforts. For improved efficiency, the memory stores solution attributes (i.e., the total cost and the number of routes) instead of the complete solutions.

As outlined in Algorithm 3, each iteration of ELS first creates a number of offspring solutions by performing random mutations to the best solution found during the search S^* (Section 3.4.4), followed by local improvement of each offspring with the RVND procedure (Section 3.4.3). The mutation strength η is selected from a given set I_m of values according to the following probability formula: $P(i) = Cnt(i)/\sum_{i=1}^{|I_m|} Cnt(i)$, where $P(i)$ is the probability of selecting the ith value of I_m. Whenever an improved offspring is found using a mutation

Algorithm 3 Evolutionary local search

Require: S_0: an initial solution; nd: maximum number of generated child solutions; I_m: a set of mutation strength values; k_{max}: max allowed iterations without improvement
Ensure: best solution S^*

$k \leftarrow 0$ /* k counts the number of consecutive non-improving iterations */
Initialize Memory
for $i = 1$ to $|I_m|$ **do**
  $Cnt(i) = 1$ /* initialize the counter array Cnt */
end for
$S^* \leftarrow RVND(S_0)$ /* see Section 3.4.3 */
Add S^* to Memory
while $k < k_{max}$ **do**
  $S' \leftarrow S^*$ /* initialize the best child solution */
  for $j = 1, 2, ..., nd$ **do**
    $i \leftarrow Probabilistic\_Select(Cnt)$
    $\eta \leftarrow I_m(i)$
    $S \leftarrow Mutate(S^*, \eta)$ /* see Section 3.4.4 */
    $S \leftarrow RVND(S)$ /* see Section 3.4.3 */
    if $f(S) < f(S')$ **then**
      $S' \leftarrow S$ /* update the best child solution */
      $Cnt(i) \leftarrow Cnt(i) + 1$
    end if
    if $S \notin Memory$ **then**
      Add S into Memory
    end if
  end for
  if $f(S') < f(S^*)$ **then**
    $S^* \leftarrow S'$
    $k \leftarrow 0$
  else
    $k \leftarrow k + 1$
  end if
end while
return S^*

of strength $I_m(i)$, the corresponding counter $Cnt(i)$ is incremented by 1, making it more likely for the mutation strength $I_m(i)$ to be selected in the subsequent iterations. The memory record is then updated with an improved offspring S obtained by RVND. Finally, the best-found solution S^* is updated with the current best offspring S' whenever S' improves on the quality of S^*. The search terminates if no improvement with respect to S^* has been achieved in $k_{\max}$ consecutive iterations, returning S^* as the final output of ELS.

3.4.1. Move operators

The proposed ELS algorithm is based on the joint use of the following five move operators:

- **Block-relocate** (N_1)**:** Remove a block and reinsert it at a new position in the same route or in a different route. A block is a segment of consecutive stations in a route, and its length is limited to 1–3.
- **Block-exchange** (N_2)**:** Exchange two blocks belonging to the same route or to different routes. The size of the blocks is restricted to 1 and 2.
- **Cross** (N_3)**:** Select two routes, and cut each route into two parts. Then, merge the first part of one route with the last part of the other route to generate two new routes.
- **Three-opt*** (N_4)**:** For a given route, delete three edges, and reconnect the three subpaths without changing the orientation of the three subpaths (Fig. 2(a)). In Mladenović et al. (2012), the authors presented a special three-opt denoted as three-opt*. This operator is very suitable for asymmetric instances of BRP.
- **Double-bridge** (N_5)**:** First introduced in Lin and Kernighan (1973) for TSP, this operator involves deleting four edges in the same tour and reconnecting the four subtours without changing the orientation of the four subtours (Fig. 2(b)).

We use $N_1 - N_5$ to denote the five neighborhoods induced by the move operators. For the inter-route *Block-relocate* move operator, an extra empty route is always maintained such that a new route can be added if the operation results in a lower cost.

Given that the length of a block in this study is limited to at most three vertices, the size of the neighborhoods N_1 and N_2 is clearly bounded by $O(n^2)$, where n is the total number of vertices. As described in Dell'Amico et al. (2016), the *Cross* operator also yields $O(n^2)$ neighboring solutions. The *Three-opt** and *Double-bridge* operators are more complex than the others. The *Three-opt** operator leads to a neighborhood whose size is bounded by $O(n^3)$ (Mladenović et al., 2012), while the *Double-bridge* operator is only used for solution mutation (Section 3.4.4) which is performed in $O(1)$ time by changing a constant number of edges.

However, note that the overall computational complexity is higher due to the feasibility checks of each neighboring solution requiring a computation from scratch in $O(n)$ time (Dell'Amico et al., 2016). As feasibility checks become expensive with the increased number of local search iterations, we use auxiliary data structures (ADSs) to accelerate the search (Section 3.4.2) by evaluating feasibility in amortized constant time.

3.4.2. Auxiliary data structures

Let $r = (v_0, v_1, \dots, v_{m+1})$ be a route of a solution S, where $v_1, \dots, v_m$ represents the m vertices while v_0 and v_{m+1} denote the depot. Let $L(v_i) = L(v_{i-1}) + q_{v_i}$ be the load on a vehicle when the vehicle leaves vertex v_i. Initially, $L(v_0) = 0$. If the vehicle leaves the depot with an empty load, then it will reach along the route r a maximum load equal to $\max_{i=0}^{m}\{L(v_i)\}$ that must never exceed the vehicle capacity Q. Notice that the minimum of the cumulative loads $\min_{i=0}^{m}\{L(v_i)\}$ along r can be negative in a feasible route. In that case, if the vehicle acquires an additional initial load with the quantity of $-\min_{i=0}^{m}\{L(v_i)\}$, the maximum load becomes $\max_{i=0}^{m}\{L(v_i)\} - \min_{i=0}^{m}\{L(v_i)\}$, which has to be less than or equal to Q. Therefore, the route r is feasible if and only if Eq. (1) is satisfied (Dell'Amico et al., 2016).

$$\max_{i=0}^{m}\{L(v_i)\} - \min_{i=0}^{m}\{L(v_i)\} \le Q \tag{1}$$

According to the analysis in Section 3.4.1, the feasibility check becomes very expensive with the increased number of the local search iterations. We therefore implement auxiliary data structures (ADS) to accelerate the feasibility checks. For each vertex v_i of r, the method stores and updates

- $L(v_i) = L(v_{i-1}) + q_{v_i}$;
- $L_{\min}(v_i) = \min\{0, L(v_1), L(v_2), \dots, L(v_i)\}$;
- $L_{\max}(v_i) = \max\{0, L(v_1), L(v_2), \dots, L(v_i)\}$;
- $\bar{L}_{\min}(v_i) = \min\{0, L(v_i), L(v_{i+1}), \dots, L(v_m)\}$;
- $\bar{L}_{\max}(v_i) = \max\{0, L(v_i), L(v_{i+1}), \dots, L(v_m)\}$.

We maintain an ADS for each route of the solution and update an ADS each time that the corresponding route is modified.

Let $(v_i, \dots, v_j)$ $(i < j)$ be a block of consecutive vertices in r starting from vertex v_i and ending at vertex v_j, and let $q_{sum}(v_i, v_j) = \sum_{k=i}^{j} q_{v_k}$ be the sum of the loads of the block.

For the above mentioned inter-route operators, fast feasibility checks of the resulting neighboring solutions are can be computed in $O(1)$ time as follows:

Block-relocate. Remove the block $(v_i, \dots, v_j)$ from r and insert it between v_{u-1} and v_u in a different route r'. The new route r is feasible if and only if

$$\max\{0, L_{\max}(v_{i-1}), \bar{L}_{\max}(v_{i+1}) - q_{sum}(v_i, v_j)\} - \min\{0, L_{\min}(v_{i-1}), \bar{L}_{\min}(v_{i+1}) - q_{sum}(v_i, v_j)\} \le Q.$$

Similarly, the new route r' is feasible if and only if

$$\max\{0, L_{\max}(v_{u-1}), L'(v_i), L'(v_{i+1}), \dots, L'(v_j), \bar{L}_{\max}(v_u) + q_{sum}(v_i, v_j)\} - \min\{0, L_{\min}(v_{u-1}), L'(v_i), L'(v_{i+1}), \dots, L'(v_j), \bar{L}_{\min}(v_u) + q_{sum}(v_i, v_j)\} \le Q,$$

where $L'(v_i), \dots, L'(v_j)$ is the new cumulative load of the block after the position change, i.e., $L'(v_i) = L(v_{u-1}) + q_{v_i}$, $L'(v_{i+1}) = L'(v_i) + q_{v_{i+1}}$.

Following the above idea, the feasibility of the inter-route *Block-exchange* moves can also be checked in $O(1)$ time.

Cross. Let u and v be a pair of vertices in two different routes of a solution S, vertices x and y be the respective immediate successors of u and v, and $r(u)$ be the route, including vertex u. Cut the link between (u, x) and (v, y), and establish a link between (u, y) and (v, x). The new route $r(u)$ is feasible if and only if

$$\max\{0, L_{\max}(u), \bar{L}_{\max}(y) - L(v) + L(u)\} - \min\{0, L_{\min}(u), \bar{L}_{\min}(y) - L(v) + L(u)\} \le Q.$$

Similarly, the new route $r(v)$ is feasible if and only if

$$\max\{0, L_{\max}(v), \bar{L}_{\max}(x) - L(u) + L(v)\} - \min\{0, L_{\min}(v), \bar{L}_{\min}(x) - L(u) + L(v)\} \le Q.$$

In addition, the feasibility of all the intra-route move operators can also be evaluated quickly using the information stored in the ADSs. These intra-route move operators always result in a change in a subpath (block) of a route. Assume that a subpath $(v_i, \dots, v_j)$ of a route r is modified to $(v'_i, \dots, v'_j)$ by an intra-route move. The new route r is feasible if and only if

$$\max\{0, L_{\max}(v_{i-1}), L'_{\max}(v'_i, v'_j), \bar{L}_{\max}(v_{j+1})\} - \min\{0, L_{\min}(v_{i-1}), L'_{\min}(v'_i, v'_j), \bar{L}_{\min}(v_{j+1})\} \le Q,$$

where $L'(v'_i), \dots, L'(v'_j)$ need to be calculated (i.e., $L'(v'_i) = L(v_{i-1}) + q_{v_i}$, $L'(v'_{i+1}) = L'(v'_i) + q_{v_{i+1}}$), and $L'_{\min}(v'_i, v'_j)$ and $L'_{\max}(v'_i, v'_j)$ are the minimum and the maximum cumulative load of a subpath $(v'_i, \dots, v'_j)$ respectively.

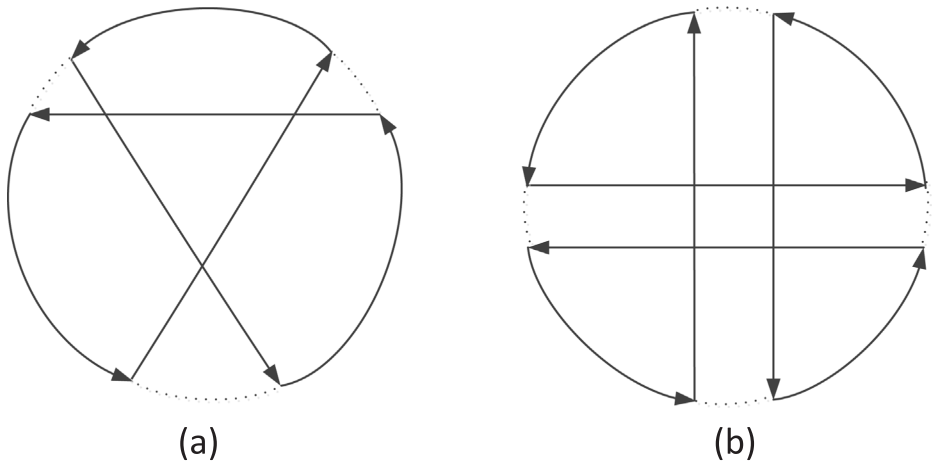


Fig. 2. (a) Three-opt* move; (b) Double-bridge move.

3.4.3. RVND procedure

Algorithm 4 RVND procedure

```
Require: S: current solution
Ensure: updated S
flag ← true
while flag = true do
    flag ← false
    r(·) ← a random permutation of 1,2,3,4;
    for i = 1, 2, 3, 4 do
        (S, improve_tag) ← LocalSearch(S, N_r(i)) /* See Algorithm 5 */
        if improve_tag = true then
            flag ← true
        end if
        if S ∈ Memory then
            flag ← false
            break
        end if
    end for
end while
return S
```

Algorithm 5 Local search procedure

```
Require: S: current solution; N: current neighborhood
Ensure: updated S
improve_tag ← false
Randomly shuffle all feasible moves in neighborhood N
for each feasible move mv ∈ N do
    S' ← S ⊕ mv
    if f(S') < f(S) then
        S ← S'
        improve_tag ← true
    end if
end for
return (S, improve_tag)
```

Algorithm 4 outlines the RVND procedure consisting of an iterative exploitation of neighborhoods $N_1 - N_4$ in a random order to attain a local optimum. For each neighborhood N under consideration and a solution S, each feasible move in N is examined in a random order, and the resulting neighboring solution S' is accepted to replace the current solution S if $f(S') < f(S)$. To reduce the computational time, each neighborhood is explored with the first improvement strategy. The RVND search terminates if a solution returned by first improvement has been previously recorded in the memory. If none of the solutions in neighborhoods $N_1 - N_4$ improves on the quality of S, S is returned as the final solution of the RVND procedure.

3.4.4. Mutation operator

Algorithm 6 Mutation procedure

```
Require: S: current solution; η: mutation strength
Ensure: S: mutated solution
w ← 0
while w < η do
    Randomly pick a neighboring solution S' ∈ N_3(S) ∪ N_5(S)
    if S' is feasible then
        S ← S'
        w ← w + 1
    end if
end while
return S
```

Mutation is performed by applying η random *Cross* or *Double-bridge* moves, where η is a parameter called the mutation strength. Specifically, a feasible neighboring solution S' is picked at random from the combined neighborhood $N_3 \cup N_5$ to replace the current solution S (Algorithm 6).

3.5. Crossover operator

Crossover operator is another important component of MA. It is well-known that the success of MA crucially depends on a crossover operator (Hao, 2012), which should be designed to ensure that the offspring solutions inherit good features of both parents. As BRP is considered as a many-to-many pickup-and-delivery VRP, we experimented with some standard VRP crossover operators (Hosny and Mumford, 2009; Vaira and Kurasova, 2013). According to our experiments, an adaptation of the route-copy-based crossover (RCX) (Hosny and Mumford, 2009) operator for BRP exhibits the best performance.

Given two parent solutions $S_1 = \{r_1^1, r_2^1, \dots, r_{m1}^1\}$ with $m1$ routes and $S_2 = \{r_1^2, r_2^2, \dots, r_{m2}^2\}$ with $m2$ routes, the first offspring o_1 is generated as follows:

Step 1: Copy each route $r_i^1 \in S_1$ over to o_1 with a 0.5 probability.

Step 2: Collect the vertices that are not in o_1 into the relocation pool, while maintaining the original order of appearance as in parent S_2.

Step 3: Use the vertices in the relocation pool to generate new routes for o_1. More precisely, each vertex in the relocation pool is inserted at the best possible position of the new routes of o_1,

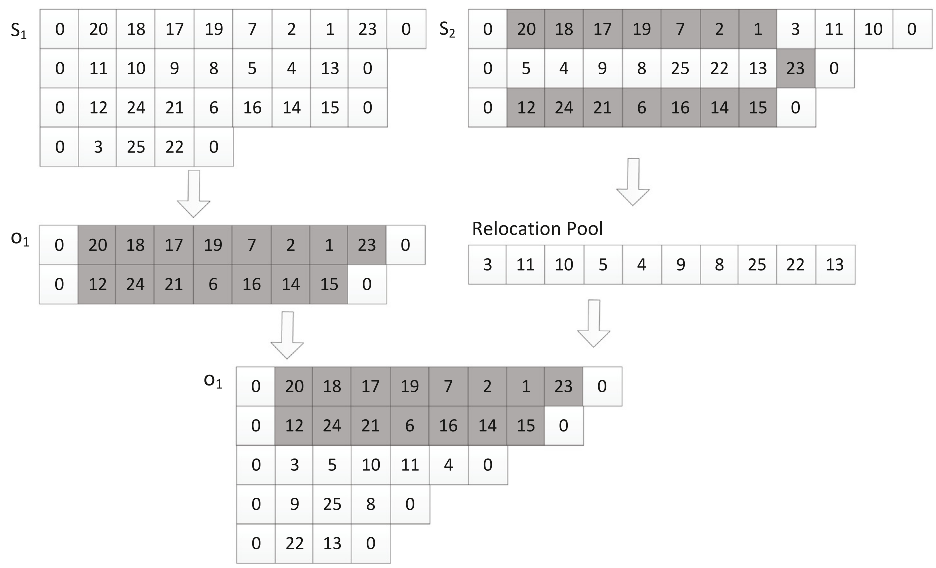


Fig. 3. An example of a crossover with RCX.

while respecting the vehicle capacity constraint. If feasibility cannot be achieved, then create a new route in o_1 containing the critical vertex.

The second offspring o_2 is created in the same way by simply reversing the roles of the two parents. Fig. 3 gives an example of the RCX operation on two parent solutions $S_1 = \{(0, 20, 18, 17, 19, 7, 2, 1, 23, 0), (0, 11, 10, 9, 8, 5, 4, 13, 0), (0, 12, 24, 21, 6, 16, 14, 15, 0), (0, 3, 25, 22, 0)\}$ with 4 routes and $S_2 = \{(0, 20, 18, 17, 19, 7, 2, 1, 3, 11, 10, 0), (0, 5, 4, 9, 8, 25, 22, 13, 23, 0), (0, 12, 24, 21, 6, 16, 14, 15, 0)\}$ with 3 routes. The first and the third route of S_1 are copied over to offspring o_1, while the vertices from the second and the fourth route of S_1 are placed into the relocation pool in the same order of their appearance in S_2. Vertices from the relocation pool are then used to generate new routes in offspring o_1 with the best insertion heuristic.

The proposed RCX operator not only preserves a part of existing routes, but also helps maintain the diversity of the population by creating new routes that are not present in the parent solutions. As confirmed by the computational experiments in Section 4.7, RCX plays the key role for the performance of MA.

3.6. Population updating rule

To manage the population, we use a traditional "Pool Worst" updating strategy to decide whether an offspring S improved by ELS is to be introduced into the population. The mechanism simply replaces the worst member S_w from the population P if $S \notin P$ and $f(S) < f(S_w)$.

3.7. Computational complexity of the proposed algorithms

In any memetic algorithm, most of the computing effort is generally spent by a local improvement procedure to improve the quality of offspring solutions. As described in Section 3.4.1, the size of the neighborhoods N_1, N_2 and N_3 is clearly bounded by $O(n^2)$, while N_4 leads to a neighborhood bounded by $O(n^3)$. Owing to ADS, feasibility checks of neighboring solutions can be performed in amortized constant time. As the RVND approach iteratively exploits the four neighborhoods $N_1 - N_4$ to attain a local optimum, the total complexity of RVND is bounded by $O(n^3 \times maxIter_1)$, where $maxIter_1$ is the maximum number of RVND iterations.

As described in Algorithm 3, each iteration of ELS generates nd offspring solutions by performing random mutations followed by local improvement with RVND. Given that the complexity of the randomized mutation procedure is $O(1)$, the total computational complexity of a ELS iteration is bounded by $O(n^3 \times maxIter_1 \times nd)$. Therefore, the overall complexity of ELS is bounded by $O(n^3 \times maxIter_1 \times nd \times maxIter_2)$, where $maxIter_2$ is the maximum number of ELS iterations. For subsequent convenience, we denote the total time complexity of ELS as $O(ELS)$.

We next consider the time complexity of the population initialization RGCP algorithm (see Algorithm 2) and the RCX crossover operator (see Section 3.5). RGCP is an adaptation of the well-known nearest neighbor heuristic for TSP. RCX generates offspring solutions by copying a subset of complete routes from a parent to an offspring, while building new routes from the remaining vertices with the best insertion heuristic. Both RGCP and RCX perform in $O(n^2)$.

In a nutshell, the proposed MA (see Algorithm 1) starts with an initial population of p solutions generated with RGCP in $O(n^2)$. At each generation of MA, the RCX crossover generates two offspring solution in $O(n^2)$ time, followed by offspring improvement with ELS in $O(ELS)$ time and the population update in $O(n)$. Consequently, the total time complexity of each generation of MA is bounded by $O(ELS)$.

4. Computational experiments

In this section, we evaluate the performance of the proposed MA by means of extensive comparisons with the existing exact and heuristic algorithms from the literature.

4.1. Benchmark instances

For experimental evaluations, we test our MA on 90 real-world instances collected from 32 BSSs across different cities, which were previously used in Dell'Amico et al. (2014, 2016). In practical BSSs, the most common vehicle capacities are 10, 20 and 30 bikes. Consequently, Dell'Amico et al. (2014, 2016) use these values for their experimental evaluations. Table 1 presents the main features of the given real-world instances, including the name of the city, the country, the number of vertices, as well as the minimal, the average, and the maximum values of the requests and of the travel distances. The asymmetric travel distances between stations were obtained from the geographical system data.

According to Dell'Amico et al. (2016), the 90 real-world instances can be grouped into three sets: small-size instances with $|V| \leq 50$, medium-size instances with $50 < |V| < 100$, and large-size instances with $|V| \geq 100$.

Table 1
Benchmark instances.

City	Country	$\|V\|$	$\min\{q_i\}$	$\text{avg}\{q_i\}$	$\max\{q_i\}$	$\min\{c_{ij}\}$	$\text{avg}\{c_{ij}\}$	$\max\{c_{ij}\}$
Bari	Italy	13	−5	−1.54	5	400	2283.97	5400
ReggioEmilia	Italy	14	−10	−2.00	3	300	2095.05	5500
Bergamo	Italy	15	−12	−1.00	10	100	1532.86	3200
Parma	Italy	15	−6	−1.07	4	200	3121.43	8800
Treviso	Italy	18	−4	−0.83	3	340	3510.99	8398
LaSpezia	Italy	20	−5	0.05	6	193	2521.61	6128
BuenosAires	Argentina	21	−20	−0.43	20	689	4676.75	12780
Ottawa	Canada	21	−5	−0.05	5	180	2219.15	5030
SanAntonio	US	23	−4	1.74	8	98	1950.98	4808
Brescia	Italy	27	−11	−1.41	4	200	2571.65	6600
Roma	Italy	28	−17	−2.36	18	200	5989.55	27400
Madison	US	28	−6	0.29	8	53	3085.99	9922
Guadalajara	Mexico	41	−11	−1.07	1	60	3278.30	14728
Dublin	Ireland	45	−11	−1.42	6	203	2190.50	4734
Denver	US	51	−8	−0.69	7	211	3873.94	12000
RiodeJaneiro	Brazil	55	−7	−1.47	7	420	5328.35	16591
Boston	US	59	−8	−0.27	16	243	3911.82	19239
Torino	Italy	75	−7	−0.49	9	23	2527.84	7200
Toronto	US	80	−11	0.15	12	150	2339.03	6283
Miami	US	82	−8	−2.24	9	68	4000.00	13771
CiudaddeMexico	Mexico	90	−17	−0.97	17	15	2551.94	7264
Minneapolis	US	116	−9	−0.79	5	6	6045.24	19468
Brisbane	Australia	150	−15	1.27	17	2	3769.07	13959
Milano	Italy	184	−18	0.88	17	6	3313.63	8126
Lille	France	200	−20	−0.42	16	163	7450.82	18877
Toulouse	France	240	−13	−1.49	12	6	3943.35	12459
Sevilla	Spain	258	−20	−1.56	10	2	4652.58	13510
Valencia	Spain	276	−20	−1.50	14	134	4241.34	13413
Bruxelles	Belgium	304	−13	−0.18	16	211	5844.93	19032
Lyon	France	336	−20	−0.70	17	18	4817.44	47657
Barcelona	Spain	410	−17	−2.56	19	2	4699.91	13671
London	U.K.	564	−28	−0.58	29	2	5833.37	19412

4.2. Experimental settings

The proposed MA is coded in C++ language and tested on a machine with an Intel E5-2670 processor (2.8 GHz) and 4 GB of RAM, running under Linux operating system.

Table 2 shows the setting of the MA parameters used in the reported experiments. The mutation strength η is adaptively chosen during the search from the set I_m of possible values. The rest of the parameters $(p, \gamma, a, k_{\max}, nd)$ were tuned using iterated F-race (IFR) (Birattari et al., 2010), a racing algorithm for offline configuration of parameterized algorithms. The tuning was performed on a random selection of 10 large BRP instances with the total time budget set to 500 executions of MA each limited to 600 seconds. To evaluate the sensitivity of each parameter, we test each considered value from Table 2 while fixing the other parameters to their final values. We perform 10 independent runs for each considered value on the 10 representative instances. The Friedman non-parametric statistical test reveals a statistically significant difference in performances for different values of parameter p (p-value = 0.05437) and parameter $k_{\max}$ (p-value = 6.173e−07), while the remaining parameters do not exhibit sensitivity.

To evaluate the performance of MA, we present comparisons with a destroy-and-repair (DR) algorithm proposed in Dell'Amico et al. (2016), which to the best of our knowledge is the only heuristic used for the generalized BRP. The results reported by DR for each instance were obtained across ten independent executions on a computer with an Intel Core i3-2100 with 3.10 GHZ and 4 GB RAM. The time limit for each run was set to 10 s for small-size instances, 600 s for medium-size instances, and 1800 s for large-size instances. We further provide comparisons with a branch-and-cut (B&C) algorithm from Dell'Amico et al. (2014), executed under the same computing platform as DR.

According to our experiments, the proposed MA always reports a solution of equal or better quality than DR within a shorter computing time. We thus reduce the cutoff times of our MA to 3 s for small-size instances, 180 s for medium-size instances, and 600 s for large-size instances. As Dell'Amico et al. (2016), we perform ten independent runs of MA for each test case with the above stopping condition.

4.3. Computational results and comparisons with the existing heuristic approach

Tables 3–5 provide detailed computational results reported by MA and DR (Dell'Amico et al., 2016) on the sets of small, medium, and large instances respectively. The first three columns show the instance name, the number of vertices ($|V|$), and the vehicle capacity (Q). For each instance and algorithm, columns "f_{best}", "f_{avg}", and "$t(s)$" respectively show the best objective value, the average objective value and the average computing time in seconds required to reach the best objective. For each small instance, we indicate the optimal objective value as reported in Dell'Amico et al. (2016). The last column provides the percentage gap between the best objective values obtained with MA and DR, computed as $gap = 100 \times \frac{f_{MA} - f_{DR}}{f_{DR}}$. The entries in bold indicate the cases where MA improves on the current best-known solution from the literature (i.e., when the percentage gap is negative). Row 'Avg.' provides the average computation time in seconds required by each algorithm to reach its best objective value, as well as the average gap between the best reported objective values obtained with MA and DR.

For the set of 41 small instances shown in Table 3, both MA a DR are able to reach the optimal solution, although DR requires in some cases up to several seconds longer than MA to find the optimum.

As shown in Tables 4–5, the proposed MA reports new improved upper bounds for 42 out of the 49 medium and large instances, while attaining the same solution quality as DR on the remaining 7 instances. Furthermore, we observe that the percentage improvement with MA generally increases with the increase in the instance size, exceeding 4% in one of the cases (Table 5). Although the experimental platforms of the two algorithms are different, another advantage of MA lies in its speed as it requires significantly less time than DR to attain the reported results.

Table 2
Parameter tuning results.

Parameter	Section	Description	Considered values	Final value
p	3.3	Population size	{5,10,20}	5
γ	3.3	Number of runs of the randomized construction procedure	{10,30,50}	30
a	3.3	Size of restricted candidate list (RCL)	{2,5,10}	5
k_{max}	3.4	Search depth during evolutionary local search	{10,30,50}	30
nd	3.4	Number of generated offspring solutions	{5,10,15}	10
I_m	3.4	A set of mutation strength values	–	{2,3,4,5,6,7,8,9,10}

Table 3
Computational results on small-size instances.

Instance	$\|V\|$	Q	$Exact$	DR (10 s)			MA (3 s)			
				f_{best}	f_{avg}	$t(s)$	f_{best}	f_{avg}	$t(s)$	gap
Bari	13	30	14600	14600	14600.0	0.00	14600	14600.0	0.01	0.00
Bari	13	20	15700	15700	15700.0	0.00	15700	15700.0	0.01	0.00
Bari	13	10	20600	20600	20600.0	0.00	20600	20600.0	0.01	0.00
ReggioEmilia	14	30	16900	16900	16900.0	0.00	16900	16900.0	0.01	0.00
ReggioEmilia	14	20	23200	23200	23200.0	0.00	23200	23200.0	0.01	0.00
ReggioEmilia	14	10	32500	32500	32500.0	0.01	32500	32500.0	0.03	0.00
Bergamo	15	30	12600	12600	12600.0	0.01	12600	12600.0	0.02	0.00
Bergamo	15	20	12700	12700	12700.0	0.01	12700	12700.0	0.02	0.00
Bergamo	15	12	13500	13500	13500.0	0.00	13500	13500.0	0.01	0.00
Parma	15	30	29000	29000	29000.0	0.00	29000	29000.0	0.01	0.00
Parma	15	20	29000	29000	29000.0	0.00	29000	29000.0	0.01	0.00
Parma	15	10	32500	32500	32500.0	0.00	32500	32500.0	0.02	0.00
Treviso	18	30	29259	29259	29259.0	0.04	29259	29259.0	0.02	0.00
Treviso	18	20	29259	29259	29259.0	0.04	29259	29259.0	0.01	0.00
Treviso	18	10	31443	31443	31443.0	0.08	31443	31443.0	0.02	0.00
LaSpezia	20	30	20746	20746	20746.0	0.03	20746	20746.0	0.02	0.00
LaSpezia	20	20	20746	20746	20746.0	0.03	20746	20746.0	0.02	0.00
LaSpezia	20	10	22811	22811	22811.0	0.12	22811	22811.0	0.03	0.00
BuenosAires	21	30	76999	76999	76999.0	0.03	76999	76999.0	0.06	0.00
BuenosAires	21	20	91619	91619	91619.2	4.20	91619	91619.0	0.47	0.00
Ottawa	21	30	16202	16202	16202.0	0.00	16202	16202.0	0.01	0.00
Ottawa	21	20	16202	16202	16202.0	0.00	16202	16202.0	0.01	0.00
Ottawa	21	10	17576	17576	17576.0	0.02	17576	17576.0	0.03	0.00
SanAntonio	23	30	22982	22982	22982.0	0.00	22982	22982.0	0.03	0.00
SanAntonio	23	20	24007	24007	24007.0	0.05	24007	24007.0	0.19	0.00
SanAntonio	23	10	40149	40149	40149.0	0.37	40149	40149.0	0.16	0.00
Brescia	27	30	30300	30300	30300.0	0.07	30300	30300.0	0.04	0.00
Brescia	27	20	31100	31100	31100.0	0.09	31100	31100.0	0.04	0.00
Brescia	27	11	35200	35200	35200.0	0.39	35200	35200.0	0.06	0.00
Roma	28	30	61900	61900	61900.0	0.36	61900	61900.0	0.06	0.00
Roma	28	20	66600	66600	66670.0	3.46	66600	66600.0	0.84	0.00
Roma	28	18	68300	68300	68300.0	0.00	68300	68300.0	0.10	0.00
Madison	28	30	29246	29246	29246.0	0.04	29246	29246.0	0.04	0.00
Madison	28	20	29839	29839	29839.0	0.04	29839	29839.0	0.04	0.00
Madison	28	10	33848	33848	33848.0	0.20	33848	33848.0	0.05	0.00
Guadalajara	41	30	57476	57476	57476.0	2.14	57476	57476.0	0.08	0.00
Guadalajara	41	20	59493	59493	59493.0	1.48	59493	59493.0	0.15	0.00
Guadalajara	41	11	64981	64981	64981.0	2.41	64981	64981.0	0.10	0.00
Dublin	45	30	33548	33548	33595.4	2.29	33548	33548.0	0.12	0.00
Dublin	45	20	39786	39786	39817.2	3.73	39786	39786.0	0.38	0.00
Dublin	45	11	54392	54392	55000.6	5.28	54392	54392.0	0.64	0.00
Avg.						0.66			0.10	0.00

The boxplot graphs in Fig. 4 compare the distribution and the range of the average results obtained with the two algorithms for medium and large instances. The average objective value f_{AVG} reported by each algorithm is normalized according to the following relation: $y = 100 * (f_{AVG} - f_{BK})/f_{BK}$, where f_{BK} is the updated best-known objective value (Tables 4–5). We perform the pairwise Wilcoxon-signed rank tests to determine whether there is a statistically significant performance difference between the two algorithms for each set of instances. The corresponding p-values are given in Fig. 4. A visible difference in the distributions of the average results further confirms the advantage of MA over DR on BRP.

Table 6 summarizes the statistical results reported by the two algorithms on the three sets of instances. The first row shows the number of new best-known solutions (optimal solutions for the small-instance set) found by each algorithm. The average percentage gap of the best/average result to the updated best-known result is provided in the second row. The last row provides the average computation time in seconds required by each algorithm to reach its best objective value. As shown in Table 6, we observe that MA significantly outperforms DR in terms of both the best and the average results. Furthermore, MA is nearly two to six times faster than DR in attaining the best solutions. These results thus indicate a highly competitive performance of MA with respect to the existing heuristic approach from the BRP literature.

Table 4
Computational results on medium-size instances.

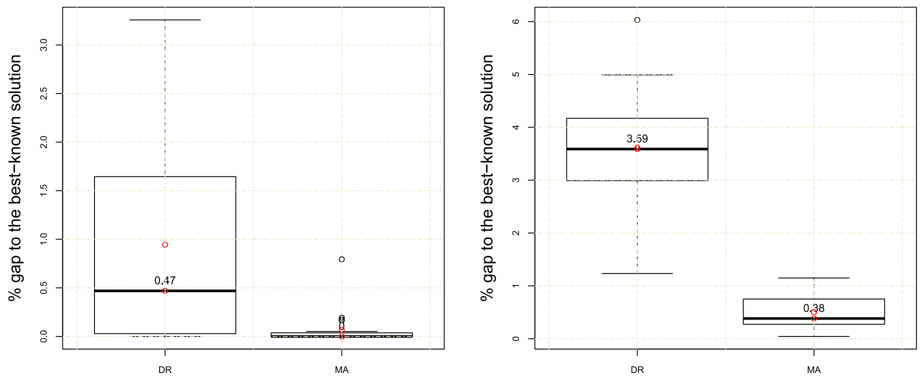
Instance	$\lvert V\rvert$	Q	DR (10 min)			MA (3 min)			
			f_{best}	f_{avg}	$t(s)$	f_{best}	f_{avg}	$t(s)$	gap
Denver	51	30	51583	51583.0	0.48	51583	51583.0	0.18	0.00
Denver	51	20	53465	53465.0	24.68	53465	53465.0	0.52	0.00
Denver	51	10	67459	67459.0	102.99	67459	67459.0	1.20	0.00
RiodeJ.	55	30	122547	122582.1	198.66	122547	122547.0	0.65	0.00
RiodeJ.	55	20	155517	155992.7	304.13	155517	155517.0	5.77	0.00
RiodeJ.	55	10	257147	257412.5	241.12	**257003**	257003.0	8.96	−0.06
Boston	59	30	65669	65669.0	16.16	65669	65669.0	0.34	0.00
Boston	59	20	71916	72057.2	178.16	**71879**	71879.0	24.30	−0.05
Boston	59	16	75085	75318.8	280.50	**75065**	75074.7	60.43	−0.03
Torino	75	30	47634	47634.0	246.44	47634	47634.0	1.78	0.00
Torino	75	20	50438	51026.0	251.83	**50204**	50204.0	23.15	−0.46
Torino	75	10	61717	62031.6	303.85	**61667**	61672.1	105.16	−0.08
Toronto	80	30	41390	41783.5	201.49	**41371**	41371.0	27.44	−0.05
Toronto	80	20	46631	46876.6	269.37	**45706**	45729.2	35.35	−1.98
Toronto	80	12	58539	58878.7	321.39	**57021**	57474.1	105.18	−2.59
Miami	82	30	154038	154344.7	335.05	**153624**	153639.8	60.39	−0.27
Miami	82	20	214250	215167.1	265.13	**211686**	211764.4	102.36	−1.20
Miami	82	10	397921	402746.8	300.28	**396109**	396109.2	80.12	−0.46
C.deMex.	90	30	72279	72797.5	213.91	**70812**	70949.3	103.35	−2.03
C.deMex.	90	20	94319	94621.6	254.22	**93243**	93405.0	112.03	−1.14
C.deMex.	90	17	103658	104213.7	359.59	**102508**	102677.1	68.18	−1.11
Avg.					222.35			44.14	−0.55

Table 5
Computational results on large-size instances.

Instance	$\lvert V\rvert$	Q	DR (30 min)			MA (10 min)			
			f_{best}	f_{avg}	$t(s)$	f_{best}	f_{avg}	$t(s)$	gap
Minneapolis	116	30	138467	139874.3	745.28	**137238**	137348.4	316.19	−0.89
Minneapolis	116	20	166150	166797.0	1003.72	**164764**	164835.4	351.51	−0.83
Minneapolis	116	10	262936	264335.2	946.37	**257698**	258178.7	409.44	−1.99
Brisbane	150	30	115120	115949.2	742.81	**113107**	113227.8	366.86	−1.75
Brisbane	150	20	146313	146930.0	957.83	**143509**	143813.3	284.40	−1.92
Brisbane	150	17	160015	160385.6	966.44	**155700**	156147.6	405.83	−2.70
Milano	184	30	167493	168931.2	871.18	**162733**	163798.6	399.87	−2.84
Milano	184	20	218249	219558.6	823.76	**213220**	214350.9	321.35	−2.30
Milano	184	18	234423	236394.9	1048.32	**229574**	230914.6	423.83	−2.07
Lille	200	30	176976	178133.5	666.67	**172235**	172902.7	257.81	−2.68
Lille	200	20	213090	215007.8	280.09	**204910**	206440.3	331.76	−3.84
Toulouse	240	30	188995	190146.8	1147.86	**182716**	183396.6	339.53	−3.32
Toulouse	240	20	228674	231062.0	1398.52	**221589**	222320.5	396.13	−3.10
Toulouse	240	13	307874	308982.7	868.98	**298194**	299852.1	335.83	−3.14
Sevilla	258	30	225076	227911.7	898.40	**217216**	217977.4	287.03	−3.49
Sevilla	258	20	279990	281492.2	1054.31	**271817**	272675.5	403.75	−2.92
Valencia	276	30	287854	292262.6	789.54	**283128**	283779.3	215.56	−1.64
Valencia	276	20	367201	370057.4	1064.40	**358119**	360820.2	397.66	−2.47
Bruxelles	304	30	311097	315487.7	851.97	**303733**	304880.9	454.69	−2.37
Bruxelles	304	20	376387	379141.4	835.66	**361580**	365018.4	333.49	−3.93
Bruxelles	304	16	424432	426992.4	1038.33	**411986**	415569.9	477.57	−2.93
Lyon	336	30	360009	364083.4	879.61	**346771**	350756.8	378.30	−3.68
Lyon	336	20	433959	437588.1	1136.03	**421511**	425429.0	388.40	−2.87
Barcelona	410	30	543929	545633.1	458.27	**527672**	530152.5	442.57	−2.99
Barcelona	410	20	771507	774818.4	1538.57	**746442**	748715.5	450.08	−3.25
Barcelona	410	19	800622	805513.2	1411.27	**778353**	780371.3	373.85	−2.78
London	564	30	699571	705232.0	1572.97	**673959**	680218.8	578.10	−3.66
London	564	29	718026	725468.0	1447.33	**684193**	692040.1	586.76	−4.71
Avg.					980.16			382.43	−2.75

Table 6
Statistical results (Note: the best performance is indicated in bold).

		DR	MA
Small-size instance set	Optimal solutions	41/41	**41/41**
	Average Gap_{best}/Gap_{avg}(%)	0.00/0.04	**0.00/0.00**
	Average time (s)	0.66	**0.10**
Medium-size instance set	Best-known solutions	7/21	**21/21**
	Average Gap_{best}/Gap_{avg}(%)	0.56/0.95	**0.00/0.07**
	Average time (s)	222.35	**44.14**
Large-size instance set	Best-known solutions	0/28	**28/28**
	Average Gap_{best}/Gap_{avg}(%)	2.84/3.61	**0.00/0.50**
	Average time (s)	980.16	**382.43**



(a) Medium-size instances(p-value=0.0004824) (b) Large-size instances (p-value=7.451e-09)

Fig. 4. Boxplots of the normalized average objective values for the medium-size and the large-size sets of instances.

4.4. Comparison with exact approach

To further investigate the performance of MA, we compare it with the branch-and-cut (B&C) algorithm (Dell'Amico et al., 2014) on the sets of medium and large instances. In Dell'Amico et al. (2014), the authors report for each medium and large instance the best upper and lower bound reached by B&C within a time limit of one hour. Tables 7–8 present the percentage gap between the best/average result obtained with MA and the lower/upper bound reported by B&C.

We observe that the gap between the best/average results and the lower bound by B&C ranges between 0.00% and 7.29% for the medium instances, implying that MA reports near-optimal or optimal solutions. Furthermore, MA reports an average improvement of the B&C solutions (upper bounds) by 3.68% on the medium instances and by 36.13% on the large instances. For some large instances, the improvement by MA compared to B&C can even exceed 50% highlighting the advantage of heuristics in case of BRPs.

4.5. Experimental results on the 1-PDVRPD instances

As the one-commodity pickup-and-delivery vehicle routing problem with distance constraints (1-PDVRPD) (Shi et al., 2009) is an extension of the generalized BRP with an additional constraint that limits the total distance traveled by each vehicle, we extend the proposed MA to tackle 1-PDVRPD. The 1-PDVRPD instances used in Shi et al. (2009) were derived from ten 1-PDTSP instances from Mladenović et al. (2012) by introducing the maximum distance constraints.

To evaluate the performance of MA on the 1-PDVRPD instances, we perform comparisons with two state-of-the-art 1-PDVRPD heuristics from the literature:

– A destroy-and-repair (DR) algorithm from Dell'Amico et al. (2016): The results reported by DR were obtained across ten executions for each instance on an Intel Core i3-2100 processor with 3.10 GHz and 4 GB RAM. The time limit for each run was set to 10 s for instances with $|V| \le 50$, 600 s for instances with $50 < |V| < 100$, and 1800 s for instances with $|V| \ge 100$.
– A genetic algorithm (GA) from Shi et al. (2009): The tests were carried out on a Pentium 1.6 GHz PC with 256 MB RAM. The stopping condition for GA was 100 generations for instances with $|V| < 100$ and 300 generations for instances with $|V| \ge 100$. However, the number of runs for each test instance in Shi et al. (2009) is not specified, and the best results are not reported.

We perform ten independent runs of MA for each test case, with the time limit per run set to 3 s for instances with $|V| < 100$ and 180 s for instances with $|V| \ge 100$ under the same experimental condition specified in Section 4.2.

Table 9 presents the results on the 1-PDVRPD benchmark. For each instance and approach, we show the best attained objective value (f_{best}), the average objective value (f_{avg}), and the average computing time in seconds (t_s) to reach the best result. The entries in bold correspond to the cases where MA improves on the best-known result from the literature.

Table 9 shows that MA outperforms both approaches on large instances with $|V| \ge 100$, while improving on the existing best-known objective values by up to 9%. Furthermore, MA seems to be considerably faster than DR despite the differences in the experimental platforms.

4.6. Experimental results on the M-M-VRPSPD instances

Recently, a new multi-commodity many-to-many vehicle routing problem with simultaneous pickup and delivery (M-M-VRPSPD) was proposed in Zhang et al. (2019) for a fast fashion retailer in Singapore. We adapt our MA to M-M-VRPSPD as the problem can be viewed as an extension of the generalized BRP by considering multiple types of bikes (commodities). For computational comparisons, we use from Zhang et al. (2019) a set of 22 instances consisting of 50 to 200 customers and 500 commodities, where the authors proposed an adaptive memory programming based algorithm (AMP) specifically designed for M-M-VRPSPD. The results reported by AMP were obtained after 10 executions per instance on a 2.8 GHz Intel Xeon E5-1603 CPU with 8 GB RAM, which has a similar performance to the machine that we use to run our experiments. Following Zhang et al. (2019), we perform 10 independent runs of MA per test case with the same stopping condition as that used by AMP. Namely, the stopping condition is the time limit set to $\lceil n/50 \rceil \times 300$ seconds, where n is the number of vertices in the given instance.

Table 10 reports the detailed results for the 22 M-M-VRPSPD instances, including the best (f_{best}) and the average objective value (f_{avg}), as well as the average computing time in seconds (t_s) to obtain the best result. We observe that our adaptation of MA competes favorably with AMP by improving 14 and matching 4 best-known results. In terms of the average objective values, MA exhibits a better performance on 15 out of the 22 instances. Despite the fact that our MA is specifically designed for the generalized BRP, we can thus conclude that its adaptation is highly effective for the M-M-VRPSPD problem.

4.7. Impact of crossover to the performance of MA

As shown in Section 4, our proposed population-based MA is very competitive with the existing algorithms from the literature. To determine whether such a performance is due to the memetic framework, namely the combination of the RCX operator (Section 3.5) and the ELS algorithm, we compare MA with a multi-start version of ELS (MELS)

Table 7
Comparison between MA and B&C on medium instances.

Instance	B&C			Best result of MA		Average result of MA	
	LB	UB	*Gap*	Gap_{LB}	Gap_{UB}	Gap_{LB}	Gap_{UB}
Denver	51583	51583	0.00	0.00	0.00	0.00	0.00
Denver	53465	53465	0.00	0.00	0.00	0.00	0.00
Denver	67459	67459	0.00	0.00	0.00	0.00	0.00
RiodeJ.	122547	122547	0.00	0.00	0.00	0.00	0.00
RiodeJ.	155446	156140	0.45	0.05	−0.40	0.05	−0.40
RiodeJ.	253690	259049	2.11	1.31	−0.79	1.31	−0.79
Boston	65669	65669	0.00	0.00	0.00	0.00	0.00
Boston	71879	71879	0.00	0.00	0.00	0.00	0.00
Boston	74790	75065	0.37	0.37	0.00	0.38	0.01
Torino	47634	47634	0.00	0.00	0.00	0.00	0.00
Torino	50204	50204	0.00	0.00	0.00	0.00	0.00
Torino	58814	64797	10.17	4.85	−4.83	4.86	−4.82
Toronto	40794	41549	1.85	1.41	−0.43	1.41	−0.43
Toronto	42621	47898	12.38	7.24	−4.58	7.29	−4.53
Toronto	54238	60763	12.03	5.13	−6.16	5.97	−5.41
Miami	152229	156104	2.55	0.92	−1.59	0.93	−1.58
Miami	209379	229237	9.48	1.10	−7.66	1.14	−7.62
Miami	390536	415762	6.46	1.43	−4.73	1.43	−4.73
C.deMex.	67894	88227	29.95	4.30	−19.74	4.50	−19.58
C.deMex.	88952	116418	30.88	4.82	−19.91	5.01	−19.77
C.deMex.	99714	109573	9.89	2.80	−6.45	2.97	−6.29
Avg.			6.12	1.70	−3.68	1.77	−3.62

Table 8
Comparison between MA and B&C on large instances.

Instance	B&C			Best result of MA		Average result of MA	
	LB	UB	*Gap*	Gap_{LB}	Gap_{UB}	Gap_{LB}	Gap_{UB}
Minneapolis	136148	137843	1.24	0.80	−0.44	0.88	−0.36
Minneapolis	157736	186449	18.20	4.46	−11.63	4.50	−11.59
Minneapolis	246133	298886	21.43	4.70	−13.78	4.89	−13.62
Brisbane	108275	158043	45.96	4.46	−28.43	4.57	−28.36
Brisbane	132419	196739	48.57	8.37	−27.06	8.60	−26.90
Brisbane	147236	234210	59.07	5.75	−33.52	6.05	−33.33
Milano	145245	227983	56.96	12.04	−28.62	12.77	−28.15
Milano	187175	295994	58.14	13.91	−27.96	14.52	−27.58
Milano	203716	299630	47.08	12.69	−23.38	13.35	−22.93
Lille	164149	231244	40.87	4.93	−25.52	5.33	−25.23
Lille	191630	440350	129.79	6.93	−53.47	7.73	−53.12
Toulouse	166653	404792	142.90	9.64	−54.86	10.05	−54.69
Toulouse	190739	427959	124.37	16.17	−48.22	16.56	−48.05
Toulouse	256036	461125	80.10	16.47	−35.33	17.11	−34.97
Sevilla	194805	461011	136.65	11.50	−52.88	11.90	−52.72
Sevilla	240210	516734	115.12	13.16	−47.40	13.52	−47.23
Valencia	245979	673479	173.80	15.10	−57.96	15.37	−57.86
Valencia	302368	698436	130.99	18.44	−48.73	19.33	−48.34
Bruxelles	255259	502920	97.02	18.99	−39.61	19.44	−39.38
Bruxelles	301100	580594	92.82	20.09	−37.72	21.23	−37.13
Bruxelles	348303	626721	79.94	18.28	−34.26	19.31	−33.69
Lyon	300950	580437	92.87	15.23	−40.26	16.55	−39.57
Lyon	356787	668971	87.50	18.14	−36.99	19.24	−36.41
Barcelona	311774	983627	215.49	69.25	−46.35	70.04	−46.10
Barcelona	449060	1088850	142.47	66.22	−31.45	66.73	−31.24
Barcelona	429327	1089070	153.67	81.30	−28.53	81.77	−28.35
London	385748	1304850	238.26	74.71	−48.35	76.34	−47.87
London	363299	1339890	268.81	88.33	−48.94	90.49	−48.35
Avg.			103.58	23.32	−36.13	23.86	−35.83

by running both algorithms under the same condition specified in Section 4.2.

Experiments were carried out on a selection of the ten large instances with $Q = 20$. The best and the average performances of the two algorithms are summarized in Table 11. The percentage gap between the two algorithms is defined as $gap = 100 \times \frac{f_{MA} - f_{MELS}}{f_{MA}}$. A negative gap indicates that MA outperforms MELS, and is outperformed otherwise.

Table 11 shows that MELS outperforms the proposed MA only on a single instance, whereas MA clearly dominates MELS on the remaining instances in terms of both the best and average performances. This comparison provides evidence that the integration of the RCX crossover within the memetic framework has a positive impact to the performance of MA.

5. Discussions

As mentioned in Section 2, heuristic and metaheuristic algorithms for BRP can be grouped into two main categories, namely the neighborhood-based local search algorithms (e.g., tabu search Chemla et al., 2013, iterated local search Cruz et al., 2017, and variable neighborhood search Rainer-Harbach et al., 2015) and the population-based evolutionary algorithms (e.g., genetic algorithm Li et al., 2016, and ant colony optimization Di Gaspero et al., 2013). As explained

Table 9
Comparison of MA with the two reference heuristics on the 1-PDVRPD instances.

Instance	$\|V\|$	Q	GA		DR			MA			
			f_{avg}	$t(s)$	f_{best}	f_{avg}	$t(s)$	f_{best}	f_{avg}	$t(s)$	gap
n20q10A	20	10	5515.4	0.65	5001	5001.0	0.10	5001	5001.0	0.04	0.00
n30q10A	30	10	6906.0	0.92	6503	6512.8	1.91	6503	6503.0	1.35	0.00
n40q10A	40	10	7476.4	1.14	7407	7474.8	5.60	7407	7460.0	4.78	0.00
n50q10A	50	10	9263.5	1.52	7283	7301.9	4.43	7283	7283.0	4.61	0.00
n60q10A	60	10	9931.6	2.01	8939	8941.1	140.87	8939	8939.0	8.15	0.00
n100q10A	100	10	14379.3	14.85	12317	12476.9	855.75	**12258**	12263.3	58.20	−0.48
n200q10A	200	10	23331.7	47.00	19341	19619.5	1181.46	**18801**	18889.6	119.51	−2.87
n300q10A	300	10	29805.3	100.25	24763	25092.3	808.87	**24017**	24304.1	130.20	−3.11
n400q10A	400	10	34574.4	156.47	33951	34209.0	1237.71	**32258**	32783.2	140.35	−5.25
n500q10A	500	10	39872.5	240.06	33108	33706.5	1203.76	**30127**	30612.7	148.55	−9.89

Table 10
Comparison of MA with AMP on the M-M-VRPSPD instances.

Instance	$\|V\|$	m	Q	AMP			MA			
				f_{best}	f_{avg}	$t(s)$	f_{best}	f_{avg}	$t(s)$	gap
301	50	500	4000	2189.62	2193.45	188.05	2189.62	2189.78	138.32	0.00
302	50	500	4000	2089.67	2095.69	156.68	2089.67	2097.80	124.62	0.00
303	75	500	3500	4217.16	4235.25	292.38	4217.16	4233.63	141.81	0.00
304	75	500	3500	3919.76	3958.92	286.63	3919.76	3956.71	113.60	0.00
305	100	500	5000	3447.87	3488.57	467.45	**3428.93**	3481.32	364.15	−0.55
306	100	500	5000	3370.53	3392.08	431.53	**3356.36**	3382.04	326.95	−0.42
307	100	500	5000	4135.17	4169.90	404.59	4148.15	4173.28	406.85	0.31
308	100	500	5000	4138.11	4197.07	423.54	**4131.84**	4196.56	323.14	−0.15
309	120	500	5000	3134.39	3213.00	775.29	**3126.95**	3203.69	674.30	−0.24
310	120	500	5000	3210.39	3278.83	646.03	**3207.77**	3275.15	540.77	−0.08
311	150	500	5000	5140.19	5175.61	748.54	**5124.90**	5172.08	638.32	−0.30
312	150	500	5000	4965.96	5030.60	724.88	**4954.27**	5021.05	753.22	−0.24
313	199	500	5000	7205.29	7369.95	989.20	7259.91	7393.37	1089.34	0.75
314	199	500	5000	6959.17	7040.47	1076.82	**6947.31**	7038.83	871.77	−0.17
315	100	500	5000	16069.18	16132.82	448.23	**16050.29**	16117.63	493.01	−0.12
316	100	500	5000	11218.59	11275.74	390.30	**11205.62**	11279.63	385.66	−0.12
317	100	500	17500	8062.61	8099.34	386.43	**8061.54**	8124.26	392.32	−0.01
318	100	500	25000	5274.22	5305.86	385.47	5286.16	5315.22	475.51	0.23
319	100	500	25000	3656.12	3677.89	375.57	3663.31	3670.59	368.54	0.20
320	200	500	5000	29259.04	29350.02	1052.59	**29244.68**	29347.29	1096.78	−0.05
321	200	500	17500	17266.41	17369.60	989.32	**17254.84**	17327.65	735.43	−0.07
322	200	500	25000	8394.92	8626.92	996.10	**8393.94**	8639.66	914.65	−0.01

Table 11
Comparison between MELS and MA.

Instance	$\|V\|$	Q	MELS		MA			
			f_{best}	f_{avg}	f_{best}	f_{avg}	gap_{best}	gap_{avg}
Minneapolis	116	20	164868	165001.6	164764	164835.4	−0.06	−0.10
Brisbane	150	20	143730	144089.3	143509	143813.3	−0.15	−0.19
Milano	184	20	214163	214759.6	213220	214350.9	−0.44	−0.19
Lille	200	20	205003	207246.7	204910	206440.3	−0.05	−0.39
Toulouse	240	20	222014	222913.5	221589	222320.5	−0.19	−0.27
Sevilla	258	20	272334	273028.0	271817	272675.5	−0.19	−0.13
Valencia	276	20	360857	362126.5	358119	360820.2	−0.76	−0.36
Bruxelles	304	20	360805	364626.3	361580	365018.4	0.21	0.11
Lyon	336	20	423157	426178.1	421511	425429.0	−0.39	−0.18
Barcelona	410	20	746986	749341.3	746442	748715.5	−0.07	−0.08

previously, MA combines the advantages of both the population-based global search and the neighborhood-based local search, which makes it highly competitive with the state-of-the-art methods from the literature. Specifically, the high performance of the proposed MA is due to the integration of two original components. First, MA employs a modified route-copy-based crossover operator for offspring solution generation. This crossover is specially designed for BRP to generate promising offspring that are structurally different from their parent solutions, which promotes the diversity of the population. As observed in Section 4.7, the used crossover significantly improves the search ability of the proposed MA. The second distinguishing feature of the proposed MA is an evolutionary local search (ELS) used for local improvement. It combines a memory-enhanced variable neighborhood descent with random neighborhood ordering (RVND) and an adaptive randomized mutation procedure. The distinguishing elements of the proposed ELS algorithm is a memory-based strategy used for the purpose of an early termination criterion of RVND and a probabilistic mechanism for an adaptive adjustment of the mutation strength.

In addition to the studied problem, we have adapted MA to tackle two similar problems (1-PDVRPD Shi et al., 2009 and M-M-VRPSPD Zhang et al., 2019) and observed very competitive performances on the benchmark instances. Therefore, the proposed algorithm is not only significant for the BSS operation management, but also contributes to the literature of other similar optimization problems.

Despite the good performance of the proposed approach, we believe that there still might be some space for improvement and future study. First, the neighborhood operators used for local search are overly simplistic. Other neighborhood operators proposed in the VRP or the

TSP literature could be considered. Second, the proposed method only considers the minimization of the total travel cost as objective. In practical bike-sharing systems, there exist various objectives, such as the minimization of the sum of travel and handling costs, minimization of the total number of loading and unloading quantities, etc.

6. Conclusions

The generalized BRP is an NP-hard problem derived from a real-world application. In this study, we presented the first population-based memetic algorithm (MA) to tackle the problem, which relies on a randomized greedy construction for initialization, an evolutionary local search procedure for offspring improvement using auxiliary data structures, and a route-copy-based crossover for generating diversified offspring solutions. Moreover, for an effective exploration of the search space, a short-term memory is employed during the evolutionary local search as an early termination mechanism, along with an adaptive probabilistic strategy for an adaptive adjustment of the mutation strength.

Extensive experiments on a set of 90 real-world benchmark instances show that the proposed MA outperforms the existing approaches in terms of both the solution quality and computing time. In particular, MA reports improved upper bounds for 42 medium and large instances while matching the best-known result for the remaining instances. Furthermore, the proposed algorithm is nearly two to six times faster in attaining the best solutions compared to the existing state-of-the-art heuristic. In addition to the problem studied in this paper, adaptations of the proposed MA have further shown to be very effective for the related 1-PDVRPD and M-M-VRPSPD problems. These studies reveal the usefulness of the MA framework for different variations of BRPs.

CRediT authorship contribution statement

Yongliang Lu: Conceptualization, Methodology, Software, Writing - original draft. **Una Benlic:** Writing - review & editing. **Qinghua Wu:** Validation, Supervision, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We are grateful to the anonymous referees for valuable suggestions and comments which helped us improve the paper. This work is supported by the National Natural Science Foundation of China [Grant No. 71771099, 71850410545, 71821001, 71810107003, 71620107002] and the China Postdoctoral Science Foundation [Grant No. 2019M662649].

References

Alvarez-Valdes, R., Belenguer, J.M., Benavent, E., Bermudez, J.D., Muñoz, F., Vercher, E., Verdejo, F., 2016. Optimizing the level of service quality of a bike-sharing system. Omega 62, 163–175.

Benchimol, M., Benchimol, P., Chappert, B., De La Taille, A., Laroche, F., Meunier, F., Robinet, L., 2011. Balancing the stations of a self service bike hire system. RAIRO Oper. Res. 45 (1), 37–61.

Berbeglia, G., Cordeau, J.-F., Gribkovskaia, I., Laporte, G., 2007. Static pickup and delivery problems: a classification scheme and survey. TOP 15 (1), 1–31.

Birattari, M., Yuan, Z., Balaprakash, P., Stützle, T., 2010. F-race and iterated F-race: An overview. In: Experimental Methods for the Analysis of Optimization Algorithms. Springer Berlin Heidelberg, pp. 311–336.

Bruck, B.P., Cruz, F., Iori, M., Subramanian, A., 2019. The static bike sharing rebalancing problem with forbidden temporary operations. Transp. Sci. 53 (3), 882–896.

Bulhões, T., Subramanian, A., Erdoğan, G., Laporte, G., 2018. The static bike relocation problem with multiple vehicles and visits. European J. Oper. Res. 264 (2), 508–523.

Cattaruzza, D., Absi, N., Feillet, D., Vidal, T., 2014. A memetic algorithm for the multi trip vehicle routing problem. European J. Oper. Res. 236 (3), 833–848.

Chemla, D., Meunier, F., Calvo, R.W., 2013. Bike sharing systems: Solving the static rebalancing problem. Discrete Optim. 10 (2), 120–146.

Contardo, C., Morency, C., Rousseau, L.M., 2012. Balancing a Dynamic Public Bike-Sharing System, Vol. 4. Cirrelt, Montreal, Canada.

Cruz, F., Subramanian, A., Bruck, B.P., Iori, M., 2017. A heuristic algorithm for a single vehicle static bike sharing rebalancing problem. Comput. Oper. Res. 79, 19–33.

Dell'Amico, M., Hadjicostantinou, E., Iori, M., Novellani, S., 2014. The bike sharing rebalancing problem: Mathematical formulations and benchmark instances. Omega 45, 7–19.

Dell'Amico, M., Iori, M., Novellani, S., Stützle, T., 2016. A destroy and repair algorithm for the bike sharing rebalancing problem. Comput. Oper. Res. 71, 149–162.

Dell'Amico, M., Iori, M., Novellani, S., Subramanian, A., 2018. The bike sharing rebalancing problem with stochastic demands. Transp. Res. B 118, 362–380.

Di Gaspero, L., Rendl, A., Urli, T., 2013. A hybrid ACO+ CP for balancing bicycle sharing systems. In: International Workshop on Hybrid Metaheuristics. Springer, Berlin, Heidelberg, pp. 198–212.

Duhamel, C., Lacomme, P., Prodhon, C., 2012. A hybrid evolutionary local search with depth first search split procedure for the heterogeneous vehicle routing problems. Eng. Appl. Artif. Intell. 25 (2), 345–358.

Duhamel, C., Lacomme, P., Quilliot, A., Toussaint, H., 2011. A multi-start evolutionary local search for the two-dimensional loading capacitated vehicle routing problem. Comput. Oper. Res. 38 (3), 617–640.

Erdoğan, G., Battarra, M., Calvo, R.W., 2015. An exact algorithm for the static rebalancing problem arising in bicycle sharing systems. European J. Oper. Res. 245 (3), 667–679.

Erdoğan, G., Laporte, G., Calvo, R.W., 2014. The static bicycle relocation problem with demand intervals. European J. Oper. Res. 238 (2), 451–457.

Espegren, H.M., Kristianslund, J., Andersson, H., Fagerholt, K., 2016. The static bicycle repositioning problem-literature survey and new formulation. In: International Conference on Computational Logistics. Springer, Cham, pp. 337–351.

Forma, I.A., Raviv, T., Tzur, M., 2015. A 3-step math heuristic for the static repositioning problem in bike-sharing systems. Transp. Res. B 71, 230–247.

Goldberg, D.E., 1989. Genetic Algorithm in Search, Optimization and Machine Learning, Vol. 13, No. 7. Addison Wesley, pp. 2104–2116.

Hao, J.K., 2012. Memetic algorithms in discrete optimization. In: Neri, et al. (Eds.), Handbook of Memetic Algorithms. Springer Berlin Heidelberg, pp. 73–94.

Ho, S.C., Szeto, W.Y., 2014. Solving a static repositioning problem in bike-sharing systems using iterated tabu search. Transp. Res. E: Logist. Transp. Rev. 69, 180–198.

Hosny, M.I., Mumford, C.L., 2009. Investigating genetic algorithms for solving the multiple vehicle pickup and delivery problem with time windows. In: Proceedings of the VIII Metaheuristics International Conference, Hamburg, Germany.

Kloimüllner, C., Papazek, P., Hu, B., Raidl, G.R., 2014. Balancing bicycle sharing systems: an approach for the dynamic case. In: European Conference on Evolutionary Computation in Combinatorial Optimization. Springer, Berlin, Heidelber, pp. 73–84.

Labadie, N., Melechovský, J., Calvo, R.W., 2011. Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. J. Heuristics 17 (6), 729–753.

Li, Y., Szeto, W.Y., Long, J., Shui, C.S., 2016. A multiple type bike repositioning problem. Transp. Res. B 90, 263–278.

Lin, J.H., Chou, T.C., 2012. A geo-aware and VRP-based public bicycle redistribution system. Int. J. Veh. Technol..

Lin, S., Kernighan, B.W., 1973. An effective heuristic algorithm for the traveling-salesman problem. Oper. Res. 21 (2), 498–516.

Lourenço, H.R., Martin, O.C., Stützle, T., 2003. Iterated local search. In: Handbook of Metaheuristics. Springer, Boston, MA, pp. 320–353.

Lu, Y., Benlic, U., Wu, Q., 2018. A hybrid dynamic programming and memetic algorithm to the traveling salesman problem with hotel selection. Comput. Oper. Res. 90, 193–207.

Lu, Y., Benlic, U., Wu, Q., Peng, B., 2019. Memetic algorithm for the multiple traveling repairman problem with profits. Eng. Appl. Artif. Intell. 80, 35–47.

Mladenović, N., Urošević, D., Ilić, A., 2012. A general variable neighborhood search for the one-commodity pickup-and-delivery travelling salesman problem. European J. Oper. Res. 220 (1), 270–285.

Nedic, N., Stojanovic, V., Djordjevic, V., 2015. Optimal control of hydraulically driven parallel robot platform based on firefly algorithm. Nonlinear Dynam. 82 (3), 1457–1473.

Neri, F., Cotta, C., 2012. Memetic algorithms and memetic computing optimization: A literature review. Swarm Evol. Comput. 2, 1–14.

Neri, F., Cotta, C., Moscato, P., 2012. Handbook of Memetic Algorithms. Springer Berlin Heidelberg.

Prins, C., Bouchenoua, S., 2005. A memetic algorithm solving the VRP, the CARP and general routing problems with nodes, edges and arcs. In: Recent Advances in Memetic Algorithms. Springer Berlin Heidelberg.

Rainer-Harbach, M., Papazek, P., Raidl, G.R., Hu, B., Kloimüllner, C., 2015. PILOT, GRASP, and VNS approaches for the static balancing of bicycle sharing systems. J. Global Optim. 63 (3), 597–629.

Raviv, T., Tzur, M., Forma, I.A., 2013. Static repositioning in a bike-sharing system: models and solution approaches. EURO J. Transp. Logist. 2 (3), 187–229.

Savino, M.M., Brun, A., Mazza, A., 2014a. Dynamic workforce allocation in a constrained flow shop with multi-agent system. Comput. Ind. 65 (6), 967–975.

Savino, M.M., Mazza, A., Neubert, G., 2014b. Agent-based flow-shop modelling in dynamic environment. Prod. Plan. Control 25 (2), 110–122.

Shi, X., Zhao, F., Gong, Y., 2009. Genetic algorithm for the one-commodity pickup-and-delivery vehicle routing problem. In: IEEE International Conference on Intelligent Computing and Intelligent Systems, Vol. 1. IEEE, pp. 175–179.

Stojanovic, V., Nedic, N., 2016. A nature inspired parameter tuning approach to cascade control for hydraulically driven parallel robot platform. J. Optim. Theory Appl. 168 (1), 332–347.

Stojanovic, V., Nedic, N., Prsic, D., Dubonjic, L., Djordjevic, V., 2016. Application of cuckoo search algorithm to constrained control problem of a parallel robot platform. Int. J. Adv. Manuf. Technol. 87 (9–12), 2497–2507.

Vaira, G., Kurasova, O., 2013. Genetic algorithms and VRP: the behaviour of a crossover operator. Baltic J. Mod. Comput. 1 (3–4), 161–185.

Wolf, S., Merz, P., 2007. Evolutionary local search for the super-peer selection problem and the p-hub median problem. In: International Workshop on Hybrid Metaheuristics. Springer, Berlin, Heidelberg, pp. 1–15.

Zhang, Z., Cheang, B., Li, C., Lim, A., 2019. Multi-commodity demand fulfillment via simultaneous pickup and delivery for a fast fashion retailer. Comput. Oper. Res. 103, 81–96.

Zhang, Z., Wei, L., Lim, A., 2015. An evolutionary local search for the capacitated vehicle routing problem minimizing fuel consumption under three-dimensional loading constraints. Transp. Res. B 82, 20–35.

Zhang, D., Yu, C., Desai, J., Lau, H.Y.K., Srivathsan, S., 2017. A time-space network flow approach to dynamic repositioning in bicycle sharing systems. Transp. Res. B 103, 188–207.