J Heuristics (2015) 21:391–431
DOI 10.1007/s10732-015-9282-5

A hybrid genetic algorithm with solution archive for the discrete $(r|p)$-centroid problem

Benjamin Biesinger · Bin Hu · Günther Raidl

Received: 3 May 2014 / Revised: 2 October 2014 / Accepted: 6 January 2015 /
Published online: 22 January 2015
© Springer Science+Business Media New York 2015

Abstract In this article we propose a hybrid genetic algorithm for the discrete $(r|p)$-centroid problem. We consider the competitive facility location problem where two non-cooperating companies enter a market sequentially and compete for market share. The first decision maker, called the leader, wants to maximize his market share knowing that a follower will enter the same market. Thus, for evaluating a leader's candidate solution, a corresponding follower's subproblem needs to be solved, and the overall problem therefore is a bi-level optimization problem. This problem is Σ_2^P-hard, i.e., harder than any problem in NP (if P $\neq$ NP). A heuristic approach is employed which is based on a genetic algorithm with tabu search as local improvement procedure and a complete solution archive. The archive is used to store and convert already visited solutions in order to avoid costly unnecessary re-evaluations. Different solution evaluation methods are combined into an effective multi-level evaluation scheme. The algorithm is tested on well-known benchmark sets of both Euclidean and non-Euclidean instances as well as on larger newly created instances. Especially on the Euclidean instances our algorithm is able to exceed previous state-of-the-art heuristic approaches in solution quality and running time in most cases.

Keywords Combinatorial optimization · Competitive facility location · Discrete $(r|p)$-centroid problem · Metaheuristics · Solution archive · Bi-level optimization

B. Biesinger (✉) · B. Hu · G. Raidl
Institute of Computer Graphics and Algorithms, Vienna University of Technology,
Favoritenstraße 9-11/1861, 1040 Vienna, Austria
e-mail: biesinger@ads.tuwien.ac.at

B. Hu
e-mail: hu@ads.tuwien.ac.at

G. Raidl
e-mail: raidl@ads.tuwien.ac.at

1 Introduction

The $(r|p)$-centroid problem (RPCP) is a competitive facility location problem, in which two decision makers compete for market share. They both want to serve customers from a given market. There are several variants of this problem which differ in the way facilities are opened, in the elasticity of the demand and especially in the behaviour of the customers. In our work we consider a discrete basic variant with the following assumptions:

- Facilities can be opened at a given finite set of possible positions. At one position at most one facility can be located.
- Both competitors open facilities sequentially, i.e., the first decision maker, called the leader, opens p facilities and is followed by the other decision maker, the follower, who successively places all of his r facilities.
- Customer decision is based on distance solely, i.e., customers always choose their serving facility to be the closest.
- Customers have a binary preference, i.e., they fulfill all of their demand by choosing the nearest facility only.

Competitive facility location problems are known since the late 20s and were first mentioned by Hotelling (1929). Hakimi (1983) introduced the name $(r|p)$-centroid and also published the first complexity results for it.

An application of this problem is when two entrepreneurs want to start selling the same products for the same price in a new market. The first one, the leader, who opens her selling facilities wants to keep as much of her market share as possible even when a competitor, the follower, enters the market. Since the leader cannot know where her competitor will place his facilities she assumes that he places them optimally. Considering that, the leader can determine her guaranteed (minimum) market share.

The discrete RPCP on not further constrained bipartite graphs is Σ_2^P-hard (Noltemeier et al. 2007). In short this means that under the assumption that the polynomial hierarchy does not collapse this problem is substantially harder to solve than any problem in NP.

In Sect. 2 we give a formal problem definition. After discussing the related work in Sect. 3, we present different candidate solution evaluation methods in Sect. 4. A novel hybrid genetic algorithm (GA), which incorporates a solution archive in order to store and transform already visited solutions, as well as a local improvement component is introduced in Sect. 5. Different concepts of how the local search method can benefit from the solution archive are investigated in Sect. 6. An extension to the GA which intends to lower the effort for solution evaluation by applying a multi-level strategy is presented in Sect. 7. Finally, we discuss computational results and compare the new method to other state-of-the-art approaches from literature in Sect. 8.

2 Problem definition

The discrete RPCP is defined on a weighted complete bipartite graph $G = (I, J, E)$ where $I = \{1, \dots, m\}$ represents the set of potential facility locations, $J = \{1, \dots, n\}$ the set of customers, and $E = I \times J$ is the set of edges. Let $w_j > 0, \forall j \in J$ be the weight of each customer, which corresponds to the turnover to be earned by the serving decision maker and d_{ij}, $\forall (i, j) \in E$, be the distances between customers and potential facility locations. The goal for the leader is to choose exactly p locations from I in order to maximize her turnover under the assumption that the follower in turn chooses r facilities from different locations maximizing his turnover.

Each customer $j \in J$ chooses the closest facility, hence the owner of this closest facility gains all of the turnover w_j; in case of equal distances the leader is preferred. In the following we give a formal definition of a candidate solution and the turnover computation. Let (X, Y) be a solution to the RPCP, where $X \subseteq I$, $|X| = p$ is the set of locations chosen by the leader and $Y \subseteq I \setminus X$, $|Y| = r$ is the associated set of follower locations. Further, let $D(j, V) = \min\{d_{ji} \mid i \in V\}$, $\forall j \in J, V \subseteq I$ be the minimum distance from customer j to all facility locations in set V. Then the set of customers which are served by one of the follower's facilities is $U^{\mathrm{f}} = \{j \in J \mid D(j, Y) < D(j, X)\}$ and the customers served by the leader is given by $U^{\mathrm{l}} = J \setminus U^{\mathrm{f}}$. The turnover of the follower is $p^{\mathrm{f}} = \sum_{j \in U^{\mathrm{f}}} w_j$ and the turnover of the leader $p^{\mathrm{l}} = \sum_{j \in J} w_j - p^{\mathrm{f}}$.

The problem of finding the optimal set of locations Y for the follower when X is given is also called the $(r|X_p)$-medianoid problem and is proven to be NP-hard (Hakimi 1983). Noltemeier et al. (2007) showed in their work that the $(r|p)$-centroid problem is Σ_2^P-hard. This result is strenghtened by Davydov et al. (2014b) who proved that the problem we consider here remains Σ_2^P-hard even for planar graphs with Euclidean distances.

3 Related work

Competitive facility location models, to which the discrete RPCP belongs, are a quite old and well-studied type of problem originally introduced by Hotelling (1929). A recent review about several kinds of competitive location models can be found in the work by Kress and Pesch (2012). They also mention the discrete RPCP, which was originally introduced by Hakimi (1983) along with some first complexity results.

Laporte and Benati (1994) developed a tabu search heuristic for the RPCP. They use an embedded second-level tabu search for solving the $(r|X_p)$-medianoid problem. The final solution quality is thus only approximated as the $(r|X_p)$-medianoid problem is not solved exactly.

Alekseeva et al. (2010) present several approaches for the discrete RPCP including an exact procedure. The first method is a hybrid memetic algorithm (HMA) which uses a probabilistic tabu search as local improvement procedure. It employs rather simple genetic operators and the tabu search utilizes a probabilistic swap neighborhood structure, which is well known from the p-median problem; see the review article by Mladenović et al. (2007) for an overview of this problem. A neighborhood of this structure contains elements only with a given probability to speed up the search. They use the linear programming relaxation of a mixed integer linear programming (MIP) model for the solution evaluation which will be described here in Sect. 4.1. The authors observe that this approach outperforms several simpler heuristics including an alternating heuristic originally proposed for a continuous variant of the problem

(Bhadury et al. 2003). In Alekseeva and Kochetov (2013) results for the tabu search alone are presented which are similar to the results of the HMA. They further describe an exact method based on a single level binary integer program with exponentially many constraints and variables. For solving this model they present an algorithm similar to a column generation approach where new sets of locations for the follower are iteratively added to the model which is then solved again. The optimal value of this model defines an upper bound and by solving the follower's problem using solutions of the model a lower bound is obtained. If the bounds coincide the optimum has been found. The HMA is applied for finding the initial family of follower solutions. Using this method the authors are able to optimally solve instances with up to 100 customers and $p = r = 5$.

Campos-Rodríguez et al. (2009) studied particle swarm optimization methods for the continuous RPCP, where the facilities can be placed anywhere on the Euclidean plane, as well as for the discrete variant (Campos-Rodríguez et al. 2012). A jumping particle swarm optimization is used with two swarms, one for the leader and one for the follower. The particles jump from one solution to another in dependence of its own best position so far, the best position of its neighborhood and the best position obtained by all particles so far, i.e, the best global position. In the experiments this algorithm was able to solve instances with 25 customers, $p = 3$ and $r = 2$ to optimality.

Davydov et al. (2012) describe another tabu search for the RPCP. They use a probabilistic swap neighborhood structure similar to the one developed by Alekseeva et al. (2010). For the solution evaluation the follower problem is approximately solved by Lagrangian relaxation. The method is tested on the instances from Alekseeva et al. (2010) and additionally on some non-Euclidean instances. For many of the instances optimal solutions are obtained.

A recent article by Davydov et al. (2014a) proposes two metaheuristics which are both based on the swap neighborhood structure. The first one uses a variable neighborhood search (VNS) with a disjoint partitioning of the swap neighborhood structure into three sub-neighborhoods *Fswap*, *Nswap*, and *Cswap*. A neighbor from the *Fswap* neighborhood is determined by closing one leader facility and opening a facility at a location chosen by the follower. The *Nswap* neighborhood consists of solution candidates that are generated by closing one leader facility and opening a facility at a location in its vicinity. *Cswap* consists of all solutions that are in the swap neighborhood but not already in *Fswap* or *Nswap*. The second method, which is called STS, uses the probabilistic swap neighborhood which has been proposed by Alekseeva et al. (2010). The STS uses the same neighborhood partioning as the VNS. Additionally, a tabu list is maintained to remove elements of the neighborhood which consist of pairs for leader facilities that have been closed and opened during the last few iterations. Both methods use the same model for the solution evaluation as Alekseeva et al. (2010). The authors were able to find good solutions for many instances faster than the tabu search by Davydov et al. (2012). Moreover, they tested their algorithms on non-euclidean instances, on which both methods, VNS and STS, showed similar performance.

Roboredo and Pessoa (2013) developed an exact branch-and-cut algorithm for the discrete RPCP. They use a single-level integer programming model which is similar to the model by Alekseeva et al. (2010) but with only a polynomial number of variables.

It consists of exponentially many constraints, one for each follower strategy, i.e., for each set of possible facility locations of the follower. An important reason for the success of their method is the introduction of strengthening inequalities by lifting the exponentially many constraints. Due to the assumption that the customers are conservative the lower bound on the leader's solution becomes zero if the follower chooses the same facility location. Therefore, for each facility location an alternative location is given which is chosen if the position has already been used by the leader. These cuts are separated either by a greedy heuristic or by solving a mixed integer programming model. For most of the benchmark instances the authors report better results than Alekseeva et al. (2010), i.e., they found optimal solutions in less time. Instances with 100 customers and up to $r = p = 15$ facilities could be solved to optimality. The authors also present promising results for $r = p = 20$ but are not able to prove their optimality within the given time limit of 10 hours.

Alekseeva and Kochetov (2013) give an overview of recent research regarding the discrete RPCP. They also improve their iterative exact method by using a model with only a polynomial number of variables and by using the strengthening inequalities introduced by Roboredo and Pessoa. This improved iterative approach is able to find optimal solutions for instances with up to 100 customers and $r = p = 15$. Especially for the instances with $r = p \in \{5, 10\}$ optimal solutions are found significantly faster than by the branch-and-cut algorithm from Roboredo and Pessoa (2013).

As mentioned before there are several other variants of the problem. Kochetov et al. (2013) describe an algorithm for the RPCP with fixed costs for opening a facility and customers splitting their demand over all facilities proportional to attraction factors. The algorithm's principle is similar to the alternating heuristic for the RPCP (Bhadury et al. 2003). Another work which assumes proportional splitting of the demands is done by Biesinger et al. (2014). There the authors employ a GA in combination with a solution archive similar to this work to solve the problem.

Ghosh and Craig (1984) work on a competitive location model for choosing good positions for retail convenience stores. They do not specify the number of stores to open beforehand but determine the optimal number as part of the process and additionally assume elastic demands, i.e., the customers may not want to fulfill their whole demand if the store is too far away. The authors develop a heuristic enumerative search algorithm for solving this problem.

Serra and Revelle (1994) propose a heuristic approach for a variant of the discrete RPCP which is based on repeatedly solving a maximum capture (MAXCAP) problem. The MAXCAP problem is similar to the $(r|X_p)$-medianoid problem with the difference that it is possible to place a facility on one of the leader's locations with the result that the captured demand is equally shared between the two players. The algorithm is basically a local search using the swap neighborhood structure and candidate solutions are evaluated by solving the MAXCAP problem by means of integer programming or by using a local search heuristic for larger instances.

Drezner (1994, 1998) and Drezner et al. (2002) use a gravity model for solving a continuous competitive facility location problem. The gravity model assumes that customers prefer being served by facilities proportional to an attraction factor and inversely proportional to their distances. The authors suggest several heuristics and metaheuristics including simulated annealing for solving the problem and compare them to each other.

4 Solution evaluation

We extend the problem definition of Sect. 2 by the following further definitions which are adopted from Alekseeva and Kochetov (2013).

Definition 1 Semi-feasible Solution
The tuple (X, Y) is called a *semi-feasible solution* to the discrete RPCP iff $X \subseteq I$ with $|X| = p$, $Y \subseteq I$ with $|Y| = r$ and $X \cap Y = \emptyset$.

Let $p^{\mathrm{l}}(X, Y)$ be the turnover of the leader and $p^{\mathrm{f}}(X, Y)$ be the turnover of the follower where X is the set of facility locations chosen by the leader and Y is the set of facility locations chosen by the follower. Then we define a feasible solution and an optimal solution as follows.

Definition 2 Feasible Solution
A semi-feasible solution (X, Y^*) is called a *feasible solution* to the discrete RPCP iff $p^{\mathrm{f}}(X, Y^*) \geq p^{\mathrm{f}}(X, Y)$ for each possible set of follower locations Y.

Definition 3 Optimal Solution
A feasible solution (X^*, Y^*) is called an *optimal solution* to the discrete RPCP iff $p^{\mathrm{l}}(X^*, Y^*) \geq p^{\mathrm{l}}(X, Y)$ for each feasible solution (X, Y).

It is easy to find a semi-feasible solution but already NP-hard to find a feasible solution because an optimal follower solution has to be found and the $(r|X_p)$-medianoid problem is NP-hard. This means that the solution evaluation of an arbitrary leader solution might be quite time-consuming. For practice there are several possibilities how to evaluate such a leader solution X. In the next section we will examine a mathematical model for the RPCP and derive different options.

4.1 Bi-Level MIP Formulation

The following bi-level MIP model has been introduced in Alekseeva et al. (2009). It uses three types of binary decision variables. Variables $x_i, \forall i \in I$, are set to one if facility i is opened by the leader and to zero otherwise. Variables $y_i, \forall i \in I$, are set to one iff facility i is opened by the follower. Finally, variables $z_j, \forall j \in J$, are set to one if customer j is served by the leader and set to zero if customer j is served by the follower.

We further define the set of facilities that allow the follower to capture customer j if the leader uses solution $x (x = (x_i)_{i \in I})$:

$$I_j(x) = \left\{ i \in I \mid d_{ij} < \min_{l \in I | x_l = 1} d_{lj} \right\} \quad \forall j \in J$$

Then we can define the upper level problem, denoted as leader's (or $(r|p)$-centroid) problem, as follows:

$$\max_x \sum_{j \in J} w_j z_j^* \tag{1}$$

s.t.

$$\sum_{i \in I} x_i = p \tag{2}$$

$$x_i \in \{0, 1\} \qquad \forall i \in I \tag{3}$$

where z^* is an optimal solution to the lower level problem, denoted as follower's (or $(r|X_p)$-medianoid) problem:

$$\max_{y,z} \sum_{j \in J} w_j (1 - z_j) \tag{4}$$

s.t.

$$\sum_{i \in I} y_i = r \tag{5}$$

$$1 - z_j \leq \sum_{i \in I_j(x)} y_i \qquad \forall j \in J \tag{6}$$

$$x_i + y_i \leq 1 \qquad \forall i \in I \tag{7}$$

$$z_j \geq 0 \qquad \forall j \in J \tag{8}$$

$$y_i \in \{0, 1\} \qquad \forall i \in I, \quad \forall j \in J \tag{9}$$

The objective function for the leader's problem (1) maximizes the leader's turnover. Equation (2) ensures that the leader places exactly p facilities. The objective function for the follower's problem (4) maximizes the follower's turnover. Similarly as in the leader problem, (5) ensures that the follower places exactly r facilities. Inequalities (6) together with the objective function ensure the z_j variables to be set correctly, i.e., decide for each customer $j \in J$ from which competitor he is served. Inequalities (7) guarantee that the follower does not choose a location where the leader has already opened a facility. Variables z_j are not restricted to binary values because in an optimal solution they will become 0 or 1 anyway.

For the simplicity of notation, we use in the remaining paper both, the set notation X and Y as well as the incidence vector notation x and y, to refer to leader and follower solutions, respectively.

4.2 Solution evaluation methods

In our metaheuristic approach we consider the following natural ways of evaluating a leader solution x; two of them use the model introduced before.

4.2.1 Exact evaluation

In the exact evaluation we solve the follower's problem (4–9) exactly using a MIP solver. In our implementation we used the IBM ILOG CPLEX Optimizer in version 12.5.

4.2.2 Linear programming (LP) evaluation

In the LP evaluation we solve the LP relaxation of the follower's problem exactly using CPLEX. This will in general yield not even semi-feasible solutions because of fractional values of some variables. For intermediate solution candidates we might, however, only be interested in an approximate objective value of a leader's solution for which purpose this method may be sufficient. This approximation yields a lower bound of the real objective value of x.

4.2.3 Greedy evaluation

In the greedy evaluation we use the following greedy algorithm for solving the follower's problem, which will yield semi-feasible solutions and therefore an upper bound to the objective value of x. Follower facilities are placed one after the other according to the following greedy criterion: For each facility every possible remaining location is checked how much turnover gain its selection would generate. The turnover gain is the sum of weights of all customers that would so far be served by the leader but are nearer to this location than to the nearest leader facility. Then the location with the maximum turnover gain is always selected. This process is iterated until r locations are chosen. Ties are broken randomly.

In Sect. 8.1 we will observe that among our evaluation algorithms LP evaluation usually offers the best compromise in terms of speed and evaluation precision. However, by applying the different solution evaluation methods in a joined way within a multi-level evaluation scheme described in Sect. 7, we will be able to significantly improve the performance.

5 Genetic algorithm with solution archive

This section describes our genetic algorithm. The framework is a rather standard steady-state GA with an embedded local improvement. It uses simple genetic operators, which are explained in Sect. 5.1. The local improvement procedure is based on the swap neighborhood structure and is addressed in Sect. 5.2. Most importantly, the GA utilizes a complete solution archive for duplicate detection and conversion, which is detailed in Sect. 5.3.

We use the leader's incidence vector x as solution representation for the GA. The initial population is generated by choosing p locations uniformly at random to ensure high diversity in the beginning. Then, in each GA iteration one new solution is derived and always replaces the worst solution of the current population. Selecting parents for crossover is performed by binary tournament selection with replacement. Mutation is applied to offsprings with a certain probability in each iteration.

5.1 Variation operators

We use the following variation operators within the GA:

Crossover operator Suppose that we have two candidate solutions $X^1 \subset I$ and $X^2 \subset I$. An offspring X' is derived from its parents X^1 and X^2 by adopting all locations that are opened in both, i.e., all locations from $S = X^1 \cap X^2$ and then choosing the remaining $p - |X^1 \cap X^2|$ locations from $(X^1 \cup X^2) \setminus S$, i.e., the set of locations that are opened in exactly one of the parents, uniformly at random.

Mutation operator Mutation is based on the swap neighborhood structure, which is also known from the p-median problem (Mladenović et al. 2007). A swap move closes a facility and re-opens it at a different, so far unoccupied position. Our mutation applies μ random swap moves, where μ is determined anew at each GA-iteration by a random sample from a Poisson distribution with mean value one so that each position is mutated independently with probability $\frac{1}{p}$.

5.2 Local search

Each new candidate solution derived in the GA via recombination and mutation whose objective value is at most $\alpha\%$ off the so far best solution value further undergoes a local improvement, with $\alpha = 5$ in our experiments presented here. Local search (LS) is applied with the swap neighborhood structure already used for mutation. The best improvement step function is used, so all neighbors of a solution that are reachable via one swap move are evaluated and a best one is selected for the next iteration. This procedure terminates with a local optimal solution when no superior neighbor can be found.

5.3 Solution archive

Solution archives for evolutionary algorithms as introduced by Raidl and Hu (2010) are data structures that efficiently store all generated solutions in order to be able to detect duplicate solution when they occur. Upon detecting a duplicate, an effective solution conversion is performed, which results in a guaranteed not yet considered solution typically in close proximity to the original (duplicate) solution. Especially when the solution evaluation is expensive, which is the case for the RPCP at least when performing an exact evaluation, costly and unnecessary re-evaluations are avoided supposedly resulting in an overall faster optimization. Additionally, diversity is maintained in the population and premature convergence is reduced or avoided as well. Successful applications of this concept on various test functions including NK landscapes and Royal Road functions and the generalized minimum spanning tree problem can be found in Raidl and Hu (2010) and Hu and Raidl (2012), respectively.

After each iteration of the genetic algorithm the newly created offspring is inserted into the archive. If this solution is already contained in the archive, the solution conversion is automatically performed and this adapted and guaranteed new solution is integrated in the population of the GA. The conversion operation can therefore also be considered as "intelligent mutation". The data structure used for the solution archive must be carefully selected in order to allow efficient realizations of the essential insert, look-up and conversion operations and in particular depends on the solution representation. As suggested in Raidl and Hu (2010) a trie data structure, which is typically used for storing a large set of strings (Gusfield 1997) like in language dictionary applications, is particularly well suited for binary representations because all the essential operations can be performed in $\mathcal{O}(h)$ time, where h is the maximum string length. In our case for the RPCP the insertion and the conversion procedure run both in $\mathcal{O}(m)$ time and almost independent of the number of created / stored solutions. The next sections describe the trie data structure and the specific operations of the solution archive we use for the RPCP in detail.

5.4 Trie structure

Our trie is a binary tree T with maximum height m, the number of possible facility locations, see Fig. 1. On each level $l = 1, \ldots, m$ of the trie there exist at most 2^{l-1} trie nodes, denoted by a rectangle in Fig. 1. Each trie node q at level l has the same

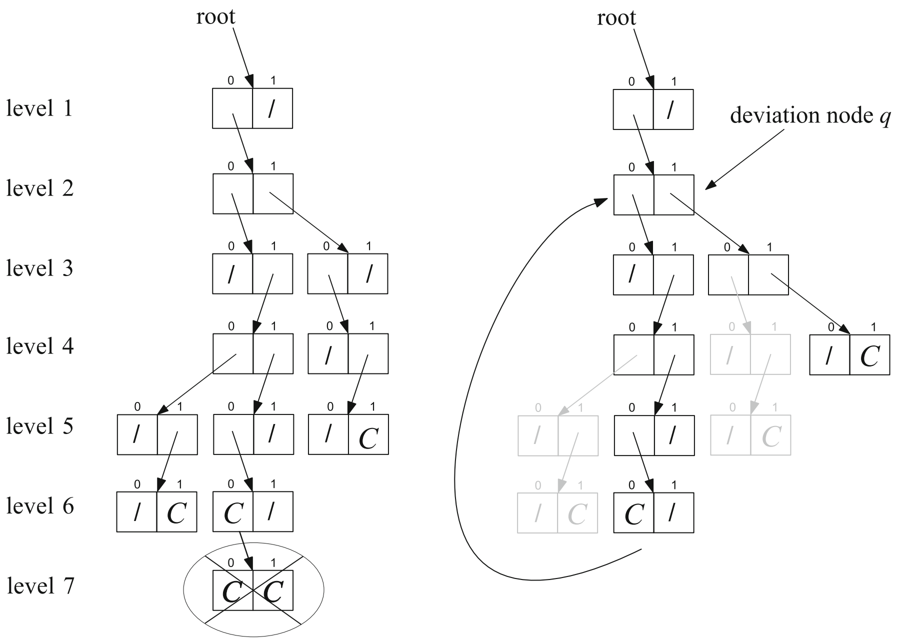

Fig. 1 Solution archive with some inserted solutions on the *lefthand side* and a conversion of (0, 0, 1, 1, 0, 0, 1) into the new solution (0, 1, 1, 1, 0, 0, 0) on the *righthand side*

Algorithm 1: insert($x, l, q, openFacs$)

```
Global Variable:
$devpoints = \emptyset$ //Set of feasible deviation positions for conversion
Input: leader solution $x$, level $l$, node $q$,
        int $openFacs$ //Number of facilities opened until level $l$
Output: boolean value whether or not $x$ is already contained in the archive
$alreadyContained = false$;
if $l \leq m \wedge q \neq complete \wedge openFacs < p$ then
    if $x_l == 1$ then
        if $m - l < p - openFacs$ then
            $q$.next[0] = $complete$;
        $openFacs = openFacs + 1$;
    if $q$.next[$1 - x_l$] $\neq complete$ then
        $devpoints = devpoints \cup \{(l, p)\}$
    if $q$.next[$x_l$] $== null$ then
        $q$.next[$x_l$] = new trienode($null, null$);
    $alreadyContained$ = insert($x, l + 1, q$.next[$x_l$]$, openFacs$);
if $q == complete$ then
    $alreadyContained = true$;
else if $l > m$ then
    $q = complete$;
//Pruning
else if $q$.next[$x_l$] $= complete \wedge q$.next[$1 - x_l$] $= complete$ then
    $q = complete$;
return $alreadyContained$;
```

structure consisting of two entries q.next[0] and q.next[1], denoted by the division of a rectangle into two parts. Each entry can be either a pointer to a subtree rooted at a successor node on level $l + 1$ (denoted by an arrow), a *null*-pointer (denoted by a slash), or a *complete*-pointer (denoted by a C).

Let $x = (x_1, \ldots, x_m)$ be the binary vector representing a candidate leader solution. Then each node q at level l is related to variable x_l and the entries q.next[0] and q.next[1] split the solution space into two subspaces with $x_l = 0$ and $x_l = 1$, respectively. In both subspaces all elements from x_1 to x_{l-1} are fixed according to the path from the root to node q. For example, in Fig. 1 the node on level 2 separates the solution space into solution candidates starting with $(0, 0)$ and $(0, 1)$. A *null*-pointer represents a yet completely unexplored subspace, while a *complete*-pointer denotes that already all solutions of the corresponding subspace have been considered. Note that such a trie is somewhat related to an explicitly stored branch-and-bound tree.

5.5 Insertion

Algorithm 1 shows how to insert a new candidate solution $x = (x_1, \ldots, x_m)$ into the trie. Initially, the recursive insertion method is called with parameters $(x, 1, root, 0)$. We start at the root node at level 1 with the first element x_1. At each level $l = 1, \ldots, m$ of the trie we follow the pointer indexed by x_l. When the p-th facility has been encountered, i.e., $openFacs = p$, at some node q the procedure stops and we set q.next[1] to *complete*. We further check at each insertion of a "one" at trie node q if enough facilities would still fit if instead a zero would be chosen. If this is not the case, q.next[0] is set to *complete* to indicate that there is no valid candidate solution in this subtrie. A set of feasible deviation positions, *devpoints*, is computed during the insertion and needed for the potentially following conversion. This set is cleared at the beginning of each solution insertion and contains all trie nodes visited during insertion where both entries are not *complete*. When we encounter a *complete*-pointer we know that this solution is already contained in the trie and it must be converted.

If we are finished with the insertion and the solution is not a duplicate, we prune the trie if possible to reduce its memory consumption. Pruning is performed by checking all trie nodes that have been visited during insertion bottom up if both entries of a trie node q are set to *complete*. If q.next[0] = q.next[1] = *complete* we prune this trie node by setting the corresponding entry of the preceding trie node to *complete*. On the left-hand side of Fig. 1 an example of a trie containing the three solutions (0, 0, 1, 1, 0, 0, 1), (0, 1, 0, 1, 1, 0, 0), and (0, 0, 1, 0, 1, 1, 0) is given. The "*C*" stands for a *complete*-pointer and the "/" for a *null*-pointer. The crossed out node at level 7 is a demonstration of setting a "zero" entry to *complete* because no more feasible solution fits in this subtrie and of the pruning that followed.

Note that no explicit look-up procedure is needed because the insertion method sketched in Algorithm 1 integrates the functionality to check whether or not a candidate solution is already contained.

5.6 Conversion

Algorithm 2: convert(x, $devpoints$)

```
Input: duplicate leader solution $x$, feasible deviation positions $devpoints$
Output: converted not yet considered solution $x$
$q$ = random entry from $devpoints$
$l$ = level of the trie node $q$
$x_l = 1 - x_l$;
while $q.next[x_l] \neq null$ do
    if $q.next[x_l] == complete$ then
        $x_l = 1 - x_l$;
    if $q.next[x_l] == null$ then
        break;
    $q = q.next[x_l]$;
    $l = l + 1$;
$openFacs$ = number of facilities opened in $x$
$k = p - openFacs$;
if $k > 0$ then
    open $k$ facilities among $x_{l+1}, \ldots, x_m$ randomly
else if $k < 0$ then
    close $|k|$ facilities among $x_{l+1}, \ldots, x_m$ randomly
insert($x$,$l$,$q$,$openFacs$);
return $x$;
```

When the insertion procedure detects a solution which is already contained in the archive, a conversion into a new solution is performed. A pseudocode of this procedure is given in Algorithm 2. In order to modify a solution, we have to apply at least two changes: open a facility and close another one. For the first change, let *devpoints* denote the set of feasible deviation points computed during insertion. A trie node q at level l is chosen from this set uniformly at random. Should this set be empty, we know that the whole search space has been covered and we can stop the optimization process with the so far best solution being a proven optimum. Otherwise we set the l-th element of the solution vector to $1 - x_l$, which corresponds to opening or closing a facility at position l. Now we have to apply a second (inverse) change at a later position in order to have exactly p facilities opened. We go down the subtrie level by level using the following strategy. For each trie node q' at level l' we prefer to follow the original solution, i.e., the pointer $q'.\text{next}[x_{l'}]$. If it is complete, we have no choice but to use the pointer $q'.\text{next}[1 - x_{l'}]$ instead (which corresponds to adding further modifications to the solution vector). As soon as we reach a *null*-pointer at a trie node q' at level l', we know that the corresponding subspace has not been explored yet, i.e., any feasible solution from this point on is a new one. Therefore, we apply the remaining necessary changes to get a feasible solution. If the number of opened facilities in x exceeds p, we close the appropriate number of facilities randomly among $\{x_{l'+1}, \ldots, x_m\}$. Otherwise, if this number is smaller than p, we open the appropriate number of facilities analogously. Finally, this new solution is inserted by applying Algorithm 1 starting from trie node q' at level l'.

On the righthand side of Fig. 1 an example of a solution conversion is shown. The duplicate solution $x = (0, 0, 1, 1, 0, 0, 1)$ is inserted into the trie and subsequently converted. Node q on level 2 is chosen as the deviation point for the first change and we set $x_2 = 1$, resulting in solution $(0, 1, 1, 1, 0, 0, 1)$. Since the alternative entry at $q.\text{next}[1]$ points to another trie node, this path is followed until a *null*-pointer is reached at level 3. Then we close the facility at the randomly chosen position 7 to get the valid solution (0, 1, 1, 1, 0, 0, 0).

5.7 Randomization of the trie

The above conversion procedure can only change values of solution elements with a greater index than the level of the deviation position. This induces an undesirable bias towards elements on positions with higher indices being changed more likely. In order to counter this problem, a technique called trie randomization is employed, which has first been suggested by Raidl and Hu (2010). For each search path of the trie we use a different ordering of the solution variables, i.e., a trie node on level l does not necessarily correspond to element x_l of the solution vector. Instead, the index of the element related to a trie node q is chosen randomly from the indices not already used in the path from the root to node q. In our case this is achieved by additionally storing the corresponding variable index at each trie node. Another possibility is to compute the next index by a deterministic pseudo random function taking the path from the root to node q as input. This method saves memory but needs more computational effort and is applied in Raidl and Hu (2010). Figure 2 shows an example of a randomized

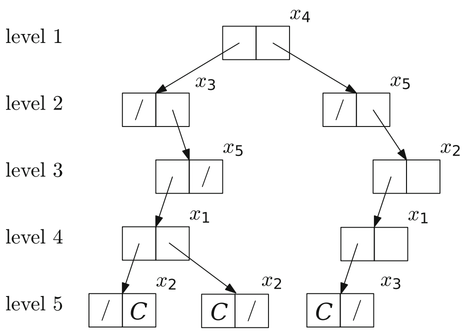

Fig. 2 Candidate solutions (0,1,1,0,0), (1,0,1,0,0), and (0,0,0,1,1) in a randomized trie, where the variables are randomly associated with the levels

trie. Although this technique cannot avoid biasing completely, the negative effect is substantially reduced.

6 Local and tabu search with solution archive

There exist several options for possibly utilizing the archive not just within the GA but also the embedded LS, based on the original swap neighborhood structure.

6.1 Complete neighborhood

The simplest way to perform LS is just to use the complete neighborhood as introduced in Sect. 5.2 without considering the solution archive. This method will find the best solution within the swap neighborhood but there is no benefit from the solution archive. We have to re-evaluate already visited solutions within the LS. However, all generated solutions during the LS are inserted into the solution archive so that the variation operators of the GA are still guaranteed to produce only not yet considered solution candidates.

6.2 Reduced neighborhood

The second option is to skip already visited solutions in the neighborhood search. After each swap it is checked if the new solution is already contained in the solution archive. If this is the case the evaluation of this solution is skipped and the LS continues with the next swap. Otherwise this solution is inserted into the solution archive. The advantage of this method is that re-evaluations of already generated solutions are completely avoided and the neighborhoods are usually much smaller, resulting in a lower runtime. A downside is, however, that due to the reduced neighborhoods LS may terminate with worse solutions that are not local optimal anymore.

6.3 Conversion neighborhood

Another possibility for a combination of the local search and the solution archive is to perform a conversion whenever an already visited solution is generated by the local search. This implies that the size of this neighborhood is the same as the complete neighborhood but instead of re-evaluating duplicates, solutions that are farther away are considered to possibly find a better solution.

6.4 Tabu search

The fourth method we consider uses a tabu search instead of a local search where the tabu list is realized by the solution archive. This means in particular that the search is not stopped when a neighborhood does not contain a better solution but a best neighbor solution that has not been visited, even when worse than the current solution, is always accepted and the search continues. In this way, the algorithm might escape local optima. This strategy can be combined with either of the latter two methods. Unlike the LS, since there is no predefined end of the tabu search, an explicit termination criterion is needed, e.g., a time limit or a number of iterations without improvement. As final solution, the best one encountered during the whole tabu search is returned.

7 Multi-level solution evaluation scheme

In this section we want to exploit several relationships between the solution values of the different evaluation methods which are described in Sect. 4. Suppose that $p^{\mathrm{f}}_{\mathrm{LP}}(x)$ is the objective value of the follower's problem obtained by LP evaluation for a given leader solution x, $p^{\mathrm{f}}_{\mathrm{exact}}(x)$ is the objective value obtained by exact (MIP-based) evaluation and $p^{\mathrm{f}}_{\mathrm{greedy}}(x)$ is the objective value of the follower's problem when using the greedy evaluation. Then $p^{\mathrm{f}}_{\mathrm{LP}}(x)$ is obviously an upper bound and $p^{\mathrm{f}}_{\mathrm{greedy}}(x)$ a lower bound to $p^{\mathrm{f}}_{\mathrm{exact}}(x)$, i.e., the following relations hold:

$$p^{\mathrm{f}}_{\mathrm{greedy}}(x) \leq p^{\mathrm{f}}_{\mathrm{exact}}(x) \leq p^{\mathrm{f}}_{\mathrm{LP}}(x) \tag{10}$$

Since we compute the turnover of the leader by subtracting the turnover of the follower from the total demand for all customers, i.e.,

$$\begin{aligned} p^{\mathrm{l}}_{\mathrm{LP}}(x) &= \sum_{j \in J} w_j - p^{\mathrm{f}}_{\mathrm{LP}}(x), \\ p^{\mathrm{l}}_{\mathrm{exact}}(x) &= \sum_{j \in J} w_j - p^{\mathrm{f}}_{\mathrm{exact}}(x), \\ p^{\mathrm{l}}_{\mathrm{greedy}}(x) &= \sum_{j \in J} w_j - p^{\mathrm{f}}_{\mathrm{greedy}}(x), \end{aligned}$$

we obtain:

$$p^{\mathrm{l}}_{\mathrm{LP}}(x) \leq p^{\mathrm{l}}_{\mathrm{exact}}(x) \leq p^{\mathrm{l}}_{\mathrm{greedy}}(x) \tag{11}$$

7.1 Basic multi-level solution evaluation scheme

Based on inequalities (11) we devise a multi-level solution evaluation scheme. Suppose that $p^1_{\text{LP}}(\hat{x})$ is the value of the leader's turnover obtained by LP evaluation of the best solution found so far $\hat{x}$. For each generated solution candidate x we evaluate it using greedy evaluation yielding a maximum achievable turnover of $p^1_{\text{greedy}}(x)$. Then we distinguish two cases:

- $p^1_{\text{greedy}}(x) \leq p^1_{\text{LP}}(\hat{x})$: This implies that $p^l_{\text{exact}}(x) \leq p^l_{\text{exact}}(\hat{x})$ and therefore x cannot be better than the so far best solution. So we do not put more effort in evaluating x more accurately.
- $p^1_{\text{greedy}}(x) > p^1_{\text{LP}}(\hat{x})$: We do not know if $p^l_{\text{exact}}(x) > p^l_{\text{exact}}(\hat{x})$ and therefore have to evaluate x more accurately. We do this by performing a more accurate (i.e., LP or exact) evaluation after the initial greedy evaluation to get a better estimate of the quality of x.

Preliminary tests showed that during an average run of our algorithm we can avoid the more accurate and thus more time-consuming solution evaluation for over 95 % of the solution candidates. Therefore it is likely that this method will reduce the overall optimization time of our algorithm in comparison to always performing an accurate evaluation. In Sect. 8.4 we will show that this multi-level solution evaluation scheme is able to improve the results significantly in terms of running time and final solution quality.

7.2 Multi-level solution evaluation scheme and local search

For intermediate local search a modification of the multi-level evaluation scheme is needed. Suppose that $\hat{x}$ is the so far best candidate solution with an objective value of $p^1_{\text{LP}}(\hat{x})$ which is obtained by LP evaluation. Furthermore let x' be the starting solution of the local search which has an objective value of $p^1_{\text{LP}}(x') \leq p^1_{\text{LP}}(\hat{x})$ also obtained by LP evaluation. Then we encounter a problem if the objective value $p^1_{\text{greedy}}(x)$ of a neighboring candidate solution x, which initially is obtained by greedy evaluation lies between $p^1_{\text{LP}}(x')$ and $p^1_{\text{LP}}(\hat{x})$, i.e.,

$$p^1_{\text{LP}}(x') < p^1_{\text{greedy}}(x) \leq p^1_{\text{LP}}(\hat{x}).$$

Since $p^1_{\text{greedy}}(x)$ is smaller than the best LP solution value found so far, x is not evaluated more accurately. It is, however, greater than the LP solution value of the starting solution of the LS so a move toward this solution is performed. This could lead to undesirable behavior because in fact we do not know if solution x is superior to solution x' and the LS would most likely perform moves towards a solution with a good greedy value instead of a solution with a good LP or exact value.

To avoid this problem we compare the solution value obtained by the initial greedy evaluation to the best LP solution value found so far in this local search call instead of the global best LP solution value for determining whether or not the solution shall be evaluated more accurately. This implies that in each iteration of the local search we start with a candidate solution that is evaluated using LP evaluation. This results in a local search towards the candidate solution with the best LP value at the cost of additional LP evaluations.

8 Computational results

In this section we present computational results of the developed methods. The instances used in all tests are partly taken from the benchmark library of *Discrete Location Problems*.[1] There are two types of instances: *Euclidean*, where the locations are chosen randomly on an Euclidean plane of size 7,000 × 7,000 and *Uniform* where all pairwise distances are chosen uniformly at random from the interval [1, 10000]. Customer demands are randomly selected from 1 to 200 or set to 1 for all customers. They have in common that the customers and the possible facility locations are on the same sites ($I = J$), the number of customers is $n = 100$ and the number of facilities to be opened is $r = p \in \{10, 15, 20\}$ for the Euclidean instances and $r = p = \{7, 25, 30\}$ for the Uniform instances. In addition we generated larger Euclidean instances[2] with 150 and 200 customers by using the same scheme. With a total of 10 instances per customer size and $r = p = \{10, 15, 20\}$, the Euclidean test set consists of 90 instances. The Uniform test set only considers instances with 100 customers and contains 30 instances. All our tests were carried out on a single core of an Intel Xeon Quadcore with 2.53 GHz and 3GB RAM. The models for the solution evaluation are solved using the IBM ILOG CPLEX Optimizer in version 12.5.

If not stated otherwise, in all of the following tests we used the GA configuration from Sect. 5 with a population size of 100. Local search is performed for a solution whose objective value is within $\alpha = 5\,\%$ of the overall best solution's value with the reduced swap neighborhood from Sect. 6 and a best improvement step function. After the algorithm terminates, the whole population is evaluated exactly to obtain the best feasible solution of the last population.

For all tables the following holds: Instances *Code111* to *Code1011* are the instances with $n = 100$ by Alekseeva et al. (2009) and instances *Code123* to *Code1023* are the uniform instances also with $n = 100$ by Davydov et al. (2014a). The other instance names contain either 150 or 200 which stands for the number of customers. The number right after *rp* corresponds to the number of facilities to place. In the first row the name of the algorithm is listed. The second row describes the columns, where $\overline{obj}$ stands for the average of the final leader objective value over 30 runs, *sd* is the corresponding standard deviation and t^* is the median time needed for finding the best solution in seconds. The highest (best) value for each instance out of the different configurations is marked in bold. All runs are terminated after 600 s to ensure comparability. First, we tested several configurations of the proposed algorithm on the Euclidean instances. Then, a comparison with different algorithms from the literature is presented. The results of computational tests on Uniform instances are described in Sect. 8.6. Due to space limitations Tables 1, 2, 3, 4, 5, 6 and 7 do not contain the numerical results of all

[1] http://math.nsc.ru/AP/benchmarks/Competitive/p_med_comp_eng.html.

[2] https://www.ads.tuwien.ac.at/w/Research/Problem_Instances#Competitive_Facility_Location_Problems.

Table 1 Results of different solution evaluation methods using the standard configuration

	Greedy			LP			Exact		
Instance	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$
Code111w_rp10	4,359.00	0.00	14.80	**4,361.00**	0.00	130.30	**4,361.00**	0.00	70.60
Code111w_rp15	4,547.11	6.01	20.10	**4,596.00**	0.00	64.10	**4,596.00**	0.00	55.60
Code111w_rp20	**4,508.50**	6.09	253.30	4,505.47	11.22	343.30	4,502.90	11.26	217.70
Code1_150w_rp10	7,132.20	130.36	250.20	7,138.37	112.88	88.60	**7,167.43**	51.47	94.00
Code1_150w_rp15	7,008.63	54.17	138.40	7,077.97	35.79	341.20	**7,088.83**	43.99	398.50
Code1_150w_rp20	7,070.67	52.46	314.20	7,198.27	19.01	380.20	**7,198.53**	22.50	370.60
Code1_200w_rp10	9,349.60	69.78	406.60	9,476.17	107.30	200.60	**9478.50**	92.39	369.70
Code1_200w_rp15	9,814.13	185.24	351.30	**10,001.40**	92.78	394.40	10,000.30	82.92	475.60
Code1_200w_rp20	9,615.13	135.94	411.90	**9,753.07**	77.54	572.80	9,697.53	85.19	586.90
Code211w_rp10	5,309.47	2.92	26.50	**5,310.00**	0.00	43.50	**5,310.00**	0.00	46.10
Code211w_rp15	**5,373.00**	0.00	97.10	**5,373.00**	0.00	111.80	**5,373.00**	0.00	95.70
Code211w_rp20	**5,431.57**	2.37	284.50	5,404.43	29.69	291.60	5,405.63	31.19	365.20
Code2_150w_rp10	7,181.53	52.91	332.10	7,247.47	53.43	292.50	**7,253.30**	71.29	291.00
Code2_150w_rp15	7,590.23	92.37	154.60	**7,743.20**	4.70	281.40	7,742.00	5.57	358.40
Code2_150w_rp20	7,673.90	83.24	255.00	**7,772.13**	40.91	349.50	7,755.50	46.49	347.80
Code2_200w_rp10	9,032.00	71.74	221.20	9,231.63	75.55	249.80	**9,254.53**	62.00	448.00
Code2_200w_rp15	9,274.23	153.66	312.40	**9,539.27**	70.94	516.40	9,505.43	109.72	438.40
Code2_200w_rp20	9,381.90	138.87	475.30	**9,579.83**	118.18	508.00	9570.30	110.98	548.80
Code311w_rp10	4,392.47	42.88	17.80	**4,483.00**	0.00	25.80	**4,483.00**	0.00	24.50
Code311w_rp15	4,782.10	18.23	221.60	**4,800.00**	0.00	73.60	**4,800.00**	0.00	63.20
Code311w_rp20	4,853.47	8.76	100.50	**4,892.80**	0.61	297.90	4,892.67	0.76	250.80
Code3_150w_rp10	7,240.20	75.69	362.80	7,286.93	16.36	310.70	**7,291.87**	13.30	369.20
Code3_150w_rp15	7,499.30	44.12	161.60	7,589.00	18.83	285.80	**7,589.27**	18.01	200.50
Code3_150w_rp20	7,520.40	63.06	303.20	**7,624.43**	34.03	309.10	7,624.37	34.20	411.20
Code3_200w_rp10	9,224.03	52.98	202.70	**9,300.23**	70.30	291.70	9,287.13	74.85	362.90
Code3_200w_rp15	9,145.17	210.97	378.30	9,304.57	71.44	386.40	**9,308.37**	70.27	459.10
Code3_200w_rp20	8,902.30	210.90	468.70	**9,197.97**	155.51	516.80	9145.73	107.33	574.10
…	…	…	…	…	…	…	…	…	…
Geometric mean	6,907.43			**6,995.12**			6,993.29		
#Best results	11			**53**			48		
#Unique best res.	6			**35**			31		

Table 2 Results of Wilcoxon rank sum tests with error levels of 5 % for the different solution evaluation methods

	Greedy	LP	Exact	Σ
Greedy	–	5	5	10
LP	75	–	6	**81**
Exact	73	4	–	77

Table 3 Results of different configurations for the GA: the pure genetic algorithm (GA), GA with local search (GA + LS), GA with solution archive (GA + solA) and GA with solution archive and local search (GA + LS + solA)

	GA			GA + LS			GA + solA			GA + LS + solA		
Instance	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$
Code111w_rp10	4,331.80	12.32	338.20	4,348.97	25.00	113.00	4,334.20	10.75	402.60	**4,361.00**	0.00	14.70
Code111w_rp15	4,572.37	21.18	461.10	4,586.43	16.65	38.50	4,582.67	6.63	412.80	**4,596.00**	0.00	16.10
Code111w_rp20	4,452.23	17.50	538.40	4,474.97	23.52	162.50	4,464.17	21.42	521.40	**4,505.47**	11.22	209.50
Code1_150w_rp10	6,503.63	123.24	538.20	**7,163.57**	54.75	113.40	6,606.03	120.76	530.10	7,138.37	112.88	29.20
Code1_150w_rp15	6,823.13	63.36	549.60	7,021.47	75.64	149.30	6,876.50	49.83	546.80	**7,077.97**	35.79	133.00
Code1_150w_rp20	6,900.63	109.91	550.20	7,163.03	58.69	187.30	6,980.63	105.85	560.30	**7,198.27**	19.01	241.40
Code1_200w_rp10	8,810.10	201.26	531.40	9,443.60	105.48	246.70	8,926.30	189.46	509.10	**9,476.17**	107.30	243.10
Code1_200w_rp15	9,079.43	265.11	572.00	9,956.77	112.63	349.90	9,212.53	227.11	553.60	**10,001.40**	92.78	297.10
Code1_200w_rp20	8,899.63	157.41	562.00	9,683.20	119.85	479.60	8,996.93	219.23	560.80	**9,753.07**	77.54	460.50
Code211w_rp10	5,289.47	25.60	521.20	**5,310.00**	0.00	51.20	5,291.13	28.09	473.40	**5,310.00**	0.00	8.10
Code211w_rp15	5,279.47	35.12	546.90	5,362.33	19.73	42.40	5,294.43	31.31	493.60	**5,373.00**	0.00	23.10
Code211w_rp20	5,250.03	52.47	556.90	5,351.80	55.16	135.10	5,268.93	47.01	546.80	**5,404.43**	29.69	82.40
Code2_150w_rp10	6,962.67	46.48	519.90	7,229.90	89.61	233.40	6,969.40	43.43	512.40	**7,247.47**	53.43	242.70
Code2_150w_rp15	7,423.00	94.91	537.40	7,702.23	103.32	148.00	7,496.67	66.09	516.30	**7,743.20**	4.70	96.90
Code2_150w_rp20	7,439.37	65.12	540.10	7,713.43	70.96	208.40	7,500.33	60.97	551.80	**7,772.13**	40.91	211.50
Code2_200w_rp10	8,609.17	151.04	524.40	9,181.07	127.89	228.80	8,717.20	131.21	529.70	**9,231.63**	75.55	130.10
Code2_200w_rp15	8,639.43	159.89	523.30	9,482.17	119.41	342.10	8,678.77	144.79	520.70	**9,539.27**	70.94	392.00
Code2_200w_rp20	8,626.47	119.66	557.10	9,508.30	119.17	521.50	8,726.30	112.43	547.80	**9,579.83**	118.18	421.30
Code311w_rp10	4,472.50	34.65	331.60	**4,483.00**	0.00	21.20	**4,483.00**	0.00	336.20	**4,483.00**	0.00	8.10
Code311w_rp15	4,775.93	19.79	473.20	4,785.40	22.85	70.40	4,781.13	21.89	534.60	**4,800.00**	0.00	13.30
Code311w_rp20	4,835.77	15.62	534.70	4,879.17	14.61	116.60	4,849.67	11.70	495.30	**4,892.80**	0.61	65.20

Table 3 continued

	GA			GA + LS			GA + solA			GA + LS + solA		
Instance	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$
Code3_150w_rp10	6,975.17	75.29	522.30	7,252.50	69.75	136.20	7,004.47	71.96	493.00	**7,286.93**	16.36	35.40
Code3_150w_rp15	7,333.73	124.31	551.10	7,554.00	47.95	131.20	7,391.33	95.72	523.60	**7,589.00**	18.83	142.50
Code3_150w_rp20	7,358.33	70.60	543.60	7,601.30	45.29	214.80	7,406.17	50.70	559.40	**7,624.43**	34.03	274.00
Code3_200w_rp10	8,832.40	181.41	553.50	9,229.40	86.17	239.00	8,856.23	185.09	535.80	**9,300.23**	70.30	227.00
Code3_200w_rp15	8,232.10	269.27	563.40	9,265.43	98.36	333.10	8,497.40	203.87	562.00	**9,304.57**	71.44	281.90
Code3_200w_rp20	8,091.83	188.37	564.80	9,150.37	174.73	509.90	8,251.03	134.42	550.60	**9,197.97**	155.51	426.90
…	…	…	…	…	…	…	…	…	…	…	…	…
Geometric mean	6,643.72			6,964.10			6,691.70			**6,995.12**		
#Best results	1			10			3			**85**		
#Unique best res.	0			5			0			**80**		

Table 4 Results of Wilcoxon rank sum tests with error levels of 5 % for the different configurations of the GA

	GA	GA + LS	GA + solA	GA + LS + solA	Σ
GA	–	0	0	0	0
GA + LS	85	–	84	0	169
GA + SA	56	0	–	0	56
GA + LS + SA	87	60	87	–	**234**

of the 90 *Euclidean* instances but only a representative selection. The full result tables can be found online.[2] In addition, the geometric mean, the number of best results and the number of unique best results are shown over all instances.

In most of the following sections there is a second table after the main results table. These tables display the results of pairwise Wilcoxon rank sum tests of the different configurations with error levels of 5 %. The value in the cell at line i and row j gives the number of instances for which configuration i yields significantly better results than configuration j. The rightmost column lists the sums over all numbers in the corresponding rows.

8.1 Solution evaluation on euclidean instances

In the following tests we compare three types of solution evaluation schemes according to Sect. 4: greedy evaluation, LP evaluation and exact evaluation. The aim of these tests is to find out which runtime/solution accuracy tradeoff is suitable for this problem.

Table 1 shows the results. As we can see, although each greedy evaluation is 4–5 times faster than the LP evaluation, the results for the greedy evaluation are rather poor because the solution with the highest greedy value often does not correspond to an optimal solution according to the exact evaluation. In contrast, the results for evaluating solutions using the LP evaluation are similiar to those obtained by using the exact evaluation. In many cases the root LP relaxation of the follower's problem is already integral and no branching has to be performed, hence the similar results. Therefore, for the remaining tests we primarily use the LP evaluation method.

8.2 Genetic algorithm on Euclidean instances

Now, we analyze different configurations of the GA. The GA was tested with and without the local search and with and without the solution archive. The aim was to see the impact of using the different techniques on the average solution quality and speed. Table 3 shows the computational results. We can make several interesting observations: As expected, the GA alone performs not very well, neither regarding solution quality nor convergence speed, but its performance is substantially improved by executing intermediate local searches. By adding the solution archive (solA) to the pure GA we were also able to significantly improve the results. The benefit of the local search seems to be greater than the benefit of the solution archive because the relative difference of the geometric mean of GA + LS and the GA is about 5 % while the difference of GA + SA and GA is only about 0.7 %. Adding both, LS and solA, to the GA clearly further improves the performance. For this combined approval not only the solution quality

Table 5 Results of using different neighborhood (NB) structures for intermediate local / tabu search (TS)

	Complete NB			Reduced NB			Conversion NB			TS with reduced NB		
Instance	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$
Code111w_rp10	**4,361.00**	0.00	133.80	**4,361.00**	0.00	130.30	**4,361.00**	0.00	72.20	**4,361.00**	0.00	47.00
Code111w_rp15	**4,596.00**	0.00	44.90	**4,596.00**	0.00	64.10	**4,596.00**	0.00	79.40	**4,596.00**	0.00	55.10
Code111w_rp20	4,488.20	22.42	299.50	4,505.47	11.22	343.30	4,497.93	14.69	343.60	**4,506.23**	8.39	268.70
Code1_150w_rp10	7,157.07	89.96	113.60	7,138.37	112.88	88.60	7,171.30	47.65	137.40	**7,180.00**	0.00	118.00
Code1_150w_rp15	7,055.70	39.71	240.40	7,077.97	35.79	341.20	7,070.77	45.79	339.40	**7,143.30**	31.63	337.00
Code1_150w_rp20	7,187.87	22.59	289.30	7,198.27	19.01	380.20	7,190.87	24.84	400.70	**7,221.50**	27.75	455.20
Code1_200w_rp10	9,471.80	85.42	286.70	9,476.17	107.30	200.60	9,508.00	67.29	400.10	**9,532.50**	56.10	350.30
Code1_200w_rp15	9,959.17	133.66	483.60	**10,001.40**	92.78	394.40	9,988.87	100.51	461.90	9,986.13	99.43	500.60
Code1_200w_rp20	9,706.37	70.27	530.90	9,753.07	77.54	572.80	9,720.67	74.29	521.80	**9,760.10**	69.67	600.00
Code211w_rp10	**5,310.00**	0.00	85.50	**5,310.00**	0.00	43.50	**5,310.00**	0.00	29.00	**5,310.00**	0.00	33.10
Code211w_rp15	5,367.60	8.39	62.40	**5,373.00**	0.00	111.80	**5,373.00**	0.00	100.40	**5,373.00**	0.00	162.80
Code211w_rp20	5,386.77	37.42	291.80	5,404.43	29.69	291.60	5,404.13	30.62	439.50	**5,427.37**	9.90	200.00
Code2_150w_rp10	7,234.77	55.54	303.70	7,247.47	53.43	292.50	7,265.53	58.72	216.00	**7,290.17**	48.81	401.10
Code2_150w_rp15	7,738.40	13.75	306.10	**7,743.20**	4.70	281.40	7,742.30	5.53	367.50	7,741.60	6.29	342.30
Code2_150w_rp20	7,737.80	51.87	360.50	7,772.13	40.91	349.50	7,771.40	39.29	390.40	**7,774.13**	59.61	317.60
Code2_200w_rp10	9,214.43	102.90	266.60	9,231.63	75.55	249.80	9,230.23	94.53	395.30	**9,252.07**	89.10	376.50
Code2_200w_rp15	9,525.37	92.86	419.00	9,539.27	70.94	516.40	9,518.10	93.23	439.20	**9,561.30**	83.08	591.40
Code2_200w_rp20	9,542.93	106.14	549.80	9,579.83	118.18	508.00	9,549.43	101.89	549.80	**9,619.70**	67.33	600.00
Code311w_rp10	**4,483.00**	0.00	25.40	**4,483.00**	0.00	25.80	**4,483.00**	0.00	23.80	**4,483.00**	0.00	31.10
Code311w_rp15	**4,800.00**	0.00	85.90	**4,800.00**	0.00	73.60	**4,800.00**	0.00	50.80	**4,800.00**	0.00	49.20
Code311w_rp20	4,890.63	3.95	183.30	**4,892.80**	0.61	297.90	4,892.67	0.76	264.90	4,892.73	0.69	213.90
Code3_150w_rp10	7,285.67	15.76	113.40	7,286.93	16.36	310.70	7,293.10	12.10	314.30	**7,299.00**	0.00	130.90

Table 5 continued

	Complete NB			Reduced NB			Conversion NB			TS with reduced NB		
Instance	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$
Code3_150w_rp15	7,567.03	34.68	193.00	7,589.00	18.83	285.80	**7,589.90**	18.86	286.60	7,583.77	20.38	192.40
Code3_150w_rp20	7,605.33	47.90	298.50	7,624.43	34.03	309.10	**7,627.93**	26.49	331.80	7,620.67	46.21	429.70
Code3_200w_rp10	9,265.97	102.41	247.40	9,300.23	70.30	291.70	9,275.87	85.79	320.20	**9,339.97**	76.46	375.40
Code3_200w_rp15	9,279.60	91.05	383.20	9,304.57	71.44	386.40	9,301.60	68.80	438.00	**9,317.83**	66.40	496.10
Code3_200w_rp20	9,149.43	118.80	518.40	9,197.97	155.51	516.80	9,147.50	115.48	517.80	**9,220.97**	105.33	600.00
…	…	…	…	…	…	…	…	…	…	…	…	…
Geometric mean	6,983.65			6,995.12			6,992.38			**7,006.53**		
#Best results	13			27			22			**74**		
#Unique best results	0			9			7			**55**		

Table 6 Results of Wilcoxon rank sum tests with error levels of 5 % for the different local search neighborhood structures and tabu search

	Complete NB	Reduced NB	Conversion NB	TS with reduced NB	Σ
Complete NB	–	1	0	0	1
Reduced NB	31	–	5	0	36
Conversion NB	23	3	–	0	26
TS with reduced NB	54	21	32	–	**107**

is the best among the configurations but these solutions in most of the cases are also found faster.

8.3 Neighborhoods of the local search and tabu search on Euclidean instances

Table 5 shows the results of using the different strategies for utilizing the solA within LS and tabu search (TS), respectively (c.f. Sect. 6). As expected the complete neighborhood strategy performed worst because of the overhead of re-evaluating already visited solutions but on some of the smaller test instances it is able to produce equally good results. Among all tested LS neighborhoods, reduced neighborhood yields the best results, so it is chosen for all further tests. While on the smaller test instances with 100 customers the conversion and the complete neighborhood can keep up with the reduced neighborhood in terms of mean objective value, on larger instances the performance gap increases. The differences in the objective value of the conversion neighborhood and the reduced neighborhood are small and the conversion neighborhood even finds the best solution in less time for some instances, e.g., *Code111w_rp10* and *Code211w_rp10*. However, this difference vanishes when considering larger instances, where the reduced neighborhood consistently finds better solutions. Apparently, for these instances conversion moves were too rarely able to improve the starting solution. The largest improvement of the overall results could be achieved by using a tabu search with the reduced neighborhood. In none of our benchmark instances any other configuration was able to find solutions with a statistical significant better mean objective value.

8.4 Multi-level evaluation scheme on Euclidean instances

The computational results for testing the multi-level evaluation scheme (ML-ES) confirms the hypothesis that it is able to speed up the algorithm significantly. We further tested if the local search using the local best LP solution (improved LS) as described in Sect. 7.2 actually improves the solution quality. Finally we investigated the tabu search approach (improved TS), which is explained in Sect. 6.4 in combination with the reduced NB. For the TS we also used the adaptation for the improved LS in a straightforward way and set a termination criterion of five iterations without improvement.

Table 7 Results for the multi-level evaluation scheme and the local/tabu search improvement compared to the standard LP solution evaluation

	GA + solA + LP + LS			GA + solA + ML-ES + LS			GA + solA + ML-ES + improved LS			GA + solA + ML-ES + improved TS		
Instance	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$
Code111w_rp10	**4,361.00**	0.00	130.30	**4,361.00**	0.00	13.00	**4,361.00**	0.00	12.70	**4,361.00**	0.00	14.70
Code111w_rp15	**4,596.00**	0.00	64.10	**4,596.00**	0.00	22.20	**4,596.00**	0.00	12.20	**4,596.00**	0.00	16.10
Code111w_rp20	4,505.47	11.22	343.30	4,507.77	4.15	181.80	4,511.47	1.38	231.30	**4,511.87**	0.73	209.50
Code1_150w_rp10	7,138.37	112.88	88.60	**7,180.00**	0.00	23.50	**7,180.00**	0.00	33.70	**7,180.00**	0.00	29.20
Code1_150w_rp15	7,077.97	35.79	341.20	7,080.23	45.92	110.30	7,130.97	36.67	291.60	**7,153.93**	0.25	133.00
Code1_150w_rp20	7,198.27	19.01	380.20	7,198.70	23.39	264.90	7,208.13	18.97	282.50	**7,247.27**	7.97	241.40
Code1_200w_rp10	9,476.17	107.30	200.60	9,515.23	54.65	201.80	9,558.83	37.26	298.50	**9,594.00**	10.37	243.10
Code1_200w_rp15	10,001.40	92.78	394.40	10,026.23	67.17	134.20	10,077.10	42.96	289.20	**10,095.00**	37.02	297.10
Code1_200w_rp20	9,753.07	77.54	572.80	9,742.10	93.20	225.20	9,807.57	74.67	460.80	**9,831.97**	56.35	460.50
Code211w_rp10	**5,310.00**	0.00	43.50	**5,310.00**	0.00	8.60	**5,310.00**	0.00	8.30	**5,310.00**	0.00	8.10
Code211w_rp15	**5,373.00**	0.00	111.80	**5,373.00**	0.00	37.00	**5,373.00**	0.00	17.80	**5,373.00**	0.00	23.10
Code211w_rp20	5,404.43	29.69	291.60	5,427.80	10.53	252.60	**5,432.00**	0.00	165.70	5,431.57	2.37	82.40
Code2_150w_rp10	7,247.47	53.43	292.50	7,276.70	61.74	98.60	7,328.97	23.91	166.70	**7,337.00**	0.00	242.70
Code2_150w_rp15	7,743.20	4.70	281.40	7,735.93	9.15	60.30	7,744.00	3.81	102.90	**7,745.00**	0.00	96.90
Code2_150w_rp20	7,772.13	40.91	349.50	7,759.60	40.59	181.30	7,789.27	28.71	177.00	**7,802.03**	15.79	211.50
Code2_200w_rp10	9,231.63	75.55	249.80	9,238.23	78.81	58.30	9,307.13	53.11	236.50	**9,321.13**	26.28	130.10
Code2_200w_rp15	9,539.27	70.94	516.40	9,471.07	78.21	119.40	9,593.53	53.40	345.90	**9,626.67**	17.34	392.00
Code2_200w_rp20	9,579.83	118.18	508.00	9,599.40	88.02	241.50	9,643.80	80.47	402.70	**9,666.37**	52.72	421.30
Code311w_rp10	**4,483.00**	0.00	25.80	**4,483.00**	0.00	5.90	**4,483.00**	0.00	7.90	**4,483.00**	0.00	8.10
Code311w_rp15	**4,800.00**	0.00	73.60	**4,800.00**	0.00	13.50	**4,800.00**	0.00	19.80	**4,800.00**	0.00	13.30
Code311w_rp20	4,892.80	0.61	297.90	4,889.33	6.19	100.10	**4,893.00**	0.00	103.30	**4,893.00**	0.00	65.20

Table 7 continued

	GA + solA + LP + LS			GA + solA + ML-ES + LS			GA + solA + ML-ES + improved LS			GA + solA + ML-ES + improved TS		
Instance	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$	$\overline{obj}$	*sd*	$t^*[s]$
Code3_150w_rp10	7,286.93	16.36	310.70	7,292.50	13.38	62.90	7,296.83	8.33	64.10	**7,299.00**	0.00	35.40
Code3_150w_rp15	7,589.00	18.83	285.80	7,580.97	23.60	45.60	7,597.77	13.80	207.80	**7,603.10**	2.75	142.50
Code3_150w_rp20	7,624.43	34.03	309.10	7,605.77	60.47	111.10	7,636.47	15.28	289.20	**7,646.87**	4.32	274.00
Code3_200w_rp10	9,300.23	70.30	291.70	9,320.53	62.72	164.60	9,358.97	48.66	283.30	**9,374.30**	28.15	227.00
Code3_200w_rp15	9,304.57	71.44	386.40	9,310.33	71.22	192.00	9,353.33	39.29	388.40	**9,365.97**	17.19	281.90
Code3_200w_rp20	9,197.97	155.51	516.80	9,265.93	107.10	252.20	9,285.73	81.47	416.80	**9,296.67**	70.96	426.90
…	…	…	…	…	…	…	…	…	…	…	…	…
Geometric mean	6,995.12			7,000.54			7,019.37			**7,027.16**		
#Best results	19			24			43			**85**		
#Unique best results	0			0			5			**47**		

Table 8 Results of Wilcoxon rank sum tests with error levels of 5 % for the multi-level evaluation scheme configurations

	GA + solA + LP + LS	GA + solA + ML-ES + LS	GA + solA + ML-ES + imp. LS	GA + solA + ML-ES + imp. TS	Σ
GA + solA + LP + LS	–	10	0	0	10
GA + solA + ML-ES + LS	17	–	1	2	18
GA + solA + ML-ES + imp. LS	60	55	–	1	115
GA + solA + ML-ES + imp. TS	63	59	29	–	**151**

Table 7 shows the results of these tests. We observe that the multi-level evaluation scheme is able to improve the solution quality for some instances, especially the larger ones with 200 customers. The largest improvement could be made in the time needed for finding the best solution. It is in general much lower than when using only the simple LP evaluation, e.g., for instance *Code111w_rp10* the time could be decreased by about 90 %. With the improved local search the mean solution quality gets better in 65 of the (mostly larger) instances, while it is equal on most of the other ones. Our best setup turned out to be GA + solA + ML-ES + improved TS, when we switched from a local search to a tabu search. We have a low standard deviation of the results and achieved a better mean objective value than the local search in 47 instances. The improvements are again mostly on the larger instances with 150 and 200 customers because, as we see in Sect. 8.5, we could find optimal solutions for many of the instances with 100 customers. In total, our best configurations of the multi-level evaluation scheme was able to produce statistically better results in 63 out of 90 instances (Table 8).

8.5 Comparison to results from the literature on Euclidean instances

In this section we compare the results of our best configuration to the state-of-the-art in the literature. Since the metaheuristic approaches of Alekseeva et al. (2010), Alekseeva and Kochetov (2013), and Davydov et al. (2014a) are the best heuristic approaches so far we compare with them. For this purpose both the probabilistic tabu search (TS_{Al}) (Alekseeva et al. 2010) and the hybrid memetic algorithm (HMA) (Alekseeva and Kochetov 2013) were re-implemented in C++. Our re-implementations were verified to exhibit nearly equal performance as published in the respective original papers. The results of the STS are taken from Davydov et al. (2014a), where they presented the objective values of single runs (obj_1). Although they developed two metaheuristics here we only compare with STS because the other one, the VNS, performed worse on Euclidean instances.

Table 9 shows the results of their approaches compared to our algorithm (GA + solA + ML-ES + imp. TS) with $n = 100$. Additionally, Tables 10 and 11 show

Table 9 Comparison to results from the literature with a runtime of 600 s and $n = 100$

	TS_{Al}			HMA			STS		GA + solA + ML-ES + imp.TS		
Instance	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$	obj_1	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$
Code111w_rp10	**4,361.00**	0.00	118.0	**4,361.00**	0.00	92.3	**4,361.00**	63.7	**4,361.00**	0.00	14.7
Code111w_rp15	**4,596.00**	0.00	38.6	**4,596.00**	0.00	106.8	**4,596.00**	173.3	**4,596.00**	0.00	16.1
Code111w_rp20	4,506.87	6.96	92.1	4,510.60	2.03	393.6	4,484.00	118.1	**4,511.87**	0.73	209.5
Code211w_rp10	**5,310.00**	0.00	11.9	**5,310.00**	0.00	26.6	**5,310.00**	23.5	**5,310.00**	0.00	8.1
Code211w_rp15	**5,373.00**	0.00	115.5	**5,373.00**	0.00	121.9	**5,373.00**	88.9	**5,373.00**	0.00	23.1
Code211w_rp20	5,428.13	6.01	167.1	5,430.67	3.40	287.2	**5,432.00**	289.2	5,431.57	2.37	82.4
Code311w_rp10	**4,483.00**	0.00	11.6	**4,483.00**	0.00	45.1	**4,483.00**	33.8	**4,483.00**	0.00	8.1
Code311w_rp15	**4,800.00**	0.00	71.3	4,799.77	1.28	122.5	**4,800.00**	91.1	**4,800.00**	0.00	13.3
Code311w_rp20	4,892.73	0.69	94.7	4,892.60	0.81	297.8	**4,893.00**	211.3	**4,893.00**	0.00	65.2
Code411w_rp10	**4,994.00**	0.00	11.7	**4,994.00**	0.00	24.0	**4,994.00**	19.3	**4,994.00**	0.00	7.9
Code411w_rp15	5,063.20	2.07	139.0	5,063.80	1.10	201.0	**5,064.00**	121.3	**5,064.00**	0.00	48.8
Code411w_rp20	**5,209.00**	0.00	105.6	5,208.93	0.25	275.5	**5,209.00**	288.9	**5,209.00**	0.00	39.6
Code511w_rp10	**4,906.00**	0.00	38.8	**4,906.00**	0.00	81.1	**4,906.00**	27.2	**4,906.00**	0.00	8.9
Code511w_rp15	5,123.00	0.00	104.1	5,127.00	4.07	263.1	**5,131.00**	216.2	5,123.00	0.00	63.4
Code511w_rp20	5,327.30	13.81	231.5	5,329.93	7.26	219.5	**5,334.00**	133.2	**5,334.00**	0.00	76.0
Code611w_rp10	**4,595.00**	0.00	82.7	**4,595.00**	0.00	93.3	**4,595.00**	44.5	**4,595.00**	0.00	17.7
Code611w_rp15	**4,881.00**	0.00	47.3	**4,881.00**	0.00	67.0	**4,881.00**	114.8	**4,881.00**	0.00	15.7
Code611w_rp20	4,951.73	1.46	137.0	4,951.20	2.44	225.5	4,944.00	198.1	**4,952.00**	0.00	96.0
Code711w_rp10	**5,586.00**	0.00	48.9	**5,586.00**	0.00	78.9	**5,586.00**	101.0	**5,586.00**	0.00	8.7
Code711w_rp15	**5,827.00**	0.00	155.5	5,826.27	4.02	160.9	**5,827.00**	210.9	**5,827.00**	0.00	31.6
Code711w_rp20	5,884.37	15.92	53.7	5,892.30	2.74	216.4	**5,893.00**	254.3	**5,893.00**	0.00	29.8
Code811w_rp10	**4,609.00**	0.00	70.5	**4,609.00**	0.00	169.6	**4,609.00**	27.2	**4,609.00**	0.00	21.6
Code811w_rp15	4,674.47	1.38	158.4	4,674.87	0.73	233.9	**4,675.00**	123.4	**4,675.00**	0.00	41.6
Code811w_rp20	4,857.63	2.01	154.6	4,854.60	6.59	271.3	**4,858.00**	118.8	**4,858.00**	0.00	24.4
Code911w_rp10	**5,302.00**	0.00	28.8	**5,302.00**	0.00	37.2	**5,302.00**	19.2	**5,302.00**	0.00	7.5
Code911w_rp15	5,157.63	1.13	204.7	5,156.90	2.01	284.7	**5,158.00**	157.8	5,157.93	0.25	220.6
Code911w_rp20	5,458.67	1.03	178.7	5,457.50	1.78	208.9	5,455.00	202.2	**5,459.00**	0.00	92.5
Code1011w_rp10	5,003.67	7.30	66.7	5,004.10	4.93	104.4	**5,005.00**	103.5	**5,005.00**	0.00	18.2
Code1011w_rp15	5,194.47	2.29	233.2	5,194.23	3.52	223.6	**5,195.00**	48.2	**5,195.00**	0.00	29.2
Code1011w_rp20	**5,399.00**	0.00	49.0	**5,399.00**	0.00	261.8	**5,399.00**	184.4	**5,399.00**	0.00	60.8
Geometric mean	5,043.99			5,044.47			5,043.74		**5,044.90**		
#Best results	16			13			**27**		**27**		
#Un. best res.	0			0			**3**		**3**		

The Tabu search (TS_{Al}), the hybrid memetic algorithm (HMA), and the STS approach by Alekseeva et al. (2010), Alekseeva and Kochetov (2013), and Davydov et al. (2014a), respectively, compared to our best configuration GA + solA + ML-ES + improved TS

Table 10 Comparison to results from the literature with a runtime of 600 s and $n = 150$

	TS_{Al}			HMA			GA + solA + ML-ES + improved TS		
Instance	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$
Code1_150w_rp10	**7,180.00**	0.00	40.10	**7,180.00**	0.00	118.50	**7,180.00**	0.00	29.20
Code1_150w_rp15	7,152.23	5.02	207.00	7,132.60	32.10	428.80	**7,153.93**	0.25	133.00
Code1_150w_rp20	**7,247.77**	7.45	368.60	7,211.07	26.63	353.10	7,247.27	7.97	241.40
Code2_150w_rp10	7,321.07	23.97	146.70	7,325.57	16.59	334.60	**7,337.00**	0.00	242.70
Code2_150w_rp15	7,736.87	8.36	174.60	7,732.87	11.63	335.90	**7,745.00**	0.00	96.90
Code2_150w_rp20	7,796.43	14.76	386.30	7,770.07	25.08	510.10	**7,802.03**	15.79	211.50
Code3_150w_rp10	**7,299.00**	0.00	103.90	**7,299.00**	0.00	267.90	**7,299.00**	0.00	35.40
Code3_150w_rp15	7,596.47	12.06	252.60	7,593.07	16.98	231.70	**7,603.10**	2.75	142.50
Code3_150w_rp20	7,610.47	63.78	217.80	7,630.60	14.08	242.60	**7,646.87**	4.32	274.00
Code4_150w_rp10	7,306.17	39.97	157.00	7,307.63	19.12	298.00	**7,318.00**	0.00	38.30
Code4_150w_rp15	7,406.73	7.08	180.10	7,392.30	18.53	259.10	**7,409.00**	0.00	71.30
Code4_150w_rp20	7,926.00	5.19	227.30	7,917.87	10.70	244.00	**7,927.50**	2.74	251.30
Code5_150w_rp10	6,972.50	5.19	173.70	6,968.90	8.56	202.40	**6,975.00**	0.00	32.90
Code5_150w_rp15	**7,154.77**	19.25	370.60	7,135.10	26.72	459.90	7,139.97	26.56	214.60
Code5_150w_rp20	7,322.50	6.30	272.60	7,316.13	13.54	457.00	**7,326.50**	3.29	227.30
Code6_150w_rp10	7,047.27	7.02	182.50	7,043.60	10.88	323.50	**7,050.00**	0.00	36.90
Code6_150w_rp15	7,184.83	4.49	183.80	7,172.50	16.84	409.60	**7,186.00**	0.00	71.60
Code6_150w_rp20	7,378.10	14.45	338.40	7,333.67	39.97	500.50	**7,386.00**	0.00	133.90
Code7_150w_rp10	6,247.10	3.59	264.10	**6,248.17**	2.57	397.90	6,248.10	0.55	190.10
Code7_150w_rp15	6,839.60	2.19	107.20	6,834.33	9.36	175.60	**6,840.00**	0.00	82.90
Code7_150w_rp20	7,284.37	18.24	129.50	7,275.30	20.64	341.10	**7,290.83**	14.02	203.10
Code8_150w_rp10	**7,732.00**	0.00	159.20	**7,732.00**	0.00	232.20	**7,732.00**	0.00	28.70
Code8_150w_rp15	7,658.23	7.54	237.50	7,650.80	20.36	443.60	**7,662.00**	0.00	103.10
Code8_150w_rp20	**7,848.80**	8.38	164.50	7,836.40	18.38	428.80	7,846.73	11.06	188.20
Code9_150w_rp10	**6,855.00**	0.00	182.20	6,853.47	5.84	309.40	**6,855.00**	0.00	55.50
Code9_150w_rp15	6,881.30	5.52	297.90	6,878.13	7.77	350.70	**6,883.40**	0.93	148.40
Code9_150w_rp20	**7,177.90**	19.76	230.80	7,145.17	35.49	453.60	7,160.40	41.30	299.90
Code10_150w_rp10	**6,715.00**	0.00	88.80	**6,715.00**	0.00	209.40	**6,715.00**	0.00	30.20
Code10_150w_rp15	7,009.07	13.99	231.00	7,008.07	16.87	405.10	**7,014.00**	0.00	104.30
Code10_150w_rp20	7,201.07	13.43	320.00	7,181.53	24.32	489.10	**7,203.40**	10.21	175.30
Geometric mean	7,260.27			7,251.42			**7,263.31**		
#Best results	9			5			**25**		
#Unique best res.	4			1			**20**		

The Tabu search (TS_{Al}) and the hybrid memetic algorithm (HMA) by Alekseeva et al. (2010) and Alekseeva and Kochetov (2013), respectively, compared to our best configuration GA + solA + ML-ES + improved TS

Table 11 Comparison to results from the literature with a runtime of 600 s and $n = 200$

	TS$_{\mathrm{Al}}$			HMA			GA + solA + ML-ES + improved TS		
Instance	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$
Code1_200w_rp10	9,545.43	35.14	402.30	9,505.07	57.16	348.50	**9,594.00**	10.37	243.10
Code1_200w_rp15	10,076.73	49.31	337.60	10,051.83	59.42	348.30	**10,095.00**	37.02	297.10
Code1_200w_rp20	**9,837.17**	53.95	405.50	9,767.93	58.96	365.60	9,831.97	56.35	460.50
Code2_200w_rp10	**9,324.50**	50.20	279.80	9,217.80	58.07	455.30	9,321.13	26.28	130.10
Code2_200w_rp15	9,578.77	46.03	370.60	9,514.93	51.54	286.20	**9,626.67**	17.34	392.00
Code2_200w_rp20	**9,667.17**	32.12	386.70	9,602.20	38.63	258.70	9,666.37	52.72	421.30
Code3_200w_rp10	9,367.07	32.45	237.10	9,329.37	53.93	350.40	**9,374.30**	28.15	227.00
Code3_200w_rp15	9,355.93	18.85	323.80	9,310.30	44.48	317.70	**9,365.97**	17.19	281.90
Code3_200w_rp20	9,286.17	67.10	358.20	9,253.50	63.57	313.00	**9,296.67**	70.96	426.90
Code4_200w_rp10	8,882.03	18.31	259.60	8,877.13	22.02	460.40	**8,888.47**	14.39	115.60
Code4_200w_rp15	9,169.93	18.46	279.30	9,116.27	68.57	365.90	**9,179.03**	32.68	241.30
Code4_200w_rp20	**9,439.13**	34.47	393.30	9,402.23	55.74	248.50	9,404.70	89.41	388.50
Code5_200w_rp10	9,227.30	48.62	294.60	9,240.40	52.15	304.80	**9,273.10**	27.45	268.20
Code5_200w_rp15	9,242.57	64.44	382.50	9,237.70	41.65	325.90	**9,252.03**	42.10	320.90
Code5_200w_rp20	9,498.80	38.81	364.40	9,422.63	52.81	379.40	**9,512.10**	42.91	345.90
Code6_200w_rp10	9,825.20	35.02	402.10	9,808.13	39.34	330.50	**9,850.53**	5.58	197.50
Code6_200w_rp15	10,119.03	52.39	401.30	10,095.73	41.17	269.30	**10,148.23**	27.71	326.70
Code6_200w_rp20	**10,283.10**	83.09	438.90	10,210.53	59.37	270.30	10,261.53	91.67	452.50
Code7_200w_rp10	9,225.70	42.60	356.90	9,183.77	55.95	382.20	**9,270.30**	20.44	222.80
Code7_200w_rp15	9,556.13	39.65	424.40	9,496.63	59.54	267.30	**9,580.30**	35.03	283.90
Code7_200w_rp20	9,902.20	43.20	430.40	9,860.03	52.13	221.80	**9,943.10**	33.88	361.90
Code8_200w_rp10	9,088.17	9.62	269.40	9,046.43	34.70	345.10	**9,092.57**	2.37	170.60
Code8_200w_rp15	9,047.13	47.40	413.80	8,987.20	41.46	244.00	**9,063.10**	41.76	357.90
Code8_200w_rp20	9,329.67	29.32	368.20	9,248.07	59.96	302.60	**9,342.90**	23.35	484.30
Code9_200w_rp10	9,009.53	3.68	381.70	8,950.47	59.78	324.80	**9,011.40**	8.76	182.90
Code9_200w_rp15	9,124.70	66.93	341.00	9,086.47	65.56	297.60	**9,168.20**	23.40	335.40
Code9_200w_rp20	9,438.00	17.91	383.40	9,404.67	42.67	300.90	**9,452.57**	16.55	416.80
Code10_200w_rp10	9,382.67	25.28	388.00	9,365.40	46.44	498.60	**9,411.00**	0.00	151.70
Code10_200w_rp15	9,290.80	49.24	408.10	9,240.83	57.79	252.70	**9,312.40**	51.91	434.30
Code10_200w_rp20	**9,741.20**	35.77	467.60	9,683.63	50.92	328.30	9,688.73	74.95	460.40
Geometric mean	9,456.10			9,411.33			**9,470.05**		
#Best results	6			0			**24**		
#Unique best res.	6			0			**24**		

The Tabu search (TS$_{\mathrm{Al}}$) and the hybrid memetic algorithm (HMA) by Alekseeva et al. (2010) and Alekseeva and Kochetov (2013), respectively, compared to our best configuration GA + solA + ML-ES + improved TS

the results of it compared to TS$_{\mathrm{Al}}$ and HMA with $n = 150$ and $n = 200$. It can be seen that especially for larger instances our algorithm achieves the best results among all three tested algorithms (see Table 12). For the instances with 100 customers the algo-

Table 12 Results of Wilcoxon rank sum tests with error levels of 5 % for the algorithms of the literature and the GA

	TS	HMA	GA + solA + ML-ES + improved TS	Σ
TS	–	45	3	57
HMA	4	–	1	7
GA + solA + ML-ES + improved TS	38	56	–	**123**

rithm described in this work gets better or equally good results than TS_{Al} and HMA in all but one instances, although the differences in the mean objective value is rather small. Compared to STS both algorithms get better objective values on 3 instances and equally good solutions on the remaining 24 instances. However, the time needed to find these solutions is much lower for most instances when using our algorithm. The differences in the objective value become larger when considering larger instances. On all instances with $n = 200$ our algorithm obtains better results than HMA and on 24 out of 30 instances it also gets better mean objective values than TS_{Al}.

We observe that because of the time-consuming local searches in the creation of the initial population the HMA was not able to finish the initialization within the timelimit for some instances, so we made further tests with an increased timelimit of 1,800 s. The results of these tests can be found in Table 13 for $n = 100$ and Table 14 for $n = 150$ and $n = 200$. In Table 13 we also show the results of the modified iterative exact method (MEM) by Alekseeva and Kochetov (2013) and the results of the branch-and-cut by Roboredo and Pessoa (2013). For more than 100 customers no results of exact methods are published in the literature. From Table 13 we conclude that our algorithm is able to find optimal solutions to all but one instance, with $n = 100$ but in much less time than the exact algorithms. Tables 14 and 15 shows that the described approach is still superior and exceeds the HMA in most instances. The HMA can compete with the GA on some of the instances with $n = 150$ and even gets better mean objective values for 3 instances, e.g., *Code5_150w_rp15*. However, the differences are rather small and for $n = 200$ the GA is better in 28 out of 30 instances with a much lower standard deviation on most instances.

It is interesting that although our algorithm did not find the optimal solution for instance *Code511* with $r = p = 15$ it always terminated with the same suboptimal solution. This is due to its solution evaluation method because even though the optimal value of the LP relaxation and the optimal value to the follower problem often coincide, it is not the case here. During our runs the algorithm might have visited the optimal solution but it was not able to identify it because its objective value is only approximated by the LP evaluation and was therefore discarded later.

8.6 Uniform instances

Next, we tested our algorithm on the Uniform instances and compared the results with the VNS and STS by Davydov et al. (2014a), who tested their algorithms on a Pentium Intel Core Dual with 2.66 GHz. They published only the result of one single run (obj_1). Since we perform 30 runs, it is not straightforward to compare these

Table 13 Comparison of the results from instances with $n = 100$ of the so far best exact methods MEM by Kochetov et al. (2013) and B&C by Roboredo and Pessoa (2013), the so far best heuristic methods HMA and STS by Alekseeva and Kochetov (2013) and Davydov et al. (2014a), respectively, and our GA + solA + ML-ES + imp

	B & C		MEM		HMA			STS		GA + solA + ML-ES + imp. TS		
Instance	$\overline{obj}$	$t^*[s]$	$\overline{obj}$	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$	obj_1	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$
Code111w_rp10	**4,361.00**	10,217.0	**4,361.00**	3,600.0	**4,361.00**	0.00	97.4	**4,361.00**	63.7	**4,361.00**	0.00	11.2
Code111w_rp15	**4,596.00**	9,752.0	**4,596.00**	4,320.0	**4,596.00**	0.00	118.4	**4,596.00**	173.3	**4,596.00**	0.00	16.3
Code111w_rp20	**4,512.00**	>36,000	**4,512.00**[a]	60.0	4,511.47	1.38	492.9	4,484.00	118.1	**4,512.00**	0.00	159.4
Code211w_rp10	**5,310.00**	9,488.8	**5,310.00**	2,520.0	**5,310.00**	0.00	34.6	**5,310.00**	23.5	**5,310.00**	0.00	8.1
Code211w_rp15	**5,373.00**	80,956.4	**5,373.00**	230,700.0	**5,373.00**	0.00	116.1	**5,373.00**	88.9	**5,373.00**	0.00	18.0
Code211w_rp20	**5,432.00**	>36,000	**5,432.00**[a]	11,100.0	**5,432.00**	0.00	284.0	**5,432.00**	289.2	**5,432.00**	0.00	75.4
Code311w_rp10	**4,483.00**	19,071.3	**4,483.00**	8,760.0	**4,483.00**	0.00	38.8	**4,483.00**	33.8	**4,483.00**	0.00	8.2
Code311w_rp15	**4,800.00**	27,707.3	**4,800.00**	23,700.0	**4,800.00**	0.00	120.5	**4,800.00**	91.1	**4,800.00**	0.00	18.9
Code311w_rp20	**4,893.00**	>36,000	**4,893.00**[a]	14,880.0	4,892.93	0.37	297.7	**4,893.00**	211.3	**4,893.00**	0.00	85.4
Code411w_rp10	**4,994.00**	13,743.9	**4,994.00**	1,980.0	**4,994.00**	0.00	34.1	**4,994.00**	19.3	**4,994.00**	0.00	8.0
Code411w_rp15	**5,064.00**	84,140.1	**5,064.00**	73,380.0	**5,064.00**	0.00	206.9	**5,064.00**	121.3	**5,064.00**	0.00	54.5
Code411w_rp20	**5,209.00**	>36,000	**5,209.00**[a]	300.0	**5,209.00**	0.00	259.6	**5,209.00**	288.9	**5,209.00**	0.00	46.8
Code511w_rp10	**4,906.00**	80,413.8	**4,906.00**	23,940.0	**4,906.00**	0.00	75.1	**4,906.00**	27.2	**4,906.00**	0.00	11.4
Code511w_rp15	**5,131.00**	79,099.6	**5,131.00**	127,200.0	5,130.67	1.49	579.1	**5,131.00**	216.2	5,123.00	0.00	58.2
Code511w_rp20	**5,334.00**	>36,000	**5,334.00**[a]	6,600.0	**5,334.00**	0.00	274.4	**5,334.00**	133.2	**5,334.00**	0.00	60.9
Code611w_rp10	**4,595.00**	51,583.2	**4,595.00**	8,580.0	**4,595.00**	0.00	154.6	**4,595.00**	44.5	**4,595.00**	0.00	21.6
Code611w_rp15	**4,881.00**	28,342.7	**4,881.00**	137,580.0	**4,881.00**	0.00	52.5	**4,881.00**	114.8	**4,881.00**	0.00	17.2
Code611w_rp20	**4,952.00**	>36,000	**4,952.00**[a]	11,400.0	**4,952.00**	0.00	281.6	4,944.00	198.1	**4,952.00**	0.00	58.1
Code711w_rp10	**5,586.00**	20,352.7	**5,586.00**	4,380.0	**5,586.00**	0.00	108.3	**5,586.00**	101.0	**5,586.00**	0.00	9.2
Code711w_rp15	**5,827.00**	48,600.5	**5,827.00**	79,200.0	**5,827.00**	0.00	172.6	**5,827.00**	210.9	**5,827.00**	0.00	33.8
Code711w_rp20	**5,893.00**	>36,000	**5,893.00**[a]	5,820.0	**5,893.00**	0.00	168.7	**5,893.00**	254.3	**5,893.00**	0.00	21.5

Table 13 continued

	B & C		MEM		HMA			STS		GA + solA + ML-ES + imp. TS		
Instance	$\overline{obj}$	$t^*[s]$	$\overline{obj}$	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$	obj_1	$t^*[s]$	$\overline{obj}$	sd	$t^*[s]$
Code811w_rp10	**4,609.00**	26,808.0	**4,609.00**	9,120.0	**4,609.00**	0.00	139.2	**4,609.00**	27.2	**4,609.00**	0.00	18.8
Code811w_rp15	**4,675.00**	115,183.5	**4,675.00**	274,200.0	**4,675.00**	0.00	266.8	**4,675.00**	123.4	**4,675.00**	0.00	52.1
Code811w_rp20	**4,858.00**	>36,000	**4,858.00**[a]	34,200.0	**4,858.00**	0.00	370.3	**4,858.00**	118.8	**4,858.00**	0.00	22.9
Code911w_rp10	**5,302.00**	2,377.9	**5,302.00**	360.0	**5,302.00**	0.00	30.6	**5,302.00**	19.2	**5,302.00**	0.00	7.5
Code911w_rp15	**5,158.00**	>36,000	**5,158.00**	>36,000	**5,158.00**	0.00	338.8	**5,158.00**	157.8	**5,158.00**	0.00	240.1
Code911w_rp20	**5,459.00**	>36,000	**5,459.00**[a]	9,900.0	5,458.90	0.55	350.6	5,455.00	202.2	**5,459.00**	0.00	146.2
Code1011w_rp10	**5,005.00**	33,765.1	**5,005.00**	5,820.0	**5,005.00**	0.00	120.0	**5,005.00**	103.5	**5,005.00**	0.00	15.0
Code1011w_rp15	**5,195.00**	72,034.4	**5,195.00**	>36,000	**5,195.00**	0.00	223.8	**5,195.00**	48.2	**5,195.00**	0.00	28.2
Code1011w_rp20	**5,399.00**	>36,000	**5,399.00**[a]	7,800.0	**5,399.00**	0.00	188.4	**5,399.00**	184.4	**5,399.00**	0.00	28.0
Geometric mean	**5,045.18**		**5,045.18**		5,045.15			5,043.74		5,044.92		
#Best results	**30**		**30**		26			27		29		
#Unique best results	0		0		0			0		0		

TS with a runtime of 1,800 s

[a] Time needed for finding solutions that are within 5 % of the optimum, i.e., the optimality is not proven

Table 14 Comparison of the results from instances with $n = 150$ and $n = 200$ of HMA and our GA + solA + ML-ES + imp.TS with a runtime of 1,800 s

	HMA		GA + solA + ML-ES + imp.TS			HMA		GA + solA + ML-ES + imp.TS	
Instance	$\overline{obj}$	*sd*	$\overline{obj}$	*sd*	Instance	$\overline{obj}$	*sd*	$\overline{obj}$	*sd*
Code1_150w_rp10	**7,180.00**	0.00	**7,180.00**	0.00	Code1_200w_rp10	9,575.27	24.62	**9,598.00**	0.00
Code1_150w_rp15	7,153.90	0.31	**7,154.00**	0.00	Code1_200w_rp15	10,107.73	31.02	**10,130.00**	0.00
Code1_150w_rp20	7,249.70	0.47	**7,249.90**	0.31	Code1_200w_rp20	9,858.93	31.22	**9,894.33**	24.19
Code2_150w_rp10	**7,337.00**	0.00	**7,337.00**	0.00	Code2_200w_rp10	9,325.33	40.56	**9,333.80**	35.42
Code2_150w_rp15	7,744.10	3.45	**7,745.00**	0.00	Code2_200w_rp15	9,615.30	23.20	**9,633.80**	6.57
Code2_150w_rp20	7,804.90	8.76	**7,809.40**	3.29	Code2_200w_rp20	9,685.57	27.10	**9,702.27**	20.60
Code3_150w_rp10	**7,299.00**	0.00	**7,299.00**	0.00	Code3_200w_rp10	9,378.57	10.85	**9,382.00**	0.00
Code3_150w_rp15	7,603.40	2.28	**7,604.00**	0.00	Code3_200w_rp15	9,362.63	11.78	**9,371.00**	0.00
Code3_150w_rp20	**7,648.00**	0.00	7,647.47	2.92	Code3_200w_rp20	9,334.60	39.89	**9,352.00**	58.98
Code4_150w_rp10	**7,318.00**	0.00	**7,318.00**	0.00	Code4_200w_rp10	8,893.73	9.19	**8,897.00**	0.00
Code4_150w_rp15	**7,409.00**	0.00	**7,409.00**	0.00	Code4_200w_rp15	9,174.73	14.64	**9,185.00**	0.00
Code4_150w_rp20	7,927.00	3.81	**7,928.00**	0.00	Code4_200w_rp20	9,458.97	23.79	**9,473.83**	6.39
Code5_150w_rp10	**6,975.00**	0.00	**6,975.00**	0.00	Code5_200w_rp10	9,266.47	28.84	**9,281.27**	9.49
Code5_150w_rp15	**7,162.13**	8.99	7,158.93	15.76	Code5_200w_rp15	**9,286.63**	30.70	9,259.57	52.77
Code5_150w_rp20	7,324.87	4.01	**7,326.77**	2.87	Code5_200w_rp20	9,541.27	23.04	**9,556.00**	0.00
Code6_150w_rp10	7,049.67	1.83	**7,050.00**	0.00	Code6_200w_rp10	9,847.47	8.89	**9,852.00**	0.00
Code6_150w_rp15	**7,186.00**	0.00	**7,186.00**	0.00	Code6_200w_rp15	10,141.63	18.54	**10,155.27**	9.03
Code6_150w_rp20	7,383.83	6.61	**7,386.00**	0.00	Code6_200w_rp20	10,333.63	30.55	**10,350.83**	39.81
Code7_150w_rp10	**6,250.30**	1.29	6,248.00	0.00	Code7_200w_rp10	9,252.13	33.59	**9,277.00**	0.00
Code7_150w_rp15	**6,840.00**	0.00	**6,840.00**	0.00	Code7_200w_rp15	9,570.43	16.92	**9,588.00**	0.00
Code7_150w_rp20	7,295.77	6.76	**7,297.00**	0.00	Code7_200w_rp20	9,930.83	31.85	**9,952.33**	17.29
Code8_150w_rp10	**7,732.00**	0.00	**7,732.00**	0.00	Code8_200w_rp10	9,092.77	4.93	**9,093.00**	0.00

Table 14 continued

	HMA		GA + solA + ML-ES + imp.TS			HMA		GA + solA + ML-ES + imp.TS	
Instance	$\overline{obj}$	*sd*	$\overline{obj}$	*sd*	Instance	$\overline{obj}$	*sd*	$\overline{obj}$	*sd*
Code8_150w_rp15	**7,662.00**	0.00	**7,662.00**	0.00	Code8_200w_rp15	9,069.63	21.61	**9,085.80**	21.12
Code8_150w_rp20	7,850.77	1.28	**7,851.00**	0.00	Code8_200w_rp20	9,326.77	30.95	**9,348.77**	18.64
Code9_150w_rp10	**6,855.00**	0.00	**6,855.00**	0.00	Code9_200w_rp10	9,012.60	1.04	**9,013.00**	0.00
Code9_150w_rp15	6,883.30	1.21	**6,883.80**	0.61	Code9_200w_rp15	9,160.20	20.13	**9,174.07**	10.59
Code9_150w_rp20	7,185.30	4.67	**7,187.00**	3.81	Code9_200w_rp20	9,450.97	13.26	**9,462.67**	1.83
Code10_150w_rp10	**6,715.00**	0.00	**6,715.00**	0.00	Code10_200w_rp10	9,406.97	4.80	**9,411.00**	0.00
Code10_150w_rp15	**7,014.00**	0.00	**7,014.00**	0.00	Code10_200w_rp15	9,323.43	34.26	**9,353.80**	25.02
Code10_150w_rp20	7,205.13	2.73	**7,206.00**	0.00	Code10_200w_rp20	**9,751.03**	18.42	9,742.53	30.86
Geometric mean	7,265.37		**7,265.68**			9,478.43		**9,490.76**	
#Best results	16		**27**			2		**28**	
#Unique best res.	3		**14**			2		**28**	

Table 15 Results of Wilcoxon rank sum tests with error levels of 5 % for HMA and the GA with longer runtime for all 90 Euclidean test instances

	HMA	GA + solA + ML-ES + improved TS	Σ
HMA	–	3	3
GA + solA + ML-ES + improved TS	**31**	–	**31**

approaches. Therefore we list for our algorithm the average objective value ($\overline{obj}$) and the objective value of the best run (obj^*).

When using the configuration that performed best in the Euclidean instances (GA + solA + ML-ES + imp. TS) we see in Table 16 that for the instances where $r = p = 7$ the algorithm quickly converges to a non-optimal solution. We observed that on Uniform instances the case occurs more frequently where a good LP value does not necessarily lead to a good (or optimal) solution. To further investigate this issue we modified the ML-ES to solve the follower's problem exactly instead of only using the LP evaluation (GA + solA + ML-ES(EE) + imp. TS). Indeed, although the runtime increases, with this modification the GA was able to find the same solutions as STS for all but one instances with $r = p = 7$. Compared to the VNS, better solutions were found for five of ten instances but more time was required. STS performed excellent on these instances and was able to find equally good or better solutions in less time. However, the best of the 30 runs for GA + solA + ML-ES(EE) + imp. TS identifies a better or equally good solution for each instance.

When considering the instances where $w_j = 1,\ \forall j \in J$ and $r = p = 25, 30$ the modification of ML-ES is apparently not beneficial because the runtime increases and the average objective value is often equal but sometimes even worse. The results in Table 16 confirm the observation of Davydov et al. (2014a) that when r and p increase at least for these instances the problem gets easier since GA + solA + ML-ES + imp. TS obtains equally good results as the VNS with a low standard deviation. For instances with $r = p = 25$ we observe the largest deviation of the objective values. While the results of the GA are often within 3 % of the VNS results in 9 out of 10 instances, they are generally worse and in one case it is equally good. However, also for these cases the best run for each instance found a solution that is equally good or better.

From these results we conclude that especially for the Uniform instances the search order of the neighborhoods is important to speed up the algorithm. It seems that it is often unnecessary to search through the whole swap neighborhood but to consider promising moves first. An interesting extension to the presented algorithm would be to incorporate the VNS into a hybrid GA, e.g., we could replace the swap neighborhood for the TS with the neighborhoods of the VNS. The combination of the strengths of both the special neighborhood structure of the VNS and STS with the hybrid GA could potentially lead to an algorithm that performs well on both Euclidean and Uniform instances.

8.7 Distribution of facilities

Finally, we want to investigate how solutions to the problem may look like and what properties they might have. Although we cannot make general statements that apply

Table 16 Comparison of results from Uniform instances of VNS and STS by Davydov et al. (2014a) with our algorithm GA + solA + ML-ES + imp. TS and our modification ML-ES(EE)

	VNS		STS		GA + solA + ML-ES + imp. TS				GA + solA + ML-ES(EE) + imp. TS			
Instance	obj_1	$t^*[s]$	obj_1	$t^*[s]$	obj^*	$\overline{obj}$	sd	$t^*[s]$	obj^*	$\overline{obj}$	sd	$t^*[s]$
123Comp-Unif_rp7	5,009.00	304.17	5,009.00	65.09	4,904.00	4,904.00	0.00	54.10	5,009.00	5,009.00	0.00	504.00
223Comp-Unif_rp7	5,459.00	182.91	5,459.00	63.18	5,459.00	5,459.00	0.00	38.40	5,459.00	5,459.00	0.00	245.50
323Comp-Unif_rp7	5,009.00	145.01	5,019.00	54.69	5,003.00	5,003.00	0.00	53.10	5,019.00	5,019.00	0.00	445.40
423Comp-Unif_rp7	4,908.00	296.63	4,908.00	145.22	4,846.00	4,846.00	0.00	58.80	4,908.00	4,908.00	0.00	641.80
523Comp-Unif_rp7	5,198.00	292.05	5,208.00	22.63	5,206.00	5,206.00	0.00	57.90	5,208.00	5,208.00	0.00	374.10
623Comp-Unif_rp7	5,032.00	296.52	5,032.00	197.08	5,032.00	5,032.00	0.00	51.00	5,032.00	5,028.50	19.17	386.70
723Comp-Unif_rp7	5,055.00	286.04	5,055.00	62.23	4,962.00	4,962.00	0.00	46.80	5,055.00	5,055.00	0.00	446.10
823Comp-Unif_rp7	4,860.00	295.77	4,951.00	74.49	4,847.00	4,847.00	0.00	83.30	4,951.00	4,951.00	0.00	441.30
923Comp-Unif_rp7	5,060.00	217.70	5,127.00	111.27	5,127.00	5,127.00	0.00	80.20	5,127.00	5,127.00	0.00	825.00
1023Comp-Unif_rp7	5,067.00	322.48	5,084.00	278.18	5,000.00	5,000.00	0.00	55.20	5,084.00	5,084.00	0.00	510.70
$w_j = 1.\ \forall j \in J$												
123Comp-Unif_rp25	62.00	16.12	62.00	30.27	62.00	61.20	1.27	89.90	62.00	61.30	1.09	154.90
223Comp-Unif_rp25	62.00	157.40	62.00	38.84	61.00	60.70	0.47	96.40	62.00	60.77	0.57	160.30
323Comp-Unif_rp25	61.00	34.60	61.00	47.48	61.00	60.20	1.30	106.60	61.00	59.50	1.83	172.70
423Comp-Unif_rp25	59.00	25.59	59.00	17.26	59.00	58.60	0.89	112.90	59.00	58.80	0.48	155.00
523Comp-Unif_rp25	63.00	70.91	63.00	41.00	63.00	61.90	0.92	95.00	63.00	61.90	0.84	164.80
623Comp-Unif_rp25	62.00	16.39	61.00	35.61	62.00	60.13	1.48	91.00	62.00	60.07	1.26	148.00
723Comp-Unif_rp25	66.00	15.69	66.00	19.42	66.00	65.87	0.73	110.40	66.00	65.87	0.73	172.90
823Comp-Unif_rp25	60.00	32.98	60.00	19.38	60.00	58.20	1.16	91.60	60.00	58.43	1.07	145.00
923Comp-Unif_rp25	63.00	18.96	63.00	17.43	63.00	61.70	1.18	98.60	63.00	61.43	1.38	164.60
1023Comp-Unif_rp25	65.00	15.49	65.00	21.58	65.00	65.00	0.00	91.90	65.00	65.00	0.00	153.70
123Comp-Unif_rp30	70.00	15.73	69.00	50.97	70.00	70.00	0.00	116.50	70.00	70.00	0.00	182.30

Table 16 continued

Instance	VNS		STS		GA + solA + ML-ES + imp. TS				GA + solA + ML-ES(EE) + imp. TS			
	obj_1	$t^*[s]$	obj_1	$t^*[s]$	obj^*	$\overline{obj}$	sd	$t^*[s]$	obj^*	$\overline{obj}$	sd	$t^*[s]$
223Comp-Unif_rp30	65.00	15.81	65.00	67.87	65.00	65.00	0.00	103.40	65.00	65.00	0.00	167.30
323Comp-Unif_rp30	65.00	15.82	65.00	86.92	65.00	65.00	0.00	114.10	65.00	65.00	0.00	174.40
423Comp-Unif_rp30	65.00	16.09	65.00	91.13	65.00	65.00	0.00	119.40	65.00	65.00	0.00	197.00
523Comp-Unif_rp30	69.00	15.58	69.00	76.31	69.00	69.00	0.00	114.80	69.00	69.00	0.00	185.20
623Comp-Unif_rp30	69.00	15.94	69.00	78.44	69.00	69.00	0.00	111.70	69.00	69.00	0.00	179.90
723Comp-Unif_rp30	72.00	15.64	72.00	74.20	72.00	72.00	0.00	113.40	72.00	72.00	0.00	182.80
823Comp-Unif_rp30	62.00	15.64	62.00	80.56	62.00	62.00	0.00	91.30	62.00	61.77	0.73	146.20
923Comp-Unif_rp30	68.00	15.42	68.00	59.36	68.00	68.00	0.00	107.50	68.00	68.00	0.00	170.00
1023Comp-Unif_rp30	73.00	15.42	71.00	126.19	73.00	73.00	0.00	115.20	73.00	73.00	0.00	176.30
Geometric mean	277.42		277.24		276.76	275.48			277.78	276.24		

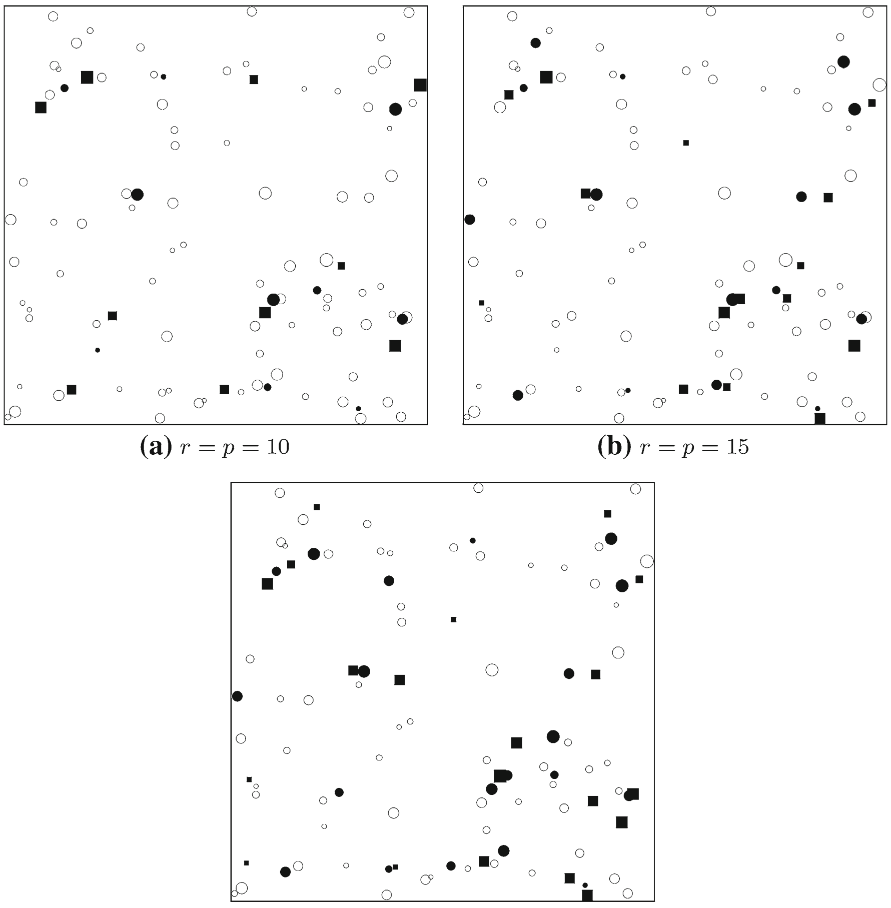

Fig. 3 Optimal solutions for instance Code311w with different r and p values. **a** $r = p = 10$. **b** $r = p = 15$. **c** $r = p = 20$

to all problem instances, we show in Fig. 3 the graphical representation of optimal solutions for one instance with 100 customers (Code311w) with different r and p values. The circles are customer locations, the filled points stand for locations chosen by the leader and the rectangles represent facilities of the follower. The size of the symbols depends on the demand of the corresponding location, i.e., the larger the symbol the higher the demand.

In Fig. 3 it seems that the leader tends to choose locations that are more or less evenly spreaded across the whole region with a focus on the more crowded areas. The follower then appears to prefer locations in the vicinity of a leader's facility. Another interesting observation is that while some locations are picked each time by the leader for different r and p values, some other locations are not always chosen. Visualizations on other instances reveal similar patterns, but as said before it is hard to draw precise general conclusions.

9 Conclusions

In this work we proposed a genetic algorithm for the discrete RPCP with several enhancements. First of all, a trie-based solution archive was used to reduce the number of unnecessary solution evaluations and to overcome premature convergence. This led to a significant efficiency gain. Another important part of the algorithm was the embedded local improvement procedure. Several ways of combining the local search with the solution archive were investigated, and the reduced neighborhood was identified to work best in practice. Different solution evaluation methods were considered and we found an effective way to combine them, which led to the multi-level evaluation scheme. Finally we improved the results of our algorithm by using a tabu search for local improvement.

Extensive tests showed that the performance on Euclidean instances is very good but for Uniform instances STS/VNS is better since they often obtain better results than our algorithm in average in less time. For many of the commonly used Euclidean instances GA + solA + ML-ES + improved TS is able to exceed previous state-of-the-art heuristic approaches and scales well to larger instances that cannot be solved with today's exact methods anymore.

We considered here only one variant of a competitive facility location problem. For future work it would be interesting if our approach will also succeed when some problem parameters are changed, e.g., if the demand of the customers is proportional to the distance to the facilities or if it is inelastic. Further research also includes other applications of the solution archive, which is expected to improve the performance of algorithms for problems that have a compact solution representation and an expensive evaluation method. The tree structure of the solution archive might also be exploited further, e.g., by computing bounds on partial solution in order to cut off subspaces, that cannot contain better solutions.

Acknowledgments This work is supported by the Austrian Science Fund (FWF) under Grant P24660-N23.

References

Alekseeva, E., Kochetov, Y.: Matheuristics and exact methods for the discrete $(r|p)$-centroid problem. In: Talbi, E.G. (ed.) Metaheuristics for Bi-Level Optimization, Studies in Computational Intelligence, vol. 482, pp. 189–219. Springer, Berlin (2013)

Alekseeva, E., Kochetova, N., Kochetov, Y., Plyasunov, A.: A hybrid memetic algorithm for the competitive p-median problem. In: Bakhtadze, N., Dolgui, A. (eds.) Information Control Problems in Manufacturing, vol. 13, pp. 1533–1537. International Federation of Automatic Control, Salvador (2009)

Alekseeva, E., Kochetova, N., Kochetov, Y., Plyasunov, A.: Heuristic and exact methods for the discrete $(r|p)$-centroid problem. In: Cowling, P., Merz, P. (eds.) Evolutionary Computation in Combinatorial Optimization, LNCS, vol. 6022, pp. 11–22. Springer, Berlin (2010)

Bhadury, J., Eiselt, H., Jaramillo, J.: An alternating heuristic for medianoid and centroid problems in the plane. Comput. Oper. Res. **30**(4), 553–565 (2003)

Biesinger, B., Hu, B., Raidl, G.: An evolutionary algorithm for the leader-follower facility location problem with proportional customer behavior. In: Pardalos, P.M., Resende, M.G., Vogiatzis, C., Walteros, J.L. (eds.) Learning and Intelligent Optimization, Lecture Notes in Computer Science, pp. 203–217. Springer, Berlin (2014)

Campos-Rodríguez, C., Moreno-Pérez, J., Noltemeier, H., Santos-Peñate, D.: Two-swarm pso for competitive location problems. In: Krasnogor, N., Melián-Batista, M., Pérez, J., Moreno-Vega, J., Pelta, D. (eds.) Nature Inspired Cooperative Strategies for Optimization (NICSO 2008), Studies in Computational Intelligence, vol. 236, pp. 115–126. Springer, Berlin (2009)

Campos-Rodríguez, C., Moreno-Pérez, J.A., Santos-Peñate, D.: Particle swarm optimization with two swarms for the discrete $(r|p)$-centroid problem. In: Moreno-Díaz, R., Pichler, F., Quesada-Arencibia, A. (eds.) Computer Aided Systems Theory (EUROCAST 2011), LNCS, vol. 6927, pp. 432–439. Springer, Berlin (2012)

Davydov, I., Kochetov, Y., Carrizosa, E.: VNS heuristic for the centroid problem on the plane. Electron. Notes Discret. Math. **39**, 5–12 (2012)

Davydov, I., Kochetov, Y., Mladenovic, N., Urosevic, D.: Fast metaheuristics for the discrete $(r|p)$-centroid problem. Autom. Remote Control **75**(4), 677–687 (2014a)

Davydov, I., Kochetov, Y., Plyasunov, A.: On the complexity of the $(r|p)$-centroid problem in the plane. TOP **22**(2), 614–623 (2014b)

Drezner, T.: Optimal continuous location of a retail facility, facility attractiveness, and market share: an interactive model. J. Retail. **70**(1), 49–64 (1994)

Drezner, T.: Location of multiple retail facilities with limited budget constraints in continuous space. J. Retail. Consum. Serv. **5**(3), 173–184 (1998)

Drezner, T., Drezner, Z., Salhi, S.: Solving the multiple competitive facilities location problem. Eur. J. Oper. Res. **142**(1), 138–151 (2002)

Ghosh, A., Craig, C.: A location allocation model for facility planning in a competitive environment. Geogr. Anal. **16**(1), 39–51 (1984)

Gusfield, D.: Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology. Cambridge University Press, New York (1997)

Hakimi, S.: On locating new facilities in a competitive environment. Eur. J. Oper. Res. **12**(1), 29–35 (1983)

Hotelling, H.: Stability in competition. Econ. J. **39**(153), 41–57 (1929)

Hu, B., Raidl, G.: An evolutionary algorithm with solution archives and bounding extension for the generalized minimum spanning tree problem. In: Soule, T. (ed.) Proceedings of the 14th annual conference on genetic and evolutionary computation (GECCO), pp. 393–400. ACM Press, Philadelphia (2012)

Kochetov, Y., Kochetova, N., Plyasunov, A.: A matheuristic for the leader-follower facility location and design problem. In: Lau H, Van Hentenryck P, Raidl G (eds) Proceedings of the 10th metaheuristics international conference (MIC 2013), Singapore, pp 32/1–32/3 (2013)

Kress, D., Pesch, E.: Sequential competitive location on networks. Eur. J. Oper. Res. **217**(3), 483–499 (2012)

Laporte, G., Benati, S.: Tabu search algorithms for the $(r|x_p)$-medianoid and $(r|p)$-centroid problems. Locat. Sci. **2**, 193–204 (1994)

Mladenović, N., Brimberg, J., Hansen, P., Moreno-Pérez, J.A.: The p-median problem: A survey of metaheuristic approaches. Eur. J. Oper. Res. **179**(3), 927–939 (2007)

Noltemeier, H., Spoerhase, J., Wirth, H.C.: Multiple voting location and single voting location on trees. Eur. J. Oper. Res. **181**(2), 654–667 (2007)

Raidl, G., Hu, B.: Enhancing genetic algorithms by a trie-based complete solution archive. In: Cowling, P., Merz, P. (eds.) Evolutionary Computation in Combinatorial Optimization, LNCS, vol. 6022, pp. 239–251. Springer, Berlin (2010)

Roboredo, M., Pessoa, A.: A branch-and-cut algorithm for the discrete $(r|p)$-centroid problem. Eur. J. Oper. Res. **224**(1), 101–109 (2013)

Serra, D., Revelle, C.: Competitive location in discrete space. In: Economics working papers 96, Department of Economics and Business, University of Pompeu Fabra, Spain, Technical Report (1994)