Memetic Computing
https://doi.org/10.1007/s12293-018-0256-7

REGULAR RESEARCH PAPER

Baldwin effect and Lamarckian evolution in a memetic algorithm for Euclidean Steiner tree problem

Michał Bereta[1]

Received: 10 December 2016 / Accepted: 20 March 2018

Abstract
The study is concerned with the Baldwin effect and Lamarckian evolution in a memetic algorithm for Euclidean Steiner tree problem (ESTP). The main aim is to examine the importance of the proposed local search procedures for the ability of the algorithm to find solutions to ESTP. Two local search procedures are proposed as a part of an evolutionary algorithm for ESTP, and the effect of their use is carefully analyzed. The correction procedure modifies the given chromosome so that it satisfies the degree and angle conditions in ESTP. The improvement procedure actively adds new Steiner points to the evaluated solution. The modified solutions are accepted with varying probability, leading to a spectrum of algorithms with a Baldwin effect on one end and Lamarckian evolution on the other. The results are carefully analyzed using proper statistical procedures (Friedman test and post-hoc procedures). To further check the ability of the proposed algorithm to find the optimal or near optimal solutions, results for problems from OR-Lib are provided.

Keywords Baldwin effect · Euclidean Steiner tree problem · Evolutionary algorithm · Genetic algorithm · Lamarckian evolution · Memetic algorithm

1 Introduction

Let p be the number of points in n-dimensional Euclidean space. The goal in the Euclidean Steiner tree problem (ESTP) is to minimize the sum of lengths of edges spanning these points. However, unlike in the minimum spanning tree (MST) problem, additional points, called Steiner points, are allowed to be added to the original set of points, called terminals. The number and locations of these additional points are difficult to find, and the problem is known to be np-complete. As an example, Fig. 1 presents a set of points in two-dimensional space which are given as terminals. In (a) they are connected to form the minimum spanning tree. Also presented (b) is an example Steiner tree which includes additional points (Steiner points). It can be observed that the addition of new points made the total sum of edges' lengths to decrease.

The Euclidean Steiner tree problem (ESTP) originated from the question formulated by Fermat (in "*Treatise on Minima and Maxima*"): how to find a point with a minimum distance from other three points that are given? Later the problem has been generalized into a problem of interconnecting a given set of points on a plane with a minimum length.

From the practical point of view, ESTP on a plane and other versions of Steiner tree problem are of wide interest. The application areas include such problems as VLSI chips layout design, distribution network planning, wireless networks, sensor networks or modeling of molecular structures. For a review of application areas of ESTP, the Reader is referred to the most up to date [1].

1.1 Related work

Exact and heuristic algorithms exist for solving ESTP. Examples of exact algorithms for ESTP are the algorithms of Melzak [2,3], Trietsch and Hwang [4] or the well known Smith algorithm [5]. Smith also suggested a generalization of his algorithm to higher dimensions. There is still a need for new heuristics. Exact algorithms for ESTP are computationally demanding, which limits their use for bigger problems, i.e., for problems with a bigger number of terminals. Although a huge breakthrough has been made in the

✉ Michał Bereta
mbereta@pk.edu.pl

[1] Institute of Computer Science, Cracow University of Technology, ul. Warszawska 24, 31-155 Kraków, Poland

Published online: 28 March 2018

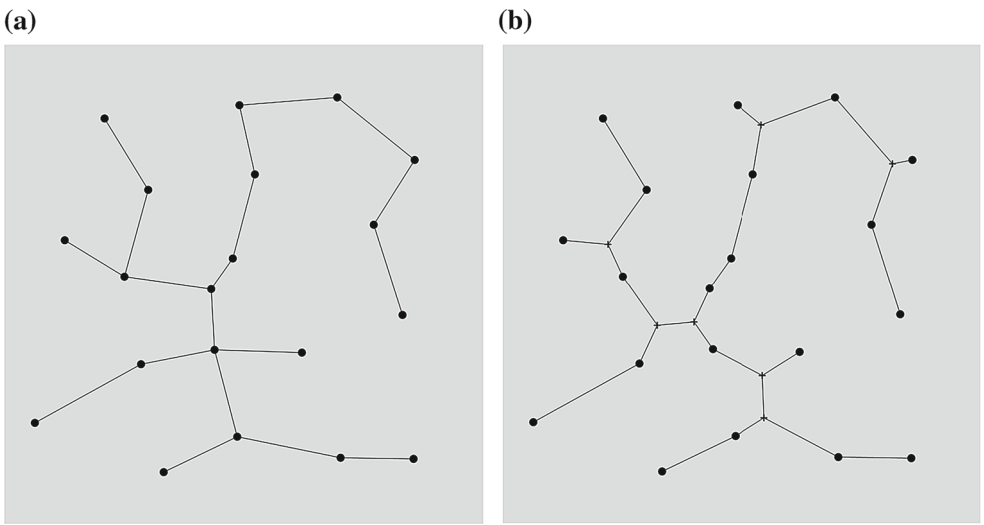

Fig. 1 **a** Minimum spanning tree for a set o terminals marked as circles; **b** Steiner tree constructed as the minimum spanning tree after adding additional points (Steiner points marked as +)

field [1], heuristic algorithms have also been proposed to solve ESTP.

A wide class of heuristic algorithms, which can also be applied to ESTP, are evolutionary algorithms. The general approach is to maintain a population of potential solutions, which are gradually improved by promoting more promising ones and applying differentiation operators to them (such as crossover or mutation) to create a new generation of potentially better solutions. Depending on the problem to be solved, the solutions are encoded using a proper data structure.

Several evolutionary approaches have already been proposed to address various types of Steiner problem, not only on Euclidean plane. Genetic algorithms (GA) have been the most popular among them; example approaches are [6,7]. There were also some other approaches such as particle swarm optimization, ant colony optimization, an artificial immune algorithm or more recently physarum optimization [8,9]. However, to the best of Author's knowledge, this work is the first attempt to analyze the importance of the Baldwin effect and Lamarckian evolution in an evolutionary algorithm applied to ESTP. From all of the cited works, the most similar approach was presented in [7]. The algorithm for relocating Steiner points proposed in [7] is somehow akin to the correction procedure proposed in this work. However, the whole new design of the evolutionary algorithm with improvement procedure proposed here and the main goal of testing the Baldwin effect and Lamarckian evolution make this study unique.

1.2 Memetic algorithms

There are several design decisions when an evolutionary algorithm is to be applied to the given problem. Two problems are particularly important in this work. The first one is that in some problems it is hard to generate even a feasible solution. Usually, an evolutionary algorithm starts with a population of randomly generated individuals. In case they are not feasible solutions, an additional repair procedure can be applied. It is possible in some problems to calculate the fitness function for such infeasible solutions in the hope that after several generations of evolutionary algorithm they will become not only feasible but also optimal. The second problem is that evolutionary algorithms are known to be good at estimating the promising areas in the search space relatively fast. However, the fine-tuning process of these solutions may take a long time. In the case of ESTP, when applying an evolutionary algorithm to solve it, both of the aforementioned problems are present and seem to be crucial. It is a straightforward procedure to generate a set of random points and encode it as an individual in an evolutionary algorithm. However, the probability that these points satisfy the conditions necessary to be a proper solution for the ESTP is slight.

In general, even if the solutions in the evolutionary algorithm are feasible, an additional procedure can be applied to each individual from the population to increase the convergence speed. This procedure can be in the form of a greedy optimization heuristic (local search procedure), and its introduction into the evolutionary framework brings the idea of a memetic algorithm (MA) and more general, memetic computation.

MAs [10] originate from pairing evolutionary framework and a set of local improvement procedures, often called local search, lifetime learning or individual learning. Local improvement method, considered on its own, often represents a greedy optimization which is usually not sufficient to find satisfactory solutions on its own. More generally, memes are essential units of knowledge about the problem being solved.

When incorporated into the evolutionary algorithm they can speed up its convergence especially in the case of complex optimization problems.

The successful application of the concept of combining evolutionary search with memes representing problem specific knowledge brought enormous research in the area known as memetic computing. That led to algorithmic constructs far beyond the idea of simple improvement step as a part of the evolutionary algorithm. According to [10] current research in the field of memetic computing can be categorized into simple hybrids (with a MA as a representative), adaptive hybrids and a memetic automaton. Another, generational description of MA has been presented in [11]. The first generation of memetic computing include algorithms resulting from the mere pairing of global search algorithm (potentially evolutionary) with local search. MAs from the second and third generations include algorithms based on pairing global search with several local optimizers. The models become more and more complicated, and the ideas used to build new memetic computing paradigms include passing to offspring the choice of local optimizer (second generation) or self-organization and coevolution (third generation).

A detailed description of all the problems concerning the design of an algorithm based on memetic computation ideas is far beyond the scope of this introduction. Memetic computation is strongly connected with the problem being solved. The results depend primarily on the strategic choices concerning the level of hybridization (such as preprocessing schemes, solution representation, problem specific search operators), population diversity management, frequency and intensity of refinement and modes of inheritance (Baldwin effect and Lamarckian learning). In the case of adaptive memetic computation, the number of factors is even bigger.

Several works describe successful applications of MA to different hard optimization problems concerned with path planning or graph structures optimizations. Examples are path planning for mobile robots with bacterial MA [12], traveling car renter problem [13] or vertex coloring problem [14]. Other combinatorial optimization problems, such as job-shop scheduling problem [15] or knapsack problem [16], have also been successfully approached by MA. These examples show a great potential of memetic computing when dealing with hard optimization problems similar to ESTP.

1.3 Baldwin effect and Lamarckian evolution

In the case of the MA with a local search procedure, there are several approaches to how the corrected solutions should be used. One approach is to use the fitness function of the corrected solution to evaluate the original solution. However, the original solution is not replaced by the corrected one. This approach is described as Baldwin effect in the evolutionary process [17]. On the other hand, the corrected solution can replace the original one in the population. This approach constitutes what is called a Lamarckian evolution [17]. Both methods have their advantages but are also potentially problematic, and it is hard to assess which one is better for a given problem. For example, Lamarckian evolution can limit the population diversity very quickly, which is not desired in general. An intermediate approach is possible, in which only some percentage of the corrected solutions is accepted as the replacement of the original solutions. This method opens up many possibilities to fine tune the evolutionary algorithm.

Both, Baldwin effect and Lamarckian evolution, have been used in memetic computations. Lamarckian learning [18,19] has been reported as a better approach to unimodal optimization problems and in problems in which the environment is static. On the other hand, Baldwin effect has been reported as the preferred method for problems in dynamic environments [20,21]. There have also been some studies suggesting that a mixed Lamarckian–Baldwinian approach could be beneficial [22]. Thus, it is not possible to assess which approach is universally better, and the comparison between the Baldwin effect and Lamarckian evolution has to be performed in the context of the given problem. One of the goals of this paper is to study the usefulness of Baldwin effect and Lamarckian evolution in the context of the ESTP.

1.4 Contribution of this work

The algorithm proposed in this work can be classified as a simple hybrid type of MA. It is based on a GA framework with crossover operator and three mutation operators. It is observed that the GA on its own is not efficient in providing not only optimal but even feasible solutions of ESTP. For that reason, ESTP specific knowledge is introduced. The memes are represented as two local search procedures specifically designed for the ESTP. The main contribution of this study is to verify the usefulness of the proposed memes to improve the performance of the GA. The second contribution is the comparison of a spectrum of different modes of inheritance ranging from Baldwin effect to Lamarckian learning for the ESTP. As stated in the previous section, such comparisons strongly depend on the problem being solved. It is expected that the results presented in this study will be helpful in designing MA for ESTP especially concerning the choice between Baldwin effect and Lamarckian evolution. The straightforward application of the greedy heuristic is also included in the comparison.

The paper is organized as follows. In Sect. 2, the most important facts about ESTP are provided, which allow defining a correction procedure and a simple greedy heuristic to improve an existing candidate solution to ESTP. In Sect. 3, a MA for 2D ESTP including the repair and local search procedures is derived. In Sect. 4, the results of empirical studies on 2D ESTP problems of various sizes are presented together

with a proper statistical analysis. Conclusions are drawn in Sect. 5.

2 Euclidean Steiner tree problem

Properties of Steiner trees on the 2D plane have been studied extensively [1]. The most important properties used in this work are as follows.

- Degree property. In a Steiner tree, each Steiner point is incident with exactly three edges thus having the degree of three.
- Angle property. The three edges incident with a Steiner point make exactly 120° with each other.
- Steiner minimal tree (SMT) for a set of p terminals has at most $p-2$ Steiner points.
- Each terminal in SMT has at most two edges incident with it.

2.1 Fermat point

The Fermat point of the triangle on the Euclidean plane is a point minimizing the total sum of distances of triangle vertices to that point. Thus, for any other point F' on the plane, the condition (1) is satisfied.

$$|F'A| + |F'B| + F'C| > |FA| + |FB| + |FC| \tag{1}$$

where F is a Fermat point, and A, B, and C are triangle's vertices.

For a simple case of three points on the Euclidean plane, the SMT can be found easily with the help of the Fermat point. Two situations are possible. If the three points form a triangle in which all angles have less than 120°, then the only Steiner point is located at the Fermat point of the triangle. The SMT is equivalent to the minimum spanning tree (MST) including three triangle points and the Fermat point. In the second case, if there is an angle equal to or greater than 120°, the Steiner point is incident with the triangle vertex in which the two sides of the triangle meet at 120 or more degrees. The SMT is the MST spanned on the three triangle points.

The procedure to find Fermat point for three points is given as follows (Fig. 2).

Procedure 1 FindFermatPoint

Input:: Three points A, B and C on the Euclidean plane

Output: Fermat point

1. If $\measuredangle ABC \geq 120°$ or $\measuredangle BAC \geq 120°$ or $\measuredangle ACB \geq 120°$, return B, A or C, respectively.
2. Select any two sides of the triangle.

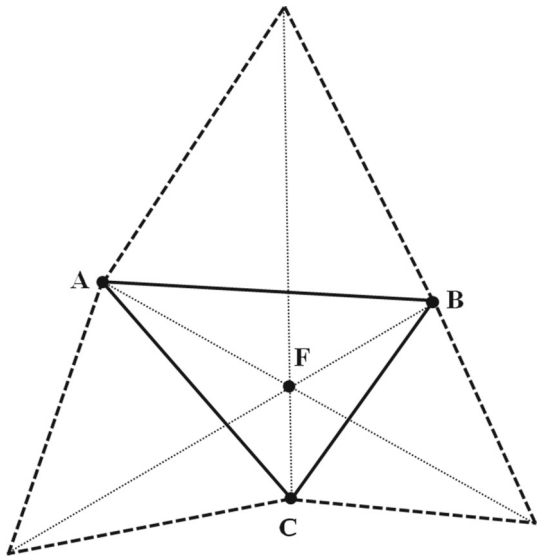

Fig. 2 Construction of the Fermat point F of the triangle ABC

3. Construct an equilateral triangle on each of the chosen sides.
4. Find two lines, going through each new vertex and its opposite vertex of the original triangle.
5. The Fermat point is the point of intersection of the two lines.

End of Procedure 1

In the subsequent part of this work only Fermat points which are not incident with the triangle point are useful in the proposed heuristic. For that reason, for the purpose of this work, such not useful Fermat points will be called *not valid* Fermat points.

2.2 Measure for the quality of the solutions

A standard measure to compare solutions for ESTP is called Steiner ratio. It is given by (2):

$$\rho = \frac{L\,(ST)}{L\,(MST)}, \tag{2}$$

where L(.) is the total length of the edges of a given tree, MST is the minimum spanning tree for the terminals and ST is the Steiner tree calculated as the minimum spanning tree for the set of points consisting of the union of the terminals and a given set of Steiner points. It is worth to mention that although Steiner points have to satisfy the degree and angle conditions mentioned earlier in this section, the Steiner ratio measure can be used to compare solutions obtained for any given set of points used as Steiner points. This interpretation allows using the Steiner ratio as a fitness function in the proposed evolutionary algorithm.

2.3 Simple heuristic for generation of candidate solutions

Finding the Fermat point for a set of three terminals is a straightforward way to find an SMT for these three points. For a bigger number of terminals, the situation is not as simple. However, based on the simple procedure of finding the Fermat point, a simple heuristic can be formulated which iteratively generates candidate Steiner points as Fermat points for groups of selected three points from a given problem. If the addition of such candidate point improves the Steiner ratio, the Fermat point is accepted as a new Steiner point. This step can be repeated until a stopping criterion is satisfied. Procedures 2 and 3 describe the idea more in detail.

Procedure 2 FindSteinerPoint

Input: $Z = T \cup S$—set of points (T—terminals; S - already added Steiner points), MST_Z –minimum spanning tree for Z, nn—number of nearest neighbors to look for.

Output: *true* if a new Steiner point f has been found and S was modified, *false* otherwise.

1. Find a random point $s = (s_x, s_y)$ on a plane, where $\min_i z_{ix} \leq s_x \leq \max_i z_{ix}$, and $\min_i z_{iy} \leq s_y \leq \max_i z_{iy}$, where $i = 1..N$ and N is a number of points in Z, $z_i \in Z$.
2. Find nn nearest neighbors of the point s from among the points in Z.
3. Randomly select three points from among the nn nearest neighbors.
4. Find the Fermat point f for the three selected points.
5. If f is a valid Steiner point, let $Z' = ZU\{f\}$ and $MST_{Z'}$ be the minimum spanning tree for Z'; otherwise return *false*.
6. If necessary, correct Z'.
7. If $\mathrm{L}(MST_{Z'}) < L(MST_Z)$ let $Z = Z'$ return *true*, otherwise return *false*.

End of Procedure 2

The meaning of step 6 in Procedure 2 will be explained in the later part of this section.

Procedure 3 Improve Solution

Input: T—set of terminals, S—set of Steiner points, *max_iters* – maximum number of iterations, *max_tries* – maximum number of iterations without improvement, nn—the number of nearest neighbors to look for in procedure ***FindSteinerPoint***.

Output: S—set of Steiner points.

1. Let $i = 0$.
2. Let $Z = T \cup S$ and MST_Z be the minimum spanning tree for Z.
3. $i = i + 1$.
4. Call procedure ***FindSteinerPoint***(Z, MST_Z, nn).
5. If procedure ***FindSteinerPoint*** returned *false* for the last *max_tries* iterations, return.
6. If $i < max\_iters$ repeat from step 2.

End of Procedure 3

In the second (or any further) iteration of procedure ***ImproveSolution***, it is possible that after the addition of point f and calculating the new minimum spanning tree $MST_{Z'}$ in step 5 of ***FindSteinerPoint***, some of the other Steiner points may no longer satisfy the angle and/or degree condition. This is especially possible when f is connected with already existing Steiner points in $MST_{Z'}$. In such cases, a correction procedure ***CorrectSolution*** can be applied.

The correction procedure is also simple and straightforward. Its main idea is first to identify Steiner points which do not satisfy the degree condition and remove them. If the degree condition is satisfied for all remaining Steiner points, the Steiner point which violates the angle condition the most is identified and recalculated as a Fermat point of the triangle formed by the three points connected to it. These two steps are repeated until no change is necessary. The Procedure 4 ***CorrectSolution*** shows the idea in more detail.

Procedure 4 CorrectSolution

Input: T—set of terminals, S—set of Steiner points (some of them potentially invalid), ε—maximum allowed violation of angle condition for Steiner points.

Output: S—set of valid Steiner points

1. Let $Z = T \cup S$ and MST_Z be the minimum spanning tree for Z.
2. Identify Steiner points in S which have degrees not equal 3 in MST_Z and remove them from S; recalculate MST_Z.
3. If any Steiner point has been eliminated in step 2, go to step 2.
4. Identify Steiner point s in S which violates the angle condition the most. If this violation is more than ε, recalculate this point as a Fermat point of the triangle formed by the three points connected to s (if the Fermat point is not valid, remove s); recalculate MST_Z.
5. If S has been modified in step 4, go to step 4.
6. If S has been modified in step 2 or step 4, go to step 2.
7. Return S.

End of procedure correctsolution

As an example, in Fig. 3 two steps of the improvement procedure are presented. The first Steiner point has been added in (b) to the set of terminals from (a). This Steiner point satisfies the degree and angle conditions and the correction procedure introduces no changes. After adding the second Steiner point

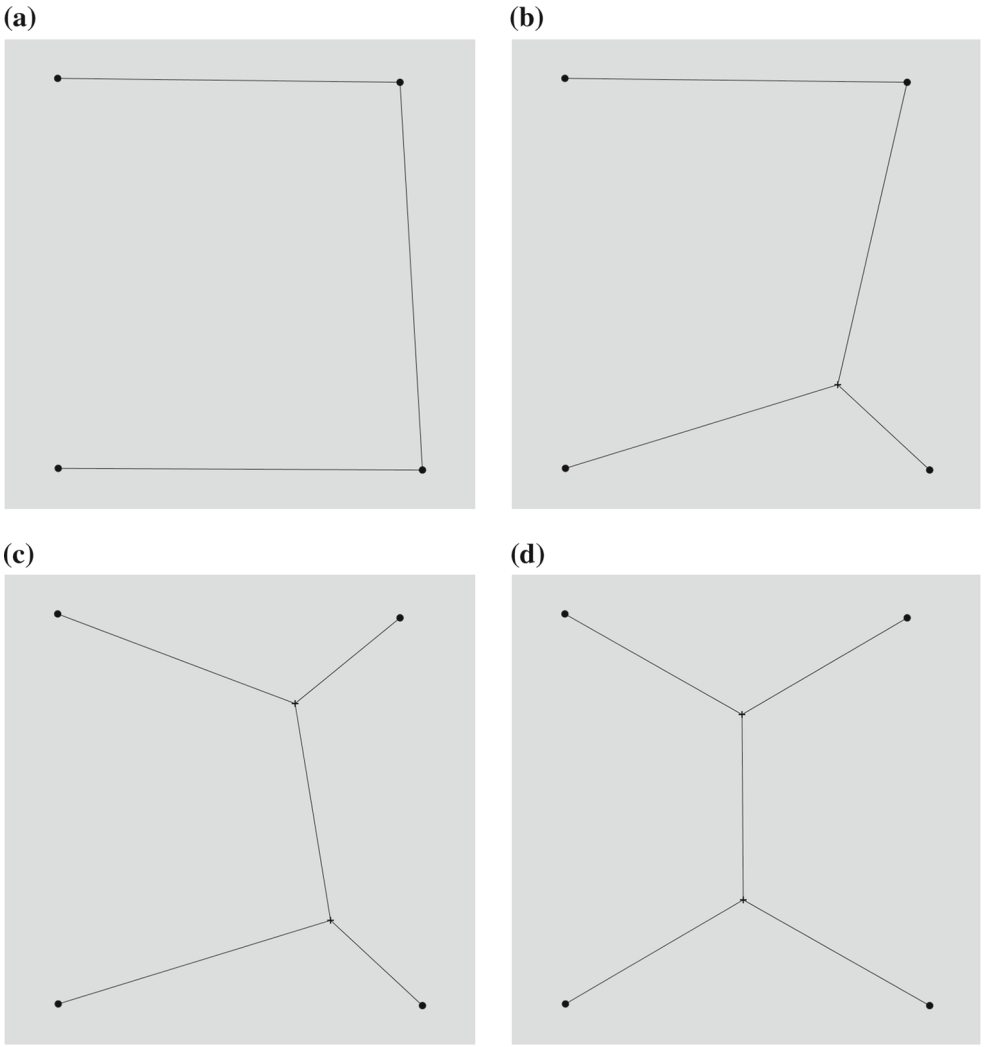

Fig. 3 Two steps of the improvement procedure: **a** MST on a set of terminals, **b** the first Steiner point is added, **c** the second Steiner point is added making the first Steiner point no longer satisfy the angle condition, **d** the correction procedure shifts both Steiner points into the correct locations (both Steiner points satisfy the angle condition)

and recalculating the MST, the first one no longer satisfies the angle condition, but it is still a point of 3 degrees (c). In (d) the situation after the correction procedure is shown, where both Steiner points have been shifted into the correct locations and satisfy the degree and angle conditions.

It should be stressed out that procedure ***FindSteinerPoint*** can modify the set S by adding a new Steiner point but also by removing some of the already existing ones using the correction procedure. However, it is done only if it is beneficial, i.e., the overall Steiner ratio decreases, which means that a better solution is found.

The proposed heuristic can be seen as a greedy optimization algorithm. Due to the randomness in step 1 of procedure ***FindSteinerPoint,*** it can be started several times in the hope that a better solution will be found. In fact, in the section describing numerical experiments, this algorithm was compared to the GA proposed in this work.

The main purpose of introducing these simple heuristics for correcting and improving ESTP solutions is to use them as local search procedures in the MA described in Sect. 3. The proposed algorithm will be tested on the effects of the usage of these correction and improvement heuristics.

The drawback of the proposed heuristic is the necessity of recalculating MST after any change is introduced to set S, by adding, removing or recalculating any of the elements of S. However, there exist efficient algorithms for calculating MST. In this work, the Prim algorithm has been used. Additionally, properly designed data structures can limit the calculations needed to find closest points in the Prim algorithm.

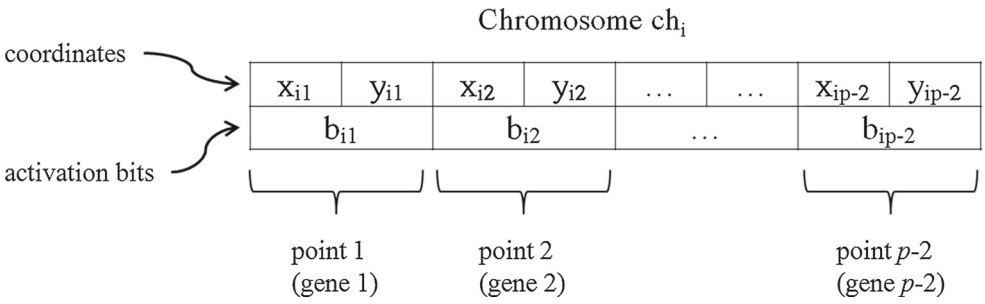

Fig. 4 The data structure used to encode a set of candidate Steiner points as a chromosome in the GA

3 Memetic algorithm for Euclidean Steiner tree problem on a plane

The MA proposed in this work is based on the most basic GA which includes selection, crossover and mutation steps. However, it is adapted to the ESTP. The most important characteristics of the proposed algorithm are as follows. (1) The properly designed data structure which allows encoding a set of two-dimensional points evolving into the Steiner points. (2) Managing the population of solutions, each encoded by the data structure. (3) Tournament selection step. (4) One-point crossover operator. (5) Three mutation operators. (6) Local search procedure (the correction procedure or the improvement procedure described in Sect. 2). (7) The balance between the Baldwin effect and Lamarckian evolution is achieved using a probability of acceptance of the modified solution into the population. (8) Usage of Steiner ratio as the fitness function (ESTP is treated as a minimization problem).

3.1 Chromosome structure

For a given set T of p terminals, the solution to ESTP is a set of Steiner points. Not only location of these points is not known but also the number of them. As described in Sect. 2, for ESTP with p points, at most p-2 Steiner points can exist in the optimal solution. For that reason, the data structure used to encode a single solution (Fig. 4) allows encoding a maximum number of $p-2$ 2D points, however additional bits are used to indicate which "genes" are active. That allows modeling the problems in which the solution consists of a number of Steiner points smaller than $p-2$. Still, the length of the chromosome is constant. However, the only candidate Steiner points that are used to calculate MST are the points with the activation bits set to *true*. The coordinates of the points are encoded as real values. The ith chromosome in the population can be described as

$$ch_i = [(x_{i1}, y_{i1}), \ldots, (x_{ip-2}, y_{ip-2}), (b_{i1}, b_{i2}, \ldots, b_{ip-2})] \tag{3}$$

where $(x_{ij}, y_{ij}) \in \mathbb{R}^2$ are the coordinates of j-th candidate Steiner point, b_{ij} is the corresponding activation bit and $j = 1, \ldots, p-2$.

3.2 The proposed memetic algorithm

The main loop of the proposed algorithm is summarized in Fig. 5. Apart from the proposed MA, the GA is also presented. It is given to alleviate the understanding of the role of the local search in the proposed MA, which is designed on the basis of the GA from Fig. 5. In fact, the GA is also used in the numerical tests described in Sect. 4 (denoted as *enorm*).

The main difference between the GA and the MA is the evaluation step. In the case of the GA, the candidate Steiner points encoded by the active genes in the chromosome are combined with the terminal points of the ESTP being solved, and the Steiner ratio of this solution is calculated and used as the evaluation of the chromosome. In the case of the MA, after extracting of candidate Steiner points from active genes, there is an approach to correct/improve this set of candidate points using the local search procedure. The local search algorithm can be either the correction procedure (Procedure 4) or the improvement procedure (Procedure 3). Which one of them is used, is a parameter of the algorithm and it is set before the start of the algorithm. It does not change during the execution of the given run. The Steiner ratio of the corrected/improved solution is used as the evaluation of the original chromosome. Additionally, to compare the usefulness of the Baldwin effect and Lamarckian evolution, the corrected/improved genes can be accepted into the initial chromosome with a probability of *prob_change*, which is a parameter of the algorithm. By setting *prob_change* to 0, we only consider the Baldwin effect, while setting it to 1, the evolution is of Lamarckian type. However, intermediate values can also be considered. The results of the numerical experiments presented in Sect. 4 show the importance of the proper setting of *prob_change*.

The next section gives a detailed description of all of the steps of the proposed MA.

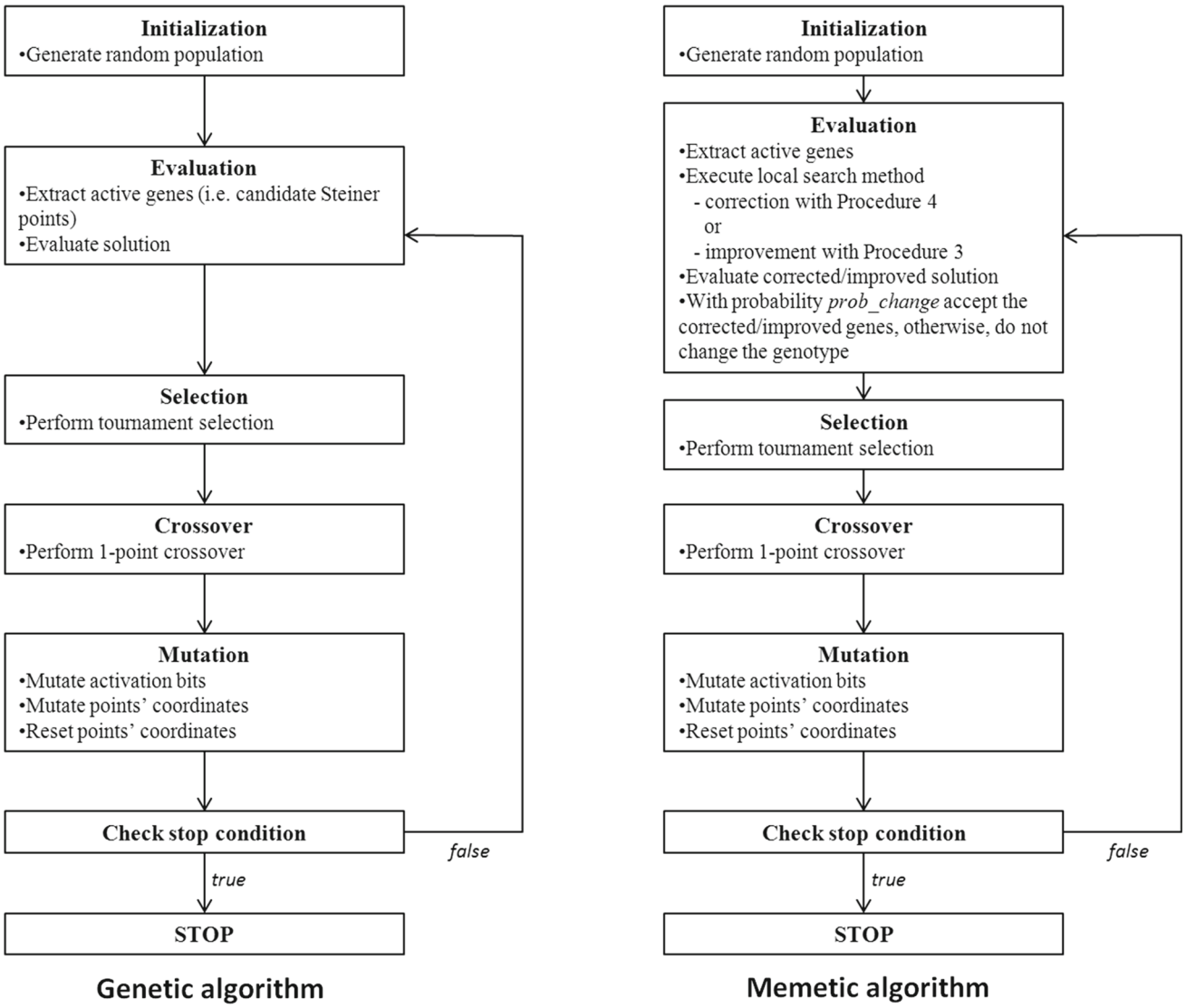

Fig. 5 The main loop of the proposed MA

3.3 Detailed description of the proposed algorithm

In what follows a detailed description of all the steps of the MA (Fig. 5) is provided. Most of the steps are the same for the GA, and the only difference is during the evaluation step.

First, during the initialization, the population of random solutions (chromosomes) is generated. For ith chromosome the coordinates of the candidate jth Steiner point are generated according to (4) and (5):

$$x_{ij} = urand \in \left[\min_{t=(t_x,t_y)\in T} t_x, \quad \max_{t=(t_x,t_y)\in T} t_x\right] \quad (4)$$

$$y_{ij} = urand \in \left[\min_{t=(t_x,t_y)\in T} t_y, \quad \max_{t=(t_x,t_y)\in T} t_y\right] \quad (5)$$

where T is a set of terminals, and *urand* is a random value generated from a uniform distribution. The activation bits are set to *true* with probability *abits_init_prob*.

Then, until the termination condition is satisfied, the following steps are repeated.

1. Evaluation of solutions. This step is the only one in which the proposed MA is different from the GA. In what follows, it is clearly stated, which steps are executed only in the case of the MA.
   Firstly, for both, genetic and MA, for each chromosome, only those candidate Steiner points with activation bits set to *true* are selected and combined with the set of terminals T. MST of the combined points becomes a candidate solution.
   Only for the GA
   a. Fitness value (Steiner ratio) of the solution becomes the fitness value of the chromosome.

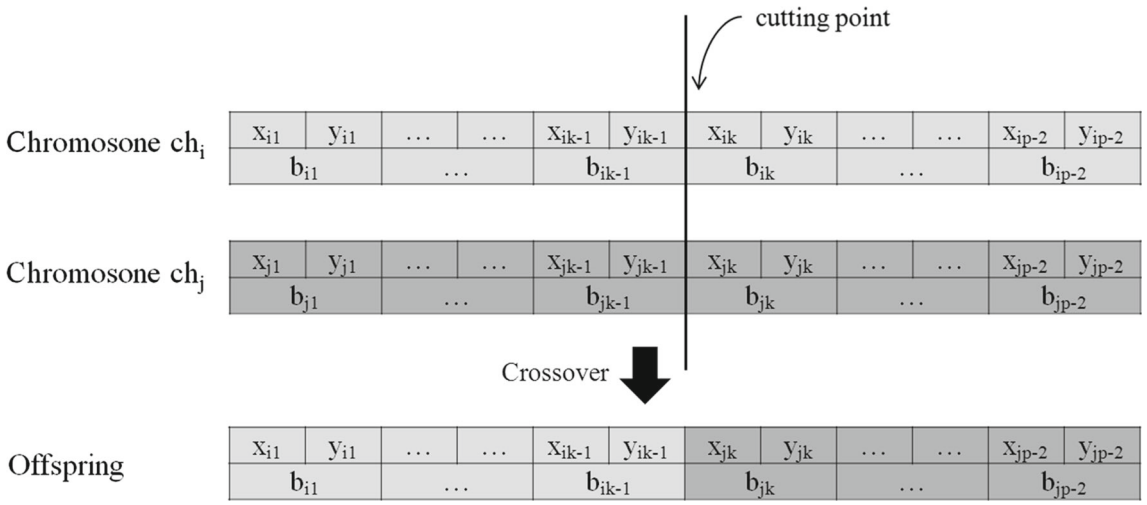

Fig. 6 Visualization of the crossover operator

Only for the MA

a. Application of local search method. There are two options, correcting the solution with Procedure 4 or improving it with Procedure 3. The choice is made before the algorithm starts, and it is fixed for a given run of the algorithm.
b. Fitness value (Steiner ratio) of the corrected/improved solution becomes the fitness value of the original chromosome.
c. With probability *prob_change* the original chromosome accepts the corrected/improved genes (Steiner points).

For both, GA and MA, if the best solution from the current population is better than the best one found until now, remember the new best solution. The remembered best solution is always corrected to guarantee that the output of the algorithm satisfies the degree and angle conditions.

2. Tournament selection is performed. The number of selected solutions equals the size of the population. The selected solutions are put on the list in the order in which they were selected.
3. Crossover step is performed. For each selected chromosome, with probability *prob_cross* it is crossed over with the next chromosome on the list of selected chromosomes. Otherwise, it is copied with no change. In the case of the last chromosome on the list, it is crossed over with the first one on the list, if necessary.
4. Mutation of activation bits is performed. With probability *prob_mut_activation* each activation bit is inverted. This probability should not be large (e.g. 0.1 has been used in this work).
5. Mutation of candidate Steiner point positions is performed. For each point encoded in the chromosome data structure (both active and inactive), with probability *prob_mut_coordinates*, its coordinates are changed using a random perturbation based on uniform distribution. The maximum allowed change is limited by the parameter *mut_range*. A given point $s = (s_x, s_y)$ is mutated according to (6) and (7):

$$s_x \rightarrow s_x + range_x * (1 - 2 * urand) \tag{6}$$
$$s_y \rightarrow s_y + range_y * (1 - 2 * urand) \tag{7}$$

where

$$range_x = mut\_range * (\max t_x - \min t_x) \tag{8}$$
$$range_y = mut\_range * (\max t_y - \min t_y) \tag{9}$$

and $t = (t_x, t_y) \in T$, where T is a set of terminals in ESTP and *urand* is a random number from uniform distribution over [0,1].

6. The third mutation step is performed. With probability *prob_mut_reset*, each point is reset to new random coordinates according to (4) and (5). This mutation type should have a low probability as it introduces significant change into the chromosome.
7. Check the stopping condition, if *true*, finish the algorithm.

The crossover operator is a simple one-point crossover. First, a cutting point is generated, and the offspring is produced by interchanging the two resulting parts. The idea is presented in Fig. 6.

There are three mutation operators. This may seem to be too complicated, however, after some consideration each of these operators is justifiable. Switching the activation bits is crucial for the algorithm to adapt to the problems in which a different number of Steiner points are optimal. Mutating the positions of the candidate Steiner points is the most expected

mutation type and does not seem to require additional justification. The last mutation operator, resetting the coordinates to the random ones, may help to escape the algorithm from local minima. However, it should be used with a small probability as it is much damaging to the chromosome. It should be noted that the second and third mutation operator is executed not depending on the value of the activation bits, i.e., the given gene in the chromosome may be inactive and still mutated by changing the coordinates or resetting them.

Point 1c of the MA requires explanation. The problem is that the corrected or improved solution may have a different number of Steiner points than there are active bits in the current chromosome. For example, let us assume that there is an ESTP with ten terminals. Thus, each chromosome encodes a solution with eight candidate Steiner points and eight activation bits. Let us further assume that for a given chromosome only 4 activation bits are set to true. To evaluate the solution these four points are extracted from the data structure and combined with terminals in T. If the local search procedure is called, some of these points may be removed by the correction procedure. If an improvement procedure is called, some Steiner points may be added. If the decision is to replace the original chromosome by its corrected/improved version, the question is how to put the new set of Steiner points back into the data structure, in which there are still inactive genes.

Let us assume that in our example the correction procedure removed one candidate Steiner point and corrected the coordinates of the remaining three. This is not much problematic, and the simple strategy is to put the three corrected points exactly at the same positions in the data structure while the removed point becomes inactive (its activation bit is changed to *false*). Another approach would be to insert the corrected points into random locations of active positions and set the remaining active bits to false. The second approach has been used in this work.

Let us now assume that during the local search step the improvement procedure removed one candidate point, corrected the coordinates of three and added two new Steiner points. The problem is now how to include the new Steiner points into the chromosome data structure. There are several options. After some preliminary numerical experiments, the approach used in this work is as follows. First, the improved Steiner points are put randomly at the positions of active genes in the chromosome. Then, in case there are still Steiner points to be placed into the data structure, inactive positions are selected randomly, set to these remaining Steiner points and their activation bits are set to true.

4 Experimental studies

In this section, the results of numerical experiments are presented together with a proper statistical analysis. The main goal of these experiments is to measure the effect of Baldwin effect and Lamarckian evolution on the ability of the proposed MA to provide a solution to ESTP. The influence of two main factors is to be measured.

The first factor is the selection of the local search method. The correction procedure from Sect. 2 (Procedure 4) is the first choice. After application of this procedure for any set of candidate Steiner points (which may not satisfy the degree and angle conditions), the result is a set of points that satisfies the degree and angle conditions (to the extent limited by the parameter ε). In the case when the candidate points are almost satisfying the conditions, the change introduced by the correction procedure is very limited. In another case the set of candidate Steiner points may be modified significantly, e.g., some points can be removed. However, this procedure does not seek for new points that could improve the solution. For that reason, this local search method can be considered as a measure of a potential of the set of candidate Steiner points. The second choice is the improvement procedure (Procedure 3) which actively seeks new candidate Steiner points and uses the correction procedure as its part. The improvement procedure modifies the original set of points more significantly than the correction procedure. It measures the potential of the solution to improvement. However, taking into account the randomness of the improvement procedure, there is a danger that the resulting evaluation is of limited use as a fitness value of the original solution, as there is no guarantee, that the improvement procedure will be able to improve it again as much when started for the second time. There is no such danger in case of correction procedure which is deterministic. Thus, it can be expected, that the results when applying improvement procedure with small values of *prob_change* may not be satisfying.

The second factor is the selection of the probability *prob_change* of acceptance of the corrected/improved chromosome instead of the original one. If *prob_change* is 0, only the Baldwin effect is considered. If *prob_change* is 1, the Lamarckian evolution is in play. Setting *prob_change* to the value between 0 and 1 gives a full spectrum of possible intermediate approaches. It is not obvious that this value should be small or high. Higher values may increase the convergence, however, by limiting the population diversity, they may force the convergence to the local minima. Low values may not use the full benefits of incorporating local search methods, and there will be a slow convergence.

4.1 Description of problems

For the purpose of this study, three sets of random problems were generated on $[0, 1]^2$. There are three groups of 20 problems, with 10, 20 and 50 terminals, respectively. Example problem with 50 terminals is presented in Fig. 7. All

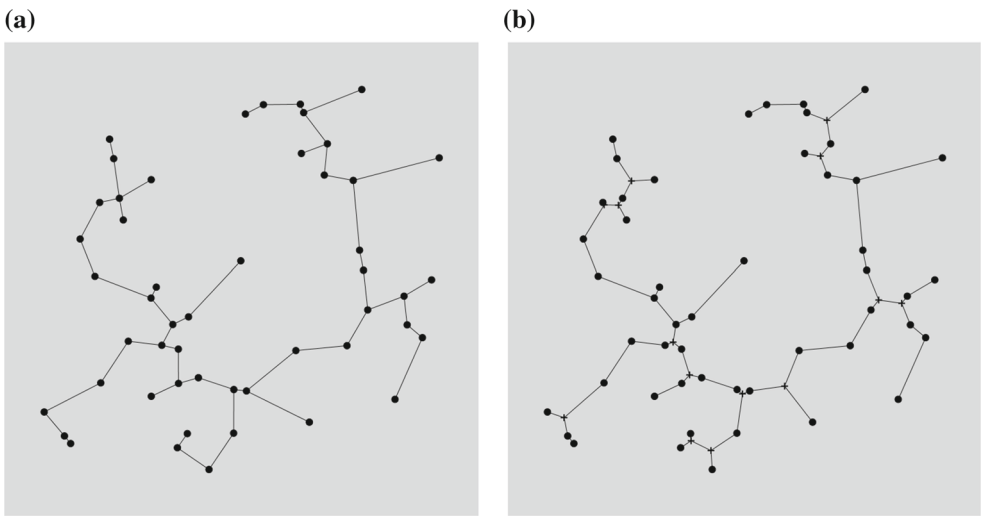

Fig. 7 Example test problem with 50 terminals (circles): **a** terminals connected in MST, **b** example solution with Steiner points (marked as +)

the problems and solutions are available upon request from Author.

4.2 Algorithms and settings

Several versions of the algorithm have been tested. First, the local search method is selected, then the probability of acceptance of the modified solution. For the correction procedure chosen as the local method, there are algorithms abbreviated as cp0.0, cp0.2, cp0.4, cp0.6, cp0.8 and cp1.0, where the values after the letters "cp" indicate the values of *prob_change*. For example, cp0.8 means that a correction procedure is used as the local search method and the modified chromosome is accepted with the probability 0.8. For the improvement procedure selected as the local search method, there are algorithms abbreviated as ip0.0, ip0.2, ip0.4, ip0.6, ip0.8 and ip1.0, where the values after the letter "ip" indicate the values of *prob_change*. Note that in all cases the Steiner ratio of the corrected/improved chromosome is used as the fitness function even in the cases when the original chromosome is not replaced.

Additionally, the GA with no local search method is tested for comparison. It means that the steps 1b, 1c, and 1d from the description of the algorithm in Sect. 3 are replaced by simply calculating the Steiner ratio for the points with activation bits set to true, even though they may not satisfy the degree and angle conditions. This allows testing the importance of any of the local search methods. The algorithm is abbreviated as enorm.

The improvement procedure (Procedure 3) is also included in the comparison. It allows testing the capability of this heuristic when used outside the evolutionary framework. In this case, the improvement procedure starts with an empty set of Steiner points and adds new candidate Steiner points one by one in a greedy way. Due to the usage of the correction procedure as a part of this improvement procedure, some of the already existing Steiner points can be removed, and the resulting dynamics can be treated as an interesting greedy optimization approach to ESTP. Three versions of the improvement procedure are tested, with the parameter *nn* (the number of the nearest neighbors in Procedure 3) equal 3, 5 and -1, where $nn = -1$ means that the three points used to calculate the Fermat point are randomly selected from among all available points.

Table 1 summarizes the algorithms tested together with the values of the parameters used. It should be observed that the improvement procedure is employed in the MA with $max\_iters = 1$, which means that there is an attempt to insert only one new candidate Steiner point. Thus, the usage of this local search method is limited to prevent the premature convergence and preserve the population diversity. Still, the parameter $max\_tries = 100$, which means that the improvement procedure actively searches for the new candidate Steiner point and stops only if it is not possible to generate new satisfying Fermat point (Procedure 2) for 100 attempts. On the other hand, algorithms nn3, nn5 and nn-1 look for multiple candidate Steiner points (parameter $max\_iters = 3^*|T| - 2$, where T is the set of terminals in a given ESTP). Although it is known that the maximum number of Steiner points can be at most $|T| - 2$ in the optimal solution, it should be remembered that the correction procedure (used by the improvement procedure) may remove some of the Steiner points after introducing a new one. Due to this dynamic character of the improvement procedure when $max\_iters > 1$, setting $max\_iters > |T| - 2$ may be beneficial.

The parameters for the evolutionary part of the MA have been set after several trial runs and have been observed to

Table 1 Algorithms and parameters

Alg.	Evolutionary framework	Local search method	Probability of acceptance of change solution ($change\_prob$)
cp0.0	Yes	Correction procedure ($\varepsilon = 1e{-}5$)	0.0
cp0.2			0.2
cp0.4			0.4
cp0.6			0.6
cp0.8			0.8
cp1.0			1.0
ip0.0		Improvement procedure ($nn = 3$, $max\_tries = 100$, $max\_iters = 1$, $\varepsilon = 1e{-}5$)	0.0
ip0.2			0.2
ip0.4			0.4
ip0.6			0.6
ip0.8			0.8
ip1.0			1.0
enorm		None	-
nn3	No	Improvement procedure ($max\_tries = 100$, $max\_iters = 3^*(\|T\| - 2)$, $\varepsilon = 1e{-}8$)	
nn5			
nn-1			

work satisfactorily. They may not be optimal. However, the important thing here is that they are common to all versions of the MA and the main aim of this study is to measure the differences resulting from different local search procedures and varying levels of the probability of acceptance of the corrected/improved solution. The final best solution from each run of the MA is finally corrected with $\varepsilon = 1e{-}8$, although $\varepsilon = 1e{-}5$ was used during the evolution to speed up the computations. This is done for a fair comparison with the results of nn3, nn5, and nn-1 algorithms, in which $\varepsilon = 1e{-}8$ has been used.

In case of the evolutionary framework the following settings are used: population size 80, number of iterations 200, tournament size 3, $abits\_init\_prob = 0.8$, $prob\_cross = 0.7$, $prob\_mut\_activation = 0.1$, $prob\_mut\_coordinates = 0.1$, $prob\_mut\_resest = 0.02$, $mut\_range = 0.1$.

4.3 Results for random problems

Each of the algorithms described in Sect. 4.2 has been run ten times for each ESTP considered. Here we report results for mean Steiner ratios of these ten runs for each problem. This is due to the statistical comparison framework that is used in this work. The main aim is to compare the average results of 16 algorithms over 20 ESTP problems in a given group of problems (10, 20 and 50 terminals). Friedman test with Shaffer's post hoc procedure is used to compare the mean Steiner ratios of each algorithm over a set of 20 problems in the group.

Friedman test is a type of non-parametric statistical test with the aim to discover differences in the mean results of several algorithms over several problems at once. This approach is much better than running several pair-wise statistical tests such as the sign tests or Wilcoxon test. The reason is that the probability of Type I error (discover a difference when there is, in fact, no difference between algorithms considered) is much smaller when running Friedman test. In the case of Friedman test, all algorithms being compared and all problems are taken up at once, and the pair-wise comparisons are performed only in case there are differences among the algorithms discovered by Friedman test. The ranking of algorithms is provided by the Friedman test, and the post-hoc procedure is used to control the Family-Wise Error Rate (FWER) for pair-wise comparisons. FWER reflects the probability of making one or more false discovering while performing pair-wise comparisons. For a much more detailed description of the Friedman test and the statistical comparison framework, the reader is referred to [23]. The results of Friedman test and Shaffer's post hoc procedure presented in this work are calculated by Keel software package [24].

It should be mentioned here, that one of the requirements for Friedman test is that the number of algorithms being compared should be smaller than the number of problems. Since in this work there are 16 algorithms and 20 problems in a group, this requirement is satisfied.

Table 2 Friedman's ranks for algorithms based on the mean results for problems with ten terminals

Algorithm	ip1.0	cp0.2	cp0.0	cp0.4	cp0.6	ip0.8	cp0.8	cp1.0	ip0.6	ip0.4	ip0.0	ip0.2	nn5	enorm	nn3	nn-1
Rank	4.175	4.55	4.6	5.875	6	6.5	6.75	7.175	9.275	9.675	10.15	10.625	12.05	12.05	12.95	13.6

Table 3 p values adjusted by Shaffer's procedure (problems with ten terminals)

	cp0.2	cp0.0	cp0.4	cp0.6	ip0.8	cp0.8	cp1.0	ip0.6	ip0.4	ip0.0	ip0.2	nn5	enorm	nn3	nn-1
ip1.0	10.561	10.561	8.283	7.450	5.146	4.186	2.639	0.057	**0.021**	**0.007**	**0.002**	**0.000**	**0.000**	**0.000**	**0.000**
cp0.2	–	10.561	9.729	9.729	7.224	5.758	3.981	0.125	0.054	**0.016**	**0.005**	**0.000**	**0.000**	**0.000**	**0.000**
cp0.0		–	9.729	9.729	7.450	5.978	4.186	0.137	0.061	**0.018**	**0.006**	**0.000**	**0.000**	**0.000**	**0.000**
cp0.4			–	10.561	10.561	10.561	9.729	1.459	0.789	0.325	0.125	**0.004**	**0.004**	**0.000**	**0.000**
cp0.6				–	10.561	10.561	9.729	1.776	0.981	0.421	0.153	**0.005**	**0.005**	**0.000**	**0.000**
ip0.8					–	10.561	10.561	3.396	2.027	0.997	0.424	**0.018**	**0.018**	**0.002**	**0.000**
cp0.8						–	10.561	4.302	2.706	1.459	0.694	**0.035**	**0.035**	**0.004**	**0.000**
cp1.0							–	6.196	4.356	2.696	1.338	0.095	0.095	**0.011**	**0.002**
ip0.6								–	10.561	10.561	9.729	3.396	3.396	0.981	0.293
ip0.4									–	10.561	10.561	5.046	5.046	1.776	0.630
ip0.0										–	10.561	7.450	7.450	3.272	1.338
ip0.2											–	9.729	9.729	5.146	2.696
nn5												–	10.561	10.561	9.400
enorm													–	10.561	9.400
nn3														–	10.561

p values which indicate significant differences (less than 0.05) are in bold

Problems with ten terminals

The Friedman's ranks achieved by each of the 16 algorithms for the 20 problems with ten terminals are presented in Table 2. These ranks are based on the average performance of the algorithms.

The first significant finding is that the idea of combining the evolutionary approach with local search procedures is justified. The "pure" evolutionary algorithm (enorm) and the greedy local search algorithms (nn3, nn5, nn-1) are performing poorly and have been assigned the worst ranks. On the other hand, the best performing algorithm is ip1.0, i.e., the MA with the improvement procedure as the local search method. It is interesting that its *prob_change* equals 1.0, which means that it is always beneficial to accept the improved solution. Algorithm ip1.0 is followed by several versions of evolutionary algorithms with the correction procedure as the local search method. In this case, it seems to be beneficial to use the Steiner ratio of the corrected solution as the fitness value of the original solution without changing it. What is more, from Table 3 it can be observed that the post hoc procedure is not able to detect significant (with significant level $\alpha = 0.05$) differences between the best performing algorithm (ip1.0) and algorithms cp0.2, cp0.0, cp0.4, cp0.6, cp0.8, and cp1.0. The p–values which indicate significant differences (less than $\alpha = 0.05$) are marked bold. Additionally, there is no difference detected between the best ip1.0 and algorithms ip0.8 and ip0.6. From this it can be concluded that the correction procedure can be used with Baldwin effect, Lamarckian evolution and any intermediate approach with no significant difference in averaged performance compared to the best performing one. On the other hand, when the improvement procedure is used, Lamarckian evolution is preferred. This can be explained by the observation that the improvement procedure modifies the solution (in a random way) much more than the correction procedure and using the Steiner ratio of the improved solution as the fitness value of the original solution is misleading to the evolutionary process. The correction procedure is better in estimating the potential of the unchanged solution and works better with low values of *change_prob*.

The evolutionary algorithm with no local search method (enorm) is performing poorly, which indicates the importance of the correction/improvement procedures. Algorithms ip1.0, cp0.2, cp0.0, cp0.4, cp0.6, ip0.8, and cp0.8 are performing significantly better than enorm. However, it can also be observed that by not selecting the *change_prob* value correctly, the resulting MA is not significantly different from the simple evolutionary algorithm enorm. There is no significant difference between enorm and algorithms cp1.0, ip0.6, ip0.4, ip0.0, ip0.2. To summarize, to achieve significant improvement by incorporating local search into the evolutionary framework, Lamarckian evolution should be used in case of improvement procedure, and Baldwin effect should be utilized in case of the correction procedure. A similar obser-

Table 4 Number of best solutions found (problems with ten terminals)

Algorithm	cp0.0	cp0.2	cp0.4	ip0.6	ip0.8	cp0.8	cp1.0	ip0.2	ip0.4	ip1.0	nn5	cp0.6	nn3	ip0.0	nn-1	enorm
Number of best solutions	19	19	19	19	19	18	18	18	18	18	18	17	17	15	15	8

Table 5 Friedman's ranks for algorithms based on the mean results for problems with 20 terminals

Algorithm	ip1.0	ip0.8	cp0.0	ip0.6	ip0.4	cp0.2	cp0.4	ip0.2	cp0.6	ip0.0	cp0.8	cp1.0	nn5	nn3	enorm	nn-1
Rank	1.65	2.975	4.85	5.8	6.325	6.725	7.5	7.9	8	9	9.275	9.95	11.65	14.2	15.1	15.1

Table 6 p values adjusted by Shaffer's procedure (problems with 20 terminals)

	ip0.8	cp0.0	ip0.6	ip0.4	cp0.2	cp0.4	ip0.2	cp0.6	ip0.0	cp0.8	cp1.0	nn5	nn3	enorm	nn-1
ip1.0	7.955	1.644	0.356	0.126	0.052	**0.008**	**0.003**	**0.002**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
ip0.8	-	6.603	2.666	1.330	0.739	0.172	0.074	0.058	**0.005**	**0.002**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
cp0.0		-	7.955	7.658	6.603	3.292	1.968	1.711	0.356	0.201	0.051	**0.001**	**0.000**	**0.000**	**0.000**
ip0.6			-	7.955	7.955	7.247	5.544	5.038	1.644	1.092	0.356	**0.008**	**0.000**	**0.000**	**0.000**
ip0.4				-	7.955	7.955	7.387	7.247	3.251	2.253	0.883	**0.029**	**0.000**	**0.000**	**0.000**
cp0.2					-	7.955	7.955	7.955	4.708	3.613	1.577	0.074	**0.000**	**0.000**	**0.000**
cp0.4						-	7.955	7.955	7.658	6.914	3.939	0.356	**0.001**	**0.000**	**0.000**
ip0.2							-	7.955	7.955	7.944	5.719	0.739	**0.002**	**0.000**	**0.000**
cp0.6								-	7.955	7.955	6.248	0.859	**0.003**	**0.000**	**0.000**
ip0.0									-	7.955	7.955	3.292	**0.040**	**0.004**	**0.004**
cp0.8										-	7.955	4.243	0.074	**0.009**	**0.009**
cp1.0											-	7.247	0.290	**0.045**	**0.045**
nn5												-	3.613	1.141	1.141
nn3													-	7.955	7.955
enorm														-	7.955

p values which indicate significant differences (less than 0.05) are in bold

vation can be done for the local search methods: any of the algorithms ip1.0, cp0.2, cp0.0, cp0.4, cp0.6, ip0.8, and cp0.8 is significantly better than any of the algorithms nn3, nn5, and nn-1.

It should be kept in mind that these preliminary observations are based on relatively simple ESTPs with just ten terminals. A similar analysis in the next subsections based on ESTP with 20 and 50 terminals show which of these conclusions can also be supported in case of more difficult ESTPs.

For the sake of completeness, in Table 4 it is summarized for each algorithm, in how many ESTPs it was able to find the best-known solution (obtained by any of the algorithms considered). There are only minor differences among the best performing (in average) algorithms already mentioned. However, it could be surprising that for example, algorithm nn5 was able to find the best-known solutions in 18 ESTPs. Still, its average performance it significantly worse than that of ip1.0.

Problems with 20 terminals

The Friedman's ranks achieved by each of the 16 algorithms for the 20 problems with 20 terminals are presented in Table 5. These ranks are based on the average performance of the algorithms.

Similarly to the results for ESTPs with ten terminals, the best performing algorithm is ip1.0. However, based on the post hoc analysis (Table 6) its averaged performance is not significantly different (with the significance level $\alpha = 0.05$) than that of algorithms ip0.8, cp0.0, ip0.6, ip0.4, and cp0.2. Again, the correction procedure seems to work better with Baldwin effect (small values of *change_prob*) and the improvement procedure works better with high values of *change_prob* (Lamarckian evolution).

In this set of ESTPs with 20 terminals, it is even more evident that the simple evolutionary algorithm enorm is not able to solve the problems. Based on the post hoc analysis (Table 6), it can be observed that enorm is significantly worse than any of the hybrid methods. On the other hand, only algorithms ip1.0, ip0.8, cp0.0, ip0.6, ip0.4 are significantly better than algorithm nn5. However, algorithm nn5 by itself is found to be significantly better than none of the considered ones.

Table 7 shows the number of best solutions found by each of the algorithms. Again, there is a remarkable ability of the

Table 7 Number of best solutions found (problems with 20 terminals)

Algorithm	ip0.6	ip1.0	nn5	ip0.4	ip0.2	ip0.8	nn3	cp0.0	cp0.2	cp0.6	cp0.4	cp0.8	ip0.0	cp1.0	nn-1	enorm
Number of best solutions	17	17	17	16	15	15	12	8	7	7	6	6	6	4	4	1

Table 8 Friedman's ranks for algorithms based on the mean results for problems with 50 terminals

Algorithm	ip1.0	ip0.8	ip0.6	ip0.4	ip0.2	nn5	cp0.8	cp0.6	cp0.2	cp0.4	cp0.0	cp1.0	ip0.0	nn3	nn-1	enorm
Rank	1.05	2.05	3.4	4.25	5.95	7.7	8.55	8.9	9	9	10	10	12.45	12.7	15	16

Table 9 p values adjusted by Shaffer's procedure (problems with 50 terminals)

	ip0.8	ip0.6	ip0.4	ip0.2	nn5	cp0.8	cp0.6	cp0.2	cp0.4	cp0.0	cp1.0	ip0.0	nn3	nn-1	enorm
ip1.0	8.370	3.675	1.409	0.069	**0.001**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
ip0.8	–	8.138	3.886	0.498	**0.012**	**0.001**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
ip0.6		–	8.370	3.161	0.249	**0.042**	**0.018**	**0.014**	**0.014**	**0.001**	**0.001**	**0.000**	**0.000**	**0.000**	**0.000**
ip0.4			–	6.471	1.009	0.249	0.119	0.098	0.098	**0.010**	**0.010**	**0.000**	**0.000**	**0.000**	**0.000**
ip0.2				–	6.372	3.030	1.952	1.754	1.754	0.393	0.393	**0.001**	**0.001**	**0.000**	**0.000**
nn5					–	8.370	8.145	8.145	8.145	3.798	3.798	0.098	0.059	**0.000**	**0.000**
cp0.8						–	8.370	8.370	8.370	8.052	8.052	0.498	0.327	**0.001**	**0.000**
cp0.6							–	8.370	8.370	8.370	8.370	0.882	0.592	**0.004**	**0.000**
cp0.2								–	8.370	8.370	8.370	1.009	0.685	**0.005**	**0.000**
cp0.4									–	8.370	8.370	1.009	0.685	**0.005**	**0.000**
cp0.0										–	8.370	3.421	2.771	0.059	**0.005**
cp1.0											–	3.421	2.771	0.059	**0.005**
ip0.0												–	8.370	3.161	0.882
nn3													–	3.798	1.221
nn-1														–	8.370

p values which indicate significant differences (less than 0.05) are in bold

nn5 algorithm to find the best solutions at least once compared to its poor averaged performance. Taking into account the average performance and the number of best solutions found, it can be concluded from this set of experiments that the evolutionary algorithm with improvement procedure and Lamarckian evolution should be preferred.

Problems with 50 terminals

The Friedman's ranks achieved by each of the 16 algorithms for the 20 problems with 50 terminals are presented in Table 8. These ranks are based on the average performance of the algorithms.

In the case of ESTPs with 50 terminals, it is evident that the evolutionary algorithms with improvement procedure and Lamarckian evolution performs best on average. It can be observed (Table 8) that ip1.0, ip0.8, ip0.6, ip0.4 and ip0.2 have the best ranks from among all algorithms. Algorithm ip1.0 is assigned the highest rank, however, a significant difference (Table 9) between ip1.0 and algorithms ip0.8, ip0.6, ip0.4 and ip0.2 cannot be discovered. Accepting the improved solution is highly beneficial as even the small values of *change_prob* (such as in ip0.2) make the evolutionary algorithm with improvement procedure significantly better than ip0.0. On the other hand, the usefulness of using the correction procedure as the local search method in the evolutionary algorithm is no longer proved when compared to the best performing ip1.0 and ip0.8 algorithms: according to Shaffer's post hoc procedure, any of the algorithms cp0.0–cp1.0 are performing significantly worse than ip1.0 and ip0.8. On the other hand, the correction procedure in MA with any *change_prob* value improves the performance of simple evolutionary algorithm enorm significantly. Taking into account that the correction procedure is much less computationally expensive than the improvement procedure, it might still be of use in difficult problems with many terminals.

Similarly to the previous set of experiments the relatively high rank of nn5 heuristic should be noticed. Again, based on the results, there is not enough evidence that its averaged behavior is better than any of the evolutionary algorithms with improvement or even correction procedures. The high rank of the nn5 algorithm might be explained based on the results presented in Table 10, where the numbers of best

Table 10 Number of best solutions found (problems with 50 terminals)

Algorithm	nn5	ip1.0	ip0.8	ip0.4	nn3	cp0.0	cp0.2	cp0.4	cp0.6	cp0.8	cp1.0	enorm	ip0.0	ip0.2	ip0.6	nn-1
Number of best solutions	13	4	2	1	1	0	0	0	0	0	0	0	0	0	0	0

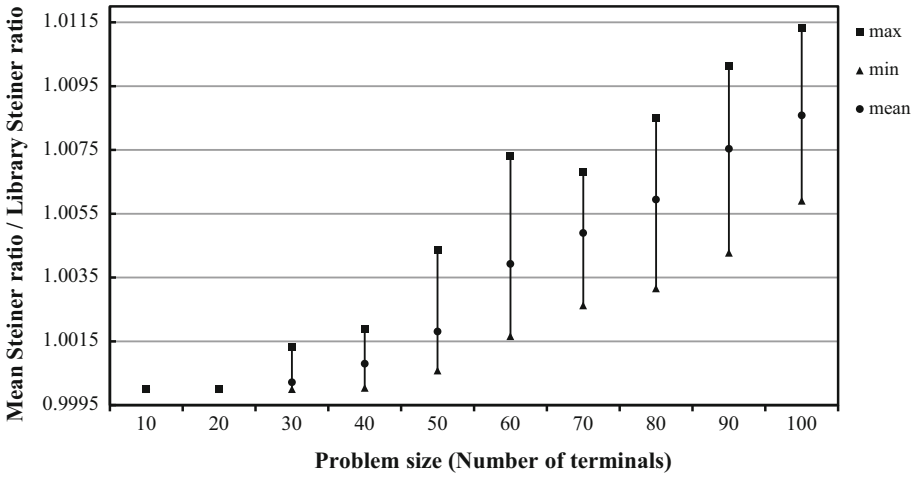

Fig. 8 Summary of average results of the MA with improvement procedure for problems from OR-library: minimum, maximum and mean values for each group of problems are presented

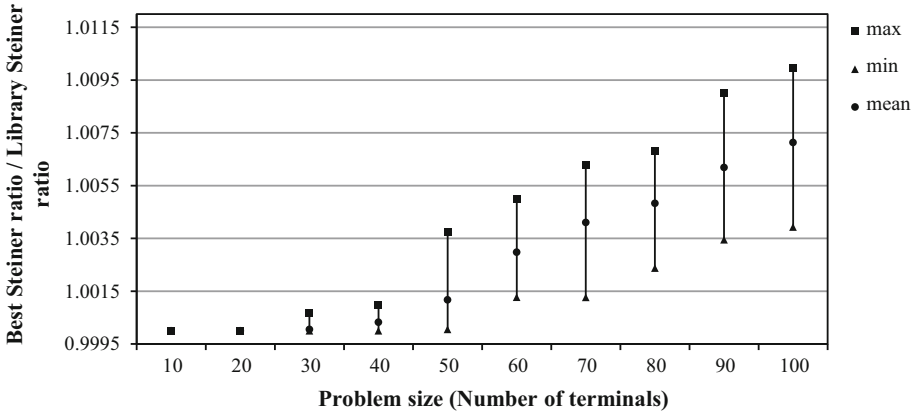

Fig. 9 Summary of best solutions found by the MA with improvement procedure for problems from OR-library: minimum, maximum and mean values for each group of problems are presented

solutions found by each of the algorithms are presented. It is interesting to note that the clear winner here is nn5 heuristic. It can be deduced that the poor average performance of nn5 is somehow compensated by its ability to find the best solutions in only limited number of the considered ten runs per problem.

4.4 Results for problems from OR-Library

In order to assess the ability of the proposed approach to provide optimal or near optimal solutions, ESTPs from OR–Library have been used [25]. In this section we compare the results of the proposed algorithm with the best performing settings (ip1.0) with the optimal solutions for the problems provided in the OR-Library files estein10, estein20, estein30, estein40, estein50, estein60, estein70, estein80, estein90 and estein100. Those files have been obtained from http://people.brunel.ac.uk/~mastjjb/jeb/orlib/esteininfo.html. Each file contains 15 EST problems with the number of terminals corresponding to the number in the name of the file. For example, in the file estein80, there are 15 problems, each with 80 terminals (points on 2D plane).

The algorithm ip1.0 with the same settings as in the previous subsection has been run ten times for each problem. The mean and the best Steiner ratios have been calculated for each problem over the ten results. Next, for each group of problems of a given size, the average, minimum and maximum Steiner ratio have been found in the 15 problems. Next, these values have been compared with the Steiner ratios of the known optimal solutions taken from the library file. The results are summarized in Figs. 8 and 9. The presented values are the quotient of the mean, minimum and maximum values of the quotient for each group of problems. The value of the quotient equal 1 indicates that the optimal solution has been found. The bigger the values, the worse the solution (bigger Steiner ratio).

From the results presented in Figs. 8 and 9, it can be seen that the proposed algorithm can provide stable performance and find near-optimal solutions. From Fig. 9 it can be seen that the Steiner ratios of the best solutions are never worse more than 1% than the Steiner ratios of the optimal solutions from OR-Library. The mean Steiner ratios are worse slightly more than 1% than the optimal ones only for a few problems with 90 and 100 terminals (Fig. 8).

The results presented in this section indicate that the proposed algorithm scales well with the size of the problems. Perfect solutions have been found for all problems with 10 and 20 terminals and near optimal solutions for the rest. No significant fall in the quality of the results can be observed when the number of terminals grows.

5 Conclusions

The main purpose of this work was to investigate the importance of local search procedures in a MA for Euclidean Steiner Tree Problem (ESTP) on a plane. ESTP is a difficult problem, and there is still a need for new heuristics as the exact methods are computationally demanding. Evolutionary algorithms are known to be efficient in locating the promising areas of the search space, on the other hand, they are slow in fine tuning the solutions. For that reason, local search methods can be used as a part of the evolutionary framework to speed up the convergence. However, this may lead to the quick loss of diversity in population and premature convergence. Two well-known approaches have been tested, Baldwin effect and Lamarckian evolution together with a set of intermediate solutions in which the corrected/improved solution is accepted with a given probability. Two local search methods have been tested, the correction procedure, which only repairs the set of candidate Steiner points making them satisfy the degree and angle condition and the improvement procedure which actively adds new candidate Steiner points. As the second procedure (the improvement procedure) modifies the candidate solution significantly more than the correction procedure, it might be expected to be more dangerous to use as it can limit the abilities of an evolutionary algorithm to explore the search space. However, as the numerical experiments showed, when the ESTPs become more difficult (problems with 10, 20 and 50 terminals have been tested), the MA based on a GA combined with the improvement procedure and Lamarckian evolution becomes the best choice when the averaged performance is considered. This was supported by statistical analysis of the results using Friedman test followed by Shaffer's post hoc procedure.

The best performing variant of the proposed MA has been tested on problems from OR-Library with known optimal solutions. There are 15 problems in each set of problems with 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 terminals on a 2D plane. The Steiner ratios of the best results of the proposed MA are never worse more than 1% than the Steiner ratios of the optimal solutions taken from the library. It can be observed that the algorithm scales well and there is no significant drop in the performance when the size of the problems (the number of terminals) grows.

This work to the best of Author's knowledge is the first attempt to test the importance of local search methods in a MA when applied to ESTP. It by no means exhaust the topic. Considering the enormous number of evolutionary heuristics that exists nowadays and possibly other local search procedures, further work can explore the other combinations of evolutionary/local heuristics. The results presented in this work suggest that to solve ESTP with the evolutionary algorithm the role of the local search method is of crucial importance and it should be allowed to modify the candidate solutions significantly. The future work can also include the application of the described method to problems with an even higher number of terminals. The inherent parallel nature of GA might be very helpful in solving such difficult problems.

References

1. Brazil M, Zachariasen M (2015) Optimal interconnection trees in the plane. Theory, algorithms and applications. Springer, Berlin. https://doi.org/10.1007/978-3-319-13915-9
2. Melzak ZA (1961) On the problem of Steiner. Can Math Bull 4:143–148
3. Winter P, Zachariasen M (1997) Euclidean Steiner minimum trees: an improved exact algorithm. Networks 30:149–166. https://doi.org/10.1002/(SICI)1097-0037(199710)30:3<149::AID-NET1>3.0.CO;2-L
4. Trietsch D, Hwang F (1990) An improved algorithm for steiner trees. SIAM J Appl Math 50:244–263. https://doi.org/10.1137/0150015
5. Smith WD (1992) How to find Steiner minimal trees in euclideand-space. Algorithmica 7:137–177. https://doi.org/10.1007/BF01758756
6. Barreiros J (2003) A hierarchic genetic algorithm for computing (near) optimal euclidean steiner trees. In: Barry AM (ed) GECCO 2003: Proceedings of the bird of a feather workshops, genetic and evolutionary computation conference. AAAI, Chigaco, pp 56–65
7. Jesus M, Jesus S, Márquez A (2004) Steiner trees optimization using genetic algorithms. Technical report
8. Hu Y, Jing T, Feng Z, Hong X-L, Hu X-D, Yan G-Y (2006) ACO-Steiner: ant colony optimization based rectilinear steiner minimal tree algorithm. J Comput Sci Technol 21:147–152. https://doi.org/10.1007/s11390-006-0147-0

9. Liu L, Song Y, Zhang H, Ma H, Vasilakos AV (2015) Physarum optimization: a biology-inspired algorithm for the steiner tree problem in networks. IEEE Trans Comput 64:818–831. https://doi.org/10.1109/TC.2013.229
10. Chen X, Ong YS, Lim MH, Tan KC (2011) A multi-facet survey on memetic computation. IEEE Trans Evol Comput 15:591–607. https://doi.org/10.1109/TEVC.2011.2132725
11. Meuth R, Lim MH, Ong YS, Wunsch DC (2009) A proposition on memes and meta-memes in computing for higher-order learning. MemetComput 1:85–100. https://doi.org/10.1007/s12293-009-0011-1
12. Botzheim J, Toda Y, Kubota N (2012) Bacterial memetic algorithm for offline path planning of mobile robots. Memet Comput 4:73–86. https://doi.org/10.1007/s12293-012-0076-0
13. Goldbarg MC, Asconavieta PH, Goldbarg EFG (2012) Memetic algorithm for the traveling car renter problem: an experimental investigation. Memet Comput 4:89–108. https://doi.org/10.1007/s12293-011-0070-y
14. Mirsaleh MR, Meybodi MR (2016) A new memetic algorithm based on cellular learning automata for solving the vertex coloring problem. Memet Comput 8:2112–222. https://doi.org/10.1007/s12293-016-0183-4
15. Hasan SMK, Sarker R, Essam D, Cornforth D (2009) Memetic algorithms for solving job-shop scheduling problems. Memet Comput 1:69–83. https://doi.org/10.1007/s12293-008-0004-5
16. Karim MR, Ryan C (2012) Attributed grammatical evolution with lookahead for the multiple knapsack problem. Memet Comput 4:279–302. https://doi.org/10.1007/s12293-012-0097-8
17. Whitley D, Gordon VS, Mathias K (1994) Lamarckian evolution, the Baldwin effect and function optimization. In: Davidor Y, Schwefel H-P, Männer R (eds) Parallel problem solving from nature—PPSN III SE-1. Springer, Berlin, pp 5–15
18. Ong YS, Keane AJ (2004) Meta–Lamarckian learning in memetic algorithms. IEEE Trans Evol Comput 8:99–110. https://doi.org/10.1109/TEVC.2003.819944
19. Elfwing S, Uchibe E, Doya K, Christensen HI (2007) Evolutionary development of hierarchical learning structures. IEEE Trans Evol Comput 11:249–264. https://doi.org/10.1109/TEVC.2006.890270
20. Paenke I, Sendhoff B, Rowe J, Fernando C (2007) On the adaptive disadvantage of lamarckianism in rapidly changing environments. In: Proceedings of the 9th European conference on advances in artificial life, Springer, Berlin, pp 355–364
21. Ishibuchi H, Kaige S, Narukawa K (2005) Comparison between lamarckian and baldwinian repair on multiobjective 0/1 knapsack problems. In: Coello Coello CA, Hernández Aguirre A, Zitzler E (eds) Evolutionary multi-criterion optimization: third international conference, EMO 2005, Guanajuato, Mexico, March 9–11, 2005. Proceedings. Springer, Berlin, pp 370–385
22. Choi S-S, Moon B-R (2005) A graph-based Lamarckian–Baldwinian hybrid for the sorting network problem. IEEE Trans Evol Comput 9:105–114. https://doi.org/10.1109/TEVC.2004.841682
23. Derrac J, García S, Molina D, Herrera F (2011) A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut Comput 1:3–18. https://doi.org/10.1016/j.swevo.2011.02.002
24. Alcalá-Fdez J, Fernández A, Luengo J, Derrac J, García S, Sánchez L, Herrera F (2011) KEEL data-mining software tool: data set repository, integration of algorithms and experimental analysis framework. J Mult-Valued Log Soft Comput 17:255–287. https://doi.org/10.1007/s00500-008-0323-y
25. Beasley JE (1990) OR-library: distributing test problems by electronic mail. J Oper Res Soc 41:1069–1072