

HYBRID DIFFERENTIAL EVOLUTION FOR THE EUCLIDEAN STEINER TREE PROBLEM

Andrey V. R. Oliveira

Federal University of Technology - Paraná
Av. Alberto Carazzai, 1640
andreyvro@hotmail.com

Danilo S. Sanches

Federal University of Technology - Paraná
Av. Alberto Carazzai, 1640
danilosanches@utfpr.edu.br

ABSTRACT

This paper presents a hybrid differential evolution algorithm for the Euclidean Steiner tree problem. This is an optimization problem whose goal is to obtain a minimum length tree to interconnect a set of fixed points, and for this to be achieved, new auxiliary points (called Steiner points) can be added. The proposed heuristic uses the Delaunay triangulation to insert Steiner points into good candidate positions, and then these points have their positions optimized by a differential evolution algorithm. From the set of points, a greedy algorithm searches for minimum spanning tree with degree restriction, if the topology presents errors, another algorithm corrects it; at the end, the Steiner points are placed in their optimum positions according to the topology, thus generating a Steiner tree. Results from numerical experiments for benchmark library problem (OR-Library) are presented.

KEYWORDS. Euclidean Steiner tree problem. Differential evolution. Hybrid algorithms.

Paper topics (MH - Metaheuristics, TAG - Theory and Algorithms in Graphs)

1. Introduction

The objective of the Euclidean Steiner tree problem (ESTP) is to find a tree of minimum length that interconnects a set of fixed points arranged in the Euclidean plane, so that to reduce the length total of the tree, it is allowed the insertion of extra points, denominated “Steiner points”. As the number of fixed points grows, the size of possible solutions to be verified increases exponentially [Gilbert and Pollak, 1968]. The discrete ESTP was presented as belonging to the class of *NP*-Hard problems by Garey et al. [Garey et al., 1977].

Due to changes in rules, constraints, metric spaces, and dimensions, Steiner’s minimal tree problem has several variants. The design of electronic circuits VLSI (Very Large Scale Integration) requires the use of taxicab geometry, rather than Euclidean geometry, and this gives rise to the *rectilinear Steiner tree problem* (RSTP), in that the Steiner points are positioned in a “Hanan grid”, which consists of points of intersection between vertical and horizontal lines connecting the fixed points [Hanan, 1966]. Already established engineering projects that require structural optimization give rise to the *network Steiner tree problem* (NSTP), also known as *Steiner tree problem in graph* (STPG) [Koch and Martin, 1998]. Often, to optimize a wireless network, it is necessary to minimize the length of the largest edges of a Steiner tree, which gives rise to *bottleneck Steiner tree problem* (BSTP), also known as *min-max Steiner tree problem* [Wang and Du, 2002].

This is a combinatorial optimization problem present in a large number of applications originating from the study of network projects [Du and Hu, 2008]. Some of the problems arise from the modeling of biological structures, construction and maintenance of systems, such as energy distribution, transport, sewage, communication, VLSI chip layout design, robotics, etc. Due to its importance for solving various real world problems, the Euclidean Steiner problem is studied, and several forms of resolution are proposed, however, most algorithms tend to use a high computational resource as the number of fixed points increase.

In order to obtain a good approximate solution of feasible computational time for ESTP in $\mathbb{R}^2$ (two-dimensional space), this work presents a heuristic based on a differential evolution algorithm (DE). Differential evolution (DE) is a stochastic search method proposed by Rainer Storn and Kenneth Price [Storn and Price, 1997], initially to solve the problem of polynomial adjustment of Chebychev, but can be used to solve problems of various natures. It is an evolutionary algorithm (EA), because it is inspired by the mechanics of natural evolution, and it looks a lot like the genetic algorithm (GA) [Goldberg, 1989], since it also manipulates populations and has selection, recombination and mutation operators. This work proposes a hybrid algorithm based on an ED, which allows the generation of good quality solutions in computational time feasible for the ESTP. The advantage of using a hybrid DE, rather than a pure DE, is to improve the convergence speed of the algorithm and to obtain solutions with higher quality.

2. The Steiner Tree Problem

2.1. History

The problem of the Euclidean Steiner tree has its origin with the Fermat-Torricelli problem in 1643, when the mathematician Pierre de Fermat (1601-1655) proposed a challenge which consisted in finding a point in the plane in which the sum of the distances to three points be as small as possible. Evangelista Torricelli (1608-1647) has proposed a solution that consists of drawing equilateral triangles outside the initial triangle formed by the three given points and draw circles around each of the three new triangles. The solution of the problem is the point of intersection of the circumferences, which became known as the *Torricelli point*, and can be seen in Figure 1(a). Subsequently, Thomas Simpson (1710-1761) proposes a solution which consists in drawing equilateral triangles outside the initial triangle formed by the three given points, and then drawing straight lines

from the outer vertices of the equilateral triangles until the opposite vertices of the initial triangle, which became known as *Simpson lines*, and can be seen in Figure 1(b) [Hwang et al., 1992].

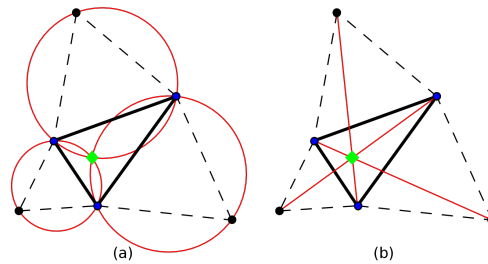


Figure 1: Torricelli point and Simpson lines

Bonaventura Cavalieri (1598-1647) showed in 1647 that the lines connecting the three initial points to the Torricelli point have 120° with each other. When one of the angles in the given triangle is less than 120° , the Torricelli point lies outside the given triangle and is no longer the minimization point, in which case the minimization point is the vertex of the obtuse angle [Brazil et al., 2013].

The first publication dealing with the Euclidean Steiner problem is attributed to the french mathematician Joseph Diaz Gergonne (1771-1859), who in 1811 published a paper in his journal, in which he questioned how to obtain a minimum-length network by interconnecting a set of vertices known in the plan, and discussed resolutions to problems with four to six points. The way Gergonne proposed the problem is different from Fermat-Torricelli's problem for multiple points, since you can add more extra points to the network [Brazil et al., 2013].

2.2. Definitions

The Steiner minimal tree problem (or SMTP) can be mathematically described as follows:

Find a geometric network $T(V, E)$ (a connected graph with a set of vertices V and a set of edges E) so that $N \subseteq V$ and $S = V \setminus N$ is a (possibly empty) set of points known as Steiner points, so that $\sum_{e \in E} \text{length}(e)$ (sum of the lengths of edges in E), is minimized [Brazil et al., 2013].

Therefore, the SMTP aims to find a network to interconnect a set of fixed points in a metric space, and can add new points in order to minimize the total distance of the network. The fixed points are called terminals and the new points added are called *Steiner points* [Hwang et al., 1992]. In this work, the metric space is the Euclidean, so the problem is called ESTP. In Figure 2 two examples of Euclidean Steiner minimal tree are presented.

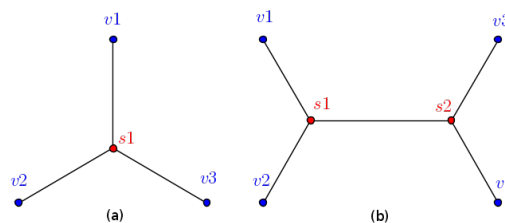


Figure 2: Minimal Steiner Tree

Figure 2(a) shows an ESTP with three terminal vertices v_1, v_2, v_3 , and a Steiner point s_1 located at the *Torricelli point* of the triangle formed by the terminal vertices. Figure 2(b) shows an

ESTP with four terminal vertices v_1, v_2, v_3, v_4 and two Steiner points s_1, s_2 , connected in order to minimize the total length of the network.

It is assumed that a solution for SMTP is a tree T (otherwise, it will not minimize the length), and that any Steiner point in T has at least three incident edges [Gilbert and Pollak, 1968].

Given v points in a metric space, in order to obtain a solution for the Steiner minimal tree (SMT), it is necessary to determine: the amount of Steiner points that will make up the solution; the coordinates of the Steiner points in metric space; and the combinatorial structure of the solution, that is, a representation of the connections between the vertices (terminals and Steiner points) of the tree found. Minimal Steiner trees are difficult to find because a tree with minimal local length is not necessarily *globally minimal*, not the absolute minimum of all possible solutions [Gilbert and Pollak, 1968].

2.3. Properties

Steiner's minimal trees have several properties, and knowing some of them can make it easier to search for better solutions. Understanding the properties of the Steiner minimal tree is necessary to seek better solutions, so the main properties are: **Angle:** Edges that are at a certain vertex have at least 120° of angle, which results in the fact that every Steiner point has three incident edges forming angles of 120° [Gilbert and Pollak, 1968]; **Degree:** A minimum Steiner tree with n fixed points has at most $n - 2$ Steiner points. A Steiner tree that has the maximum number of Steiner points is called a *full Steiner tree* (FST) [Gilbert and Pollak, 1968]; **Full components:** Through the breaking of some edges, a Steiner topology G can be divided into full Steiner subtopologies, and these are called *full components* of G [Hwang et al., 1992].

2.4. Length

As the problem is approached in $\mathbb{R}^2$, for each vertex is assigned a value x and y , so as to represent its position in the Cartesian plane, therefore, distance between two points $P = (p_x, p_y)$ and $Q = (q_x, q_y)$ is given by $D(P, Q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}$ [Deza and Deza, 2009].

Let V be the set of vertices of a tree, and A an adjacency matrix that determines the topology of the tree, such that, if $V_i \in V_j$ are adjacent, then $A_{i,j} = 1$, else $A_{i,j} = 0$, the total Euclidean length of the tree $L(V)$ is given by $L = \sum_{i < j} A_{i,j} D(V_i, V_j)$ [Gilbert and Pollak, 1968].

3. Related Work

Several studies in the literature of graph theory show approximation algorithms, exact and heuristic intended to solve the ESTP.

Smith [Smith, 1992] developed the first exact algorithm to solve the ESTP in $\mathbb{R}$ with dimension greater than 2. The algorithm uses a “branch-and-bound” method to construct FSTs and stores them in a structure called “vector-topology”. For an SMT to be obtained from a vector-topology, it is necessary that the Steiner points are in optimal positions, and for this purpose a set of linear equations is used iteratively to reposition the Steiner points. These approach is inspired by a physical model that simulates the relaxation of a system of ideal springs. [Fampa and Anstreicher, 2008] proposed enhancements to the Smith algorithm, creating a new algorithm that was named “Smith+”, and presenting excellent results for ESTP in $\mathbb{R}^d$ para $2 \leq d \leq 5$, where d is the number of dimensions.

Winter [Winter, 1985] developed the exact algorithm “Geosteiner”, which was perfected by Winter and Zachariasen [Winter and Zachariasen, 1997]. The algorithm enumerates some FSTs for a set of fixed points, and then concatenates the subtrees in order to obtain an SMT. In the enumeration process, a geometric approach is used that avoids the creation of FSTs that do not meet the structural requirements necessary to form an SMT. Warne [Warne, 1998] has shown that

Geosteiner can solve problems with more than 1000 points optimally. Since its launch as open source software in the year 2000, several enhancements have been implemented, and are presented in Juhl et al. [Juhl et al., 2018].

Some heuristics use MST as the basis for generating SMT. Thompson [Thompson, 1973] generates the MST, searches for sets of three vertices connected with angles smaller than 120° at the intersection of the edges, and then inserts a Steiner point into the Torricelli point of the triangle formed, disconnects the vertices, and reconnects them to the Steiner point. The process of inserting Steiner points is repeated iteratively until forming an FST, which then has its Steiner points repositioned in optimal positions in order to obtain the SMT. Dreyer and Overton [Dreyer and Overton, 1998] developed a heuristic based on Thompson's heuristics, which repositioned the Steiner points after insertion of all Steiner points.

Some studies show the application of geometric structures as a way to start the Steiner tree. Smith et al. [Smith et al., 1981] propose a heuristic that uses the *Delaunay triangulation* to partition the set of fixed points into triangles that are used to insert Steiner points, then uses the Voronoi diagram and the MST of the set to find possible SMTs. Beasley and Goffinet [Beasley and Goffinet, 1994] use a "simulated annealing" algorithm that creates solutions by applying Delaunay triangulation to generate candidate Steiner points, and then uses the MST to remove redundant Steiner points. Laarhoven and Ohlmann [Laarhoven and Ohlmann, 2010] employ Delaunay triangulation to insert Steiner points, and second order conic programming to optimize position, and thus solve problems in $2 \leq d \leq 5$, where d is the number of dimensions.

Lundy [Lundy, 1985] proposed a "simulated annealing" approach to create FSTs by changing edges between vertices. Forte et al. [Forte et al., 2015] developed a "Iterated local search" algorithm for problems in n dimensions, and used the Smith vector-topology as a way to code the solutions. Bereta [Bereta, 2017] uses "Monte Carlo Tree Search" and a heuristic based on the Torricelli point calculation to improve the positioning of Steiner points.

Barreiros [Barreiros, 2003] uses a two-phase GA; in the first phase it decides how to partition a set of vertices and in the second phase it finds the SMT of each partition. The GA of Jesus et al. [Jesus et al., 2004] codes the solution through a vector representing the position and points adjacent to the Steiner points. Yang [Yang, 2006] presents a GA whose approach is based on moving the Steiner points, constructing a tree joining nearby points, and at the end eliminating Steiner points that do not have degree three and invalidate the angle property (shown in sub-session 2.3). Bereta [Bereta, 2018] proposes a memetic algorithm, which encodes the solution through a vector that stores the coordinates of the Steiner points, and a bit that informs whether the Steiner point is active or not.

The state-of-art for ESTP in R^d , with $d = 2$, is the exact algorithm "Geosteiner" [Winter and Zachariasen, 1997], and for $d > 2$, it is the exact algorithm "Smith+" [Fampa and Anstreicher, 2008]. As for the non-deterministic algorithms, Laarhoven and Ohlmann [Laarhoven and Ohlmann, 2010] present good results in viable computational time, since Yang [Yang, 2006] shows a better quality in the solutions, but with high computational time when compared to the other heuristics.

4. A Differential Evolution for Euclidean Steiner Tree Problem

4.1. Differential Evolution

Differential Evolution (DE) is an evolutionary algorithm (EA) most commonly used in optimization of continuous problems modeled by real numbers. Although it has a simple operation and is easy to implement and control, the DE presents a fast convergence when compared to other heuristics [Storn and Price, 1997]. Its operation resembles that of a genetic algorithm [Goldberg,

1989], as it also uses recombination and mutation operators to create new individuals (solutions) in a population.

The initial population of a DE is composed of n individuals, and each individual is represented by a vector of parameters X , whose size d does not change over the g generations. If we do not have any knowledge about the problem to be solved, the parameter vectors can be defined through a uniform probability distribution.

After the population is initialized, each vector has its calculated fitness. Then each *target vector* is manipulated by mutation and crossover generating a *trial vector* which replaces it if it is more fit. To control the search, we initially define values for the F parameters, which controls the mutation, Cr that controls the crossover, and n representing the number of vectors in the population.

For each *target vector* X_i^g , where i is the vector index in the population, and g is the current generation, three different individuals (X_a^g, X_b^g, X_c^g) are randomly chosen to generate a *trial vector* (X_i^{g+1}) through Eq. 1.

$$X_i^{g+1} = X_a^g + F(X_b^g - X_c^g) \quad (1)$$

In Eq. 1 a, b, c are integers in the range $[0, n - 1]$, different from each other, and different from i , so an DE must have a population with at least four individuals. The parameter F is a real and constant factor in the range $[0.2]$, which controls the amplification of the differential variation $((X_b^g - X_c^g))$ [Storn and Price, 1997].

After the mutation, some components j of each *trial vector* (X_i^{g+1}) are replaced by components of the *target vector* (X_i^g) through the crossover operator, defined by Eq. 2. In that operator, a real number in the interval $[0, 1]$ denoted by Cr represents the crossover rate.

$$X_{i,j}^{g+1} = \begin{cases} X_{i,j}^{g+1}, & \text{if } rand() \leq Cr \\ X_{i,j}^g, & \text{Otherwise} \end{cases} \quad (2)$$

In Eq. 2, $rand()$ is a random real number in the interval $[0, 1]$, i is the index of the individual in the population, j is the index of the component in the i -th vector, and g represents the generation. In this operator, the higher the Cr value, the farther a *trial vector* is from a *target vector*.

At the end, each *trial vector* has its calculated fitness, and if it is better than its respective *target vector*, it replaces it in the population, so that a natural evolution in the quality of the solutions occurs. The process can be terminated when the algorithm reaches convergence, or run for a certain number of generations or period of time.

4.2. Methodology

To solve the ESTP, it is necessary to obtain the topology, the quantity and the position of the Steiner points; and for these purposes there are several approaches. In this work, we chose to work with full Steiner trees (FST), that is, trees that have $n - 2$ Steiner points, where n is the number of fixed points, since in an DE the number of components of a vector is fixed throughout the search process. At the end of the process, an algorithm is used to remove unfeasible Steiner points, ie Steiner points that are above or very close to other Steiner points. When working with FSTs, the search for the number of points during the execution of the algorithm is avoided, but the memory and processing consumption is higher, since the trees will have the maximum number of Steiner points in all generations.

To work the ESTP in $\mathbb{R}^2$ as a problem of a continuous nature, in the DE each component of a vector represents the coordinate (x and y) of a Steiner point in the Euclidean plane. Establishing a set of fixed points and Steiner points, algorithms are used sequentially to create an FST, store its

structure in an adjacency matrix, and calculate its length, which is considered the fitness of the solution.

To initialize the population, the Delaunay triangulation is used on fixed points. Delaunay triangulation is a technique employed to rapidly generate non-overlapping triangular meshes that reflect the closest neighborhood of a set of points in $\mathbb{R}^2$, and for this purpose a simple *sweep - line algorithm* based on the works of Biniarz et al. [Biniarz and Dastghaibfard, 2012] and Sinclair [Sinclair, 2010]. After constructing the Delaunay triangulation for n fixed points, $n - 2$ triangles are chosen randomly, and for each triangle, a Steiner point is inserted into its Torricelli point. The studies of Smith et al. [Smith et al., 1981], Beasley and Goffinet [Beasley and Goffinet, 1994] and Laarhoven and Ohlmann [Laarhoven and Ohlmann, 2010] show that this type of triangulation has good structural properties, since the MST of a tree is contained in the vertices of its triangulation, and each triangle tends to be equilateral.

From a set of fixed points and Steiner points arranged in the two-dimensional plane, the algorithm of Kruskal [Kruskal, 1956] is used to construct a tree. Since a Steiner point must have degree three to satisfy its angle property, a constraint is added to the algorithm so that new edges are not added to a Steiner point that already has three incident edges. When performing such a process, some Steiner points may end up being less than three incident edges, so they should be removed as they do not contribute to minimizing distance. To remove a Steiner point of degree one, simply remove the point and its incident edge, to remove a Steiner point v_1 connected to two points v_2 and v_3 , v_1 and its incident edges are removed, and adds a new edge connecting v_2 to v_3 .

After removing unfeasible Steiner points, it is necessary to add new Steiner points to correct the FST topology, and for this the Thompson method [Thompson, 1973] is used. In an FST topology, each fixed point has degree one, so it is necessary to correct all fixed points that have degree greater than one. If v_1 is a fixed point connected to two points v_2 and v_3 , it removes the edges that connects them, inserts a Steiner point in the Torricelli point of the triangle formed by v_1, v_2, v_3 , and reconnects the three points to Steiner point. This process is called *local steinerization* [Hwang et al., 1992], and is performed until all fixed points in the tree have degree one, resulting in a valid FST topology.

In order to have an FST it is necessary that the Steiner points are in optimal positions, and for this purpose the algorithm of Smith [Smith, 1992] is used; which consist of an iterative method, similar to the Gauss-Seidel method, where at each step the coordinates of the Steiner points tend to the optimal position. The algorithm treats the edges as a set of ideal springs, and at each iteration solves a system of linear equations, in order to minimize the potential energy of the spring system. Smith shows that for any initial position defined for the Steiner points, this method converges to the only optimal coordinates of the Steiner points. The studies of Fampa and Anstreicher [Fampa and Anstreicher, 2008] and Forte et al. [Forte et al., 2015] also use this method to reposition Steiner points.

Figure 3 presents the process of constructing a Steiner tree from the Delaunay triangulation to the sixth instance of the ten-point problem of the OR-Library [Beasley, 1990].

In Figure 3(a) is shown 13 triangles on fixed points, generated through Delaunay triangulation. As the instance in question presents $n = 10$ fixed points, to create an FST it is necessary to insert $n - 2$ Steiner points, so in Figure 3(b), are chosen randomly 8 triangles so that Steiner points are added to your Torricelli points. In Figure 3(c) the algorithm of Kruskal [Kruskal, 1956] is applied to generate a MST with degree restriction, so that the Steiner points have a maximum degree three. Figure 3(d) presents an FST after the topology correction process, which removes Steiner points with degree less than three, and reinserts them in suitable places by the method of Thomp-

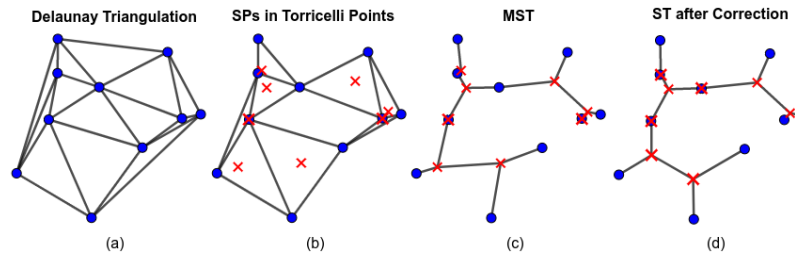


Figure 3: Construction of a Steiner tree

son [Thompson, 1973]; and the Steiner points position correction process, through the algorithm of Smith [Smith, 1992].

After initialization of the population, the ED operators modify the position of the Steiner points, and a new topology is created by constructing the MST (Figure 3(c)), and corrected through the removing, reinserting and repositioning of Steiner points (Figure 3(d)). Once the DE completes the iterations, an FST is returned as the best solution, and if there are Steiner points placed on other points, they can be removed, because they do not contribute to the distance minimization. To eliminate a Steiner point s_1 that is positioned above another point called v_1 , simply connect the s_1 edges in v_1 , and then delete s_1 .

5. Computational Experiments

To evaluate GA's ability to provide good quality solutions, ESTPs from the OR-Library [Beasley, 1990] benchmark library are used. Each problem has 15 Instances, and the optimal solutions of the problems with 10, 20, 30, 40, 50, 60, 70, 80, 90 and 100 points in $\mathbb{R}^2$ are used as reference.

The proposed algorithm was implemented in C++ language, compiled with GCC 8.2.1, and multicore support was added through the OpenMP library; and experiments were run on a computer with an Intel Core i5-5200U 2.20GHz processor, 4Gb of memory with the Manjaro Linux operating system 18.0.2 for 64 bit.

The quality of the solutions is calculated by the percentage reduction of the solution over the MST, so, since T is a solution obtained for a set of points X , the quality of the solution is given by $Q = ((l(MST) - l(T)) * 100) / l(MST)$. As the algorithm is stochastic in nature, in each experiment, each of the instances are tested thirty times, and at the end, statistics are generated that show the best solution and the mean solution.

5.1. Differential Evolution Mutation Strategies

There are several ways to implement the mutation operator for DE [Price et al., 2005]. The mutation strategy presented in subsection 4.1 is called random mutation, and is used in classic DE. To represent several variants of mutation strategies, the notation $DE/x/y/z$ is commonly used in the literature; so that x determines the vector to be mutated, which can be “*rand*” representing a randomly chosen vector in population, or “*best*” representing the fittest vector in population; y is the number of difference vectors used in the mutation; and z represents the crossover operator, which can be “*bin*” which means binomial or “*exp*” which means exponential.

In order to investigate the effects of different mutation strategies, the first experiments test the performance of six mutation schemes with binomial crossover operator (shown in the subsection 4.1). The analyzed strategies are shown in table 1.

In table 1, X_i^{g+1} represents the *trialvector*, where i is the index of the vector and g represents the generation. If Np is the number of individuals, the integers $a, b, c, d, e \in [1, Np]$ are

Mutation	Notation
$X_i^{g+1} = X_a^g + F(X_b^g - X_c^g)$	$DE/rand/1$
$X_i^{g+1} = X_{best}^g + F(X_b^g - X_c^g)$	$DE/best/1$
$X_i^{g+1} = X_a^g + F_1(X_b^g - X_c^g) + F_2(X_d^g - X_e^g)$	$DE/rand/2$
$X_i^{g+1} = X_{best}^g + F_1(X_b^g - X_c^g) + F_2(X_d^g - X_e^g)$	$DE/best/2$
$X_i^{g+1} = X_a^g + F_1(X_b^g - X_a^g) + F_2(X_c^g - X_d^g)$	$DE/ctor/1$
$X_i^{g+1} = X_a^g + F_1(X_{best}^g - X_a^g) + F_2(X_b^g - X_c^g)$	$DE/ctob/1$

Table 1: Analyzed DE Mutation Strategies

random and the integer *best* is the index of the fittest individual in the population. The parameters $F, F_1, F_2 \in [0, 2]$ are the scaling factors of the operators. The notation *ctor* refers to the “current-to-rand” mutation strategy, and the *ctob* notation refers to “current-to-best”, where the current vector is moved toward another vector (*rand* or *best*) before being disturbed by a scaled difference.

In these experiments the mutation schemes are tested with the fifteenth instance of 40 points and the first instance of 50 points of the OR-Library. Regarding the initial parameterization; it is defined that the population size is $Np = 25$; the F, F_1, F_2 parameters used in the mutations, and Cr which represents the crossover rate are defined as 0.5; in both experiments the DE executes for 30 generations. Figure 4 shows the mean convergence curve of thirty tests for each instance.

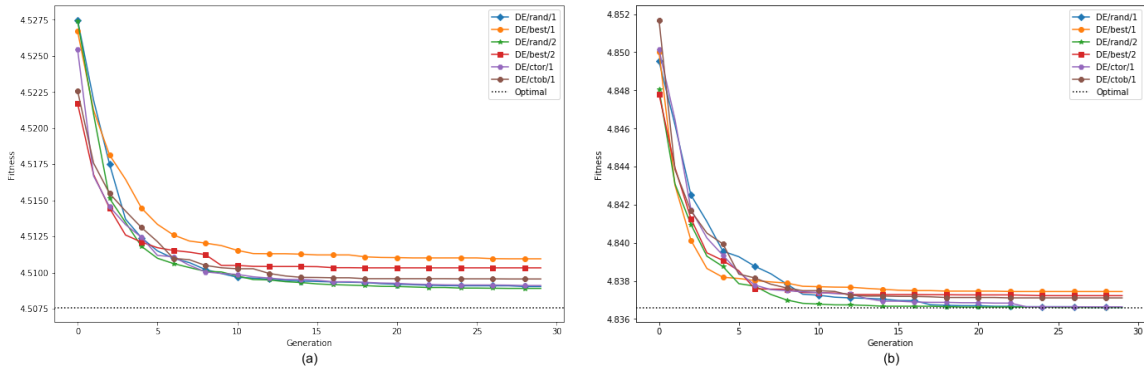


Figure 4: Mutation Strategies on Instance 40.15(a) and 50.1(b)

Figure 4(a) and Figure 4(b) show the convergence curve of the instance of 40 and 50 points, respectively. In the 40-point instance, the best strategy is $DE/rand/2$ which has a 3.1680% minimization average over MST, followed by $DE/rand/1$ with 3.1669% and $DE/ctor/1$ with 3.1661%; despite being good solutions, none of the strategies reached the optimal of 3.1947% reduction average. In the 50-point instance, $DE/rand/1$ and $DE/rand/2$ had the same reduction rate of 2.6375%, followed by $DE/ctor/1$ with 2.6370%. Taking advantage of the quality of solution criterion in an experiment, and converging faster in two experiments, the $DE/rand/2$ operator proves to be the best strategy to use in the next experiment.

5.2. Performance Analysis

In order to obtain an overview of the performance of the algorithm, the next experiment uses the $DE/rand/2/bin$ mutation strategy, with $Np = 50$, $F_1, F_2, Cr = 0.5$, and execution ends when reaching the optimal, run for 50 generations, or in case of stagnation for 20 generations.

For this experiment the problems with 10 to 100 points of the OR-Library are used. Each problem has 15 instances, and for each instance 30 tests are performed.

Let P_x be a problem with x points; $I_{x,y}$ is the y instance of a problem with x points, and $T_{x,y}^z$ is the reduction percentage of a z test of the y instance of a problem with x points; the average per instance is $\overline{I_{x,y}} = (\sum_{z=1}^{30} T_{x,y}^z)/30$, so the average per point is $\overline{P} = (\sum_{y=1}^{15} \overline{I_{x,y}})/15$. If $Ib_{x,y} = \max(T_{x,y}^z | 1 \leq z \leq 30)$ is the best test of an instance y with x points, the best fitness value per point is $P_{best} = (\sum_{y=1}^{15} Ib_{x,y})/15$. The standard deviation per point is obtained through $Std = \sqrt{Sse/(29 * 15)}$, where $Sse = \sum_{y=1}^{15} \sum_{z=1}^{30} (T_{x,y}^z - \overline{I_{x,y}})^2$.

The results are compared to Beasley and Goffinet [Beasley and Goffinet, 1994] and Laarhoven and Ohlmann [Laarhoven and Ohlmann, 2010], because they are algorithms that have stochastic nature and use the Delaunay triangulation. Beasley and Goffinet study is used for historical contextualization, and Laarhoven and Ohlmann is the most recent work that enables a comparison of results, since it uses a benchmark library. Table 2 presents the results.

n	Beasley and Goffinet (1994)	Laarhoven and Ohlmann (2010)		Hybrid Differential Evolution		Opt.
	$\overline{P} \pm Std$	P_{best}	$\overline{P} \pm Std$	P_{best}	$\overline{P} \pm Std$	
10	3.22±1.875	3.25	3.22±0.151	3.25	3.25±0.000	3.25
20	3.12±0.972	3.15	3.13±0.058	3.16	3.16±0.000	3.16
30	2.95±0.754	3.07	3.03±0.068	3.07	3.06±0.057	3.07
40	2.97±0.633	3.14	3.11±0.051	3.14	3.14±0.018	3.14
50	2.92±0.423	3.03	3.00±0.049	3.03	3.03±0.010	3.03
60	3.18±0.371	3.27	3.23±0.058	3.27	3.26±0.024	3.27
70	2.95±0.374	3.11	3.08±0.050	3.11	3.10±0.031	3.11
80	2.92±0.686	3.03	2.98±0.048	3.03	3.00±0.032	3.04
90	2.95±0.502	3.11	3.04±0.062	3.11	3.08±0.033	3.12
100	3.07±0.337	3.26	3.19±0.061	3.25	3.19±0.045	3.27

Table 2: Performance Analysis

In table 2, it can be observed that the DE presents better results than the other heuristics. In DE, the average percentage reduction ($\overline{P}$) is the optimal for 10 and 20 points, and very close to the optimal for 40 and 50 points; for 30,60 and 70 is 0.01% less than the optimal. As for the best percentage of reduction (P_{best}), the DE has the same general average as the Laarhoven and Ohlmann approach, reaching the optimum at 70% of the problems.

6. Conclusions

ESTP is a difficult problem to be solved, and since exact methods can be computationally demanding, this study presents a heuristic for ESTP in $\mathbb{R}^2$ based on meta-heuristic differential evolution. The algorithm has a method that allows the construction of Steiner trees from a set of points (fixed and Steiner points) arranged in the Euclidean plane.

Initially, six mutation strategies for DE were tested with two OR-Library instances. Among the schemes compared, the most efficient was the *DE/rand/2/bin* operator. The following experiment applies this operator in instances of 10 to 100 points in the OR-Library, and compares the results with relevant studies in the literature. As a result, the DE presents good quality solutions in viable computational time. This positive result is due to the use of techniques that enable a good performance, such as initialization of the population with good quality individuals, the application

of corrections in the solutions during the evolution process to avoid the creation of invalid solutions, and the choice of good strategies and evolutionary operators.

Given the importance of STP in the real world, future work includes the investigation and creation of adaptive mutation strategies for ESTP in $\mathbb{R}^d$ with $d > 2$, or with instances that have a high number of endpoints; and given the properties of DE, investigate parallel processing techniques in GPU.

References

- Barreiros, J. (2003). An hierarchic genetic algorithm for computing (near) optimal Euclidean Steiner trees. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*. Springer-Verlag.
- Beasley, J. E. (1990). OR-Library: Distributing test problems by electronic mail. *Journal of the Operational Research Society*.
- Beasley, J. E. and Goffnet, F. (1994). A Delaunay triangulation-based heuristic for the Euclidean Steiner problem. *Networks*.
- Bereta, M. (2017). Monte carlo tree search algorithm for the Euclidean Steiner tree problem. *Journal of Telecommunications and Information Technology*.
- Bereta, M. (2018). Baldwin effect and lamarckian evolution in a memetic algorithm for Euclidean Steiner tree problem. *Memetic Computing*.
- Biniaz, A. and Dastghaibifard, G. (2012). A faster circle-sweep Delaunay triangulation algorithm. *Advances in Engineering Software*.
- Brazil, M., Graham, R. L., Thomas, D. A., and Zachariasen, M. (2013). *On the Story of Euclidian Steiner Tree Problem*. Archive for History of Exact Sciences.
- Deza, M. M. and Deza, E. (2009). *Encyclopedia of Distances*. Springer Berlin Heidelberg.
- Dreyer, D. and Overton, M. (1998). Two heuristics for the Euclidean Steiner tree problem. *Journal of Global Optimization*.
- Du, D. and Hu, X. (2008). *Steiner Tree Problems In Computer Communication Networks*. World Scientific Publishing Co., Inc.
- Fampa, M. and Anstreicher, K. M. (2008). An improved algorithm for computing Steiner minimal trees in Euclidean d-space. *Discrete Optimization*.
- Forte, V. L. d., Montenegro, F. M. T., Brito, J. A. d. M., and Maculan, N. (2015). Iterated local search algorithms for the Euclidean Steiner tree problem in n dimensions. *International Transactions in Operational Research*.
- Garey, M., Graham, R., and Johnson, D. (1977). The complexity of computing Steiner minimal trees. *SIAM Journal on Applied Mathematics*.
- Gilbert, E. and Pollak, H. (1968). Steiner minimal trees. *SIAM Journal on Applied Mathematics*.
- Goldberg, D. E. (1989). *Genetic algorithms in search, optimization and machine learning*. Addison Wesley.

- Hanan, M. (1966). On Steiner's problem with rectilinear distance. *SIAM Journal on Applied Mathematics*.
- Hwang, F. K., Richards, D. S., and Winter, P. (1992). *The Steiner Tree Problem*. Elsevier Science Publishers B. V.
- Jesus, M., Jesus, S., and Márquez, A. (2004). Steiner trees optimization using genetic algorithms. In *Technical report*. Centro de Simulação e Cálculo.
- Juhl, D., Warme, D. M., Winter, P., and Zachariasen, M. (2018). The geoSteiner software package for computing Steiner trees in the plane: an updated computational study. *Mathematical Programming Computation*.
- Koch, T. and Martin, A. (1998). Solving Steiner tree problems in graphs to optimality. *Networks*.
- Kruskal, J. B., Jr. (1956). On the shortest spanning tree of a graph and the traveling salesman problem. *Proc. Amer. Math. Soc.*
- Laarhoven, J. W. and Ohlmann, J. W. (2010). A randomized Delaunay triangulation heuristic for the Euclidean Steiner tree problem in $\mathbb{R}^d$. *Journal of Heuristics*.
- Lundy, M. (1985). Applications of the annealing algorithm to combinatorial problems in statistics. *Biometrika*.
- Price, K., Storn, R., and Lampinen, J. (2005). *Differential Evolution: A Practical Approach to Global Optimization*. Springer.
- Sinclair, D. (2010). S-hull: a fast radial sweep-hull routine for Delaunay triangulation. *CoRR*.
- Smith, J. M., Lee, D. T., and Liebman, J. S. (1981). An $o(n \log n)$ heuristic for Steiner minimal tree problems on the Euclidean metric. *Networks*.
- Smith, W. D. (1992). How to find Steiner minimal trees in Euclidean d-space. *Algorithmica*.
- Storn, R. and Price, K. (1997). Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*.
- Thompson, E. A. (1973). The method of minimum evolution. *Annals of human genetics*.
- Wang, L. and Du, D.-Z. (2002). Approximations for a bottleneck Steiner tree problem. *Algorithmica*.
- Warme, D. M. (1998). *Spanning Trees in Hypergraphs with Applications to Steiner Trees*. PhD thesis, University of Virginia.
- Winter, P. (1985). An algorithm for the Steiner problem in the Euclidean plane. *Networks*.
- Winter, P. and Zachariasen, M. (1997). Euclidean Steiner minimum trees: An improved exact algorithm. *Networks*.
- Yang, B. (2006). A hybrid evolutionary algorithm for the Euclidean Steiner tree problem using local searches. In *Knowledge-Based Intelligent Information and Engineering Systems*.