A Genetic Algorithm Approach for the Euclidean Steiner Tree Problem with Soft Obstacles

Manou Rosenberg
The University of Western Australia
Perth, Australia
manou.rosenberg@uwa.edu.au

Tim French
The University of Western Australia
Perth, Australia
tim.french@uwa.edu.au

Mark Reynolds
The University of Western Australia
Perth, Australia
mark.reynolds@uwa.edu.au

Lyndon While
The University of Western Australia
Perth, Australia
lyndon.while@uwa.edu.au

ABSTRACT

In this paper we address the Euclidean Steiner tree problem in the plane in the presence of soft and solid polygonal obstacles. The Euclidean Steiner tree problem is a well-known NP-hard problem with different applications in network design. Given a set of terminal nodes in the plane the aim is to find a shortest-length interconnection of the terminals allowing further nodes, so-called Steiner points, to be added. In many real-life scenarios there are further constraints that need to be considered. Regions in the plane that cannot be traversed or can only be traversed at a higher cost can be approximated by polygonal areas that either need to be avoided (solid obstacles) or come with a higher cost of traversing (soft obstacles). We propose a genetic algorithm that uses problem-specific representation and operators to solve this problem and show that the algorithm can solve various test scenarios of different sizes. The presented approach appears to outperform current heuristic approaches for the Steiner tree problem with soft obstacles and was evaluated on larger test instances as well.

CCS CONCEPTS

• **Applied computing** → **Mathematics and statistics**; • **Computing methodologies** → *Randomized search*; • **Mathematics of computing** → *Trees*.

KEYWORDS

Steiner tree problem, Soft obstacles, Genetic algorithm

ACM Reference Format:
Manou Rosenberg, Tim French, Mark Reynolds, and Lyndon While. 2021. A Genetic Algorithm Approach for the Euclidean Steiner Tree Problem with Soft Obstacles. In *2021 Genetic and Evolutionary Computation Conference (GECCO'21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449639.3459397

GECCO '21, July 10–14, 2021, Lille, France

ACM ISBN 978-1-4503-8350-9/21/07…$15.00
https://doi.org/10.1145/3449639.3459397

1 INTRODUCTION

The Euclidean Steiner tree problem (ESTP) without the consideration of any obstacles was first introduced by the mathematician Gergonne in 1811 [10] and is a well-known NP-hard optimisation problem [8]. Similarly to finding a minimum spanning tree, the aim is to connect a number of nodes (called *terminals*) in the plane minimising the total distance. Additionally, the Euclidean Steiner minimum tree allows for further intersection points (*Steiner points*) to be added. The additional task of considering obstacles that either need to be avoided or can only be traversed with a penalty per intersecting unit (*soft obstacles*) makes the problem even more complex. While there are several approaches to the standard Steiner tree problem, the literature on the this problem with soft obstacles is limited. Approaches that consider the Euclidean Steiner tree problem with some kind of constraints mostly use a discretised structure of the plane or the obstacles at some point to reduce the complexity of the problem (see [3, 7, 9] etc.). That is why we propose an evolutionary approach that is adapted to the specific problem structure of the ESTP and exploits its geometric properties. A two-part chromosome with a combination of a fixed and variable length part is introduced and the crossover and mutation operators are adapted accordingly. Furthermore, the separation between the genotype and phenotype of the chromosome is noteworthy in this case. While the genotype only contains information of Steiner and obstacle corner points, the phenotype represents the interconnected Steiner minimum tree.

1.1 The ESTP

Despite the ESTP stating no further requirements, Euclidean Steiner minimum trees have several interesting geometric characteristics (see e.g. [2]). Some of those are summarised in the following:

Properties 1. *Consider the ESTP with n terminals. A solution to this problem is called Euclidean Steiner Minimum Tree (ESMT or SMT) and is a tree in the plane with the following properties:*

(1) *There are at most $n-2$ Steiner points.*
(2) *Terminals are connected to at most three other nodes and Steiner points are adjacent to exactly three other nodes (degree condition).*
(3) *Each pair of edges in the tree meets at an angle of at least $2\pi/3$ and the three edges at the Steiner point meet exactly at an angle of $2\pi/3$ (angle condition).*

The problem of finding a minimal length connection between a set of nodes has several applications, for example in the area of electricity, communication and computer chip network design, or in parallel computing (*e.g.* [6], [1] [11]). In most real-world applications though further constraints apply. Planning an infrastructure network for example, such as an electricity or road networks, the presence of restricted regions, like nature reserves, private property, risk prone areas or other constraints, can make it infeasible to interconnect the nodes via a standard Steiner tree. That is why the obstacle-avoiding Steiner tree is considered. In the planar obstacle-avoiding Euclidean Steiner tree problem, restricted regions can be represented as non self-intersecting polygons. The polygons are considered *solid obstacles* in the graph that cannot have any intersection with a resulting Steiner minimum tree. However, in some real-life cases a region does not necessarily need to be avoided, instead intersecting it comes at a higher risk or cost. In terms of graphs, these regions are called *soft obstacles*. In this paper, we consider the Euclidean Steiner tree problem with polygonal soft obstacles with a homogeneous weight per unit length. We formulate this problem as follows:

Definition 1.1. Consider a set of n terminals and o simple polygonal obstacles with crossing weights w_o in the plane.
Find a connected network T that spans all terminals and has minimal total weighted length but can include further nodes.

If an edge of the tree intersects with the inside of a soft obstacle, the length of the intersecting part is multiplied with the crossing weight w_o of this obstacle. In case of a solid obstacle the traversed obstacle causes a high penalty cost. Going along the the edges or only touching the boundary of an obstacle comes at no higher cost. Equation 1 in Section 3 outlines this in more detail. An example for a Euclidean Steiner tree problem instance with soft and solid obstacles and ten terminals is portrayed in Figure 1.

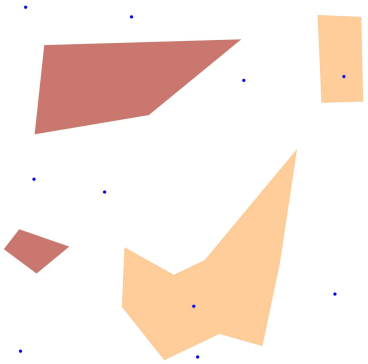


Figure 1: Example of a Steiner tree problem instance with soft (light orange) and solid (red) obstacles

In the case of obstacles in a Euclidean Steiner tree problem, it can be observed that Properties 1 for a minimum Steiner tree solution still hold locally, where there is no obstacle boundary.

Observation 1. *Consider a solution to a Euclidean Steiner tree problem with n terminals and soft or solid obstacles with a total number of k obstacle corners.*

(1) *We distinguish between three types of Steiner points: Steiner points that lie either inside or outside an obstacle but not on any obstacle boundary, Steiner points that are on the boundary of an obstacle but not a corner point and obstacle corner points (hereinafter not called Steiner points in this paper).*
(2) *Steiner points in- or outside of obstacles have a degree of exactly 3 and the three incident edges meet at an angle of $2\pi/3$ (compare Property 1.3), while Steiner points on the boundary or obstacle corner points can have degree 2.*
(3) *If the problem instance only contains solid obstacles a Steiner minimum tree contains Steiner points of degree 3 and obstacle corner points only.*

Some of the above mentioned properties are encoded in the chromosome representation and operators of the proposed genetic algorithm to avoid exploring many infeasible solutions.

2 LITERATURE REVIEW

The Steiner tree problem with obstacles or other constraints has several real-world applications. The paper by Fletcher *et al.* [6] for example deals with connecting electricity consumers and substations in an electricity distribution network in geographically constrained rural areas using a genetic algorithm. The constrained areas are represented as a rasterised map and a shortest path algorithm is used to determine the connection between terminals. A genetic algorithm then alters the connections by adding a Steiner point connecting three selected terminals, removing Steiner points, or reconnecting nodes to a different part of the graph. While this approach appears to create feasible solutions for the specific application, it does not aim to find the optimal solution for a minimum Steiner tree problem with obstacles. Another application to a power distribution network was presented by Duan *et al.* [5]. Their "genetic shortest-path" algorithm does not consider the Steiner tree problem with obstacles but includes complex-flows on the networks branches and voltage constraints. The authors translate their application into the capacitated Steiner tree problem in graphs. Most of the approaches for solving the Euclidean Steiner tree problem deal with solid obstacles only. For example, Provan *et al.* [18] developed an approximation scheme on a visibility graph to find a Steiner tree solution considering solid obstacles. By constructing a visibility graph first, the problem is transformed into the Steiner tree problem in graphs. The authors of [16] present a polynomial-time approximation scheme for the Steiner tree problem with polynomial solid obstacles based on a similar idea. They also make use of the visibility graph and then place a grid locally around each of its edges to find Steiner candidate points. Weng *et al.* [20] propose an algorithm for the Euclidean Steiner tree problem with one solid obstacle only and further restricting conditions. Their approach is based on Melzak's algorithm [14] for the Steiner tree problem without obstacles.

Despite several areas of application, the literature on the Euclidean Steiner tree problem with soft obstacles is sparse. Garrote *et al.* [9] consider the Euclidean Steiner tree problem for Communication networks where disaster-prone areas are represented as soft obstacles. The authors claim to be the first to consider the ESTP with soft obstacles and suggest a deterministic approach. Their algorithm makes use of Dijkstra's shortest path algorithm to calculate

the shortest connection between terminals while discretising the boundaries of included obstacles. Further Steiner points are then added at angles of less than $2\pi/3$ using a local search procedure considering soft obstacles. The authors test their algorithm on a range of solid obstacle instances and compare their results to the ones computed by an exact method by Zachariasen*et al.*[21]. Some of them are used as a comparison in this paper as well.

3 METHODOLOGY

To determine a minimised connection of terminal nodes in the presence of solid and/or soft obstacles in the plane, we propose a genetic algorithm with problem-specific chromosome representation and mutation and crossover operators. The presented GA finds solutions to the (Euclidean) Steiner tree problem with soft and solid obstacles by using a mix of adapted mutations and some of the properties described in Section 1 and is hereinafter called *StOBGA*. The details of the algorithm are described in the following:

3.1 GA for the ESTP with Soft Obstacles

The StOBGA is an adaptive genetic algorithm that uses three types of mutations with changing probability and one-point crossover to determine a minimised connection of terminals in the plane. Assuming an ESTP with n terminal nodes and soft or solid obstacles with a total number of k obstacle corner nodes. An example is shown in Figure 1. The aim for the StOBGA is to determine the number and location of Steiner points needed as well as which obstacle corner points should be used in the solution such that the total distance of the tree connecting all the terminals is minimised. Each possible solution can be evaluated using its fitness value, which is in this case the total (weighted) distance of the tree that is to be minimised. Once the Steiner and obstacle corner points of a possible solution are established the Steiner tree can be determined using a weighted minimum spanning tree algorithm, such as Prim's algorithm [17]. The weight of each edge connecting two nodes represents the Euclidean distance between these nodes and a multiplied penalty cost per unit for crossing obstacles. To avoid infeasible solutions the weight is set to infinity if the edge intersects the inside of a solid obstacle. This forces the algorithm to avoid even very small intersections with any solid obstacle. In case of the edge crossing a soft obstacle, the weight is determined by

$$\text{edge weight} = dist_{out} + \sum_{ob} dist_{ob} \cdot c_{ob} \,, \tag{1}$$

where $dist_{out}$ is the length of the edge's part that lies outside of any obstacle and $dist_{ob}$is the length of the part that lies inside an obstacle ob. Multiplication with the crossing factor $c_{ob}(> 1.0)$ penalises traversing through soft obstacles. Going along an obstacles edge or touching the obstacles boundary is not penalised. Connecting nodes using the boundary of an obstacle comes with no higher cost. Each generation the genetic algorithm evolves through crossover and mutations. Tournament selection (see e.g. [15]) is used to decide which of the population's chromosomes are chosen to produce offspring as well as to select which chromosomes will die. First, for each tournament the fittest solution is selected as the parent chromosomes to produce offspring. After creating offspring through mutation and crossover, tournament selection is used again. For each tournament the least fit solution is selected to die such that the population size remains the same. A tournament size greater than two therefore guarantees that the fittest solution always survives. When the fittest solution does not improve over a certain number of generations the StOBGA has converged and the algorithm stops.

3.1.1 Representation and Initialisation. Each chromosome in the algorithm represents a possible Steiner tree solution that belongs to a given Steiner tree problem with obstacles. Therefore, the chromosome has access to the given terminals and obstacles and stores information on the Steiner point locations and obstacle corners to build the Steiner tree. The StOBGA uses a chromosome representation that consists of two parts: One of variable length and a binary part with fixed length. The first part stores a list of Steiner points, which contain the x- and y-coordinates of the points $(< x_1, y_1 >, < x_2, y_2 >, \dots)$. This list varies in length and can also be empty if no Steiner points are used in the represented solution. The second part consists of a binary string of fixed length that represents whether an obstacle corner node is included in the possible solution. Therefore, each chromosome has the following structure:

$$\{< x_1, y_1 >, < x_2, y_2 >, \dots\} + [0 \mid 1 \mid 1 \mid 1 \mid 0 \mid 0 \mid 1 \mid 0 \mid 0 \mid 0]$$

The chromosome can be translated to represent an interconnected Steiner tree by connecting all terminals, Steiner points and included obstacle corners using a weighted MST algorithm like Prim's algorithm [17]. The weights of each straight-line connection between the points is calculated as described above.

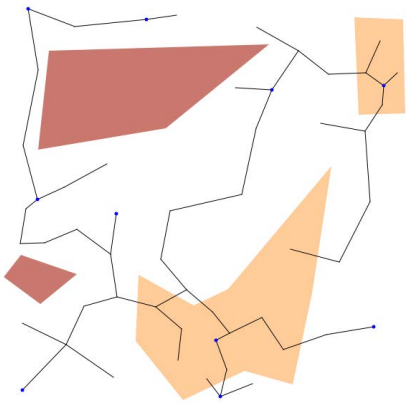


Figure 2: Initial chromosome example with soft (light orange) and solid (red) obstacles

In addition to the representation of a chromosome in a genetic algorithm, the way the first generation is initialised can also play a crucial role in the success of the algorithm. An unfavourably chosen initial population could for example lead to early convergence to a local minimum. Different initialisation techniques have therefore been considered (see e.g. [13]). In our case it is important that the initial population is diverse but also that at least some of the initial Steiner point placements are not "too far away" from their optimal location. In case of solid obstacles being included, an initial population starting with only infeasible solutions, *i.e.* at least one of the edges crosses a solid obstacle, can lead to a slow start or in the worst case to convergence to an infeasible solution. Therefore,

we have chosen three different types of initial chromosomes that start the first generation:

(1) The first one uses the Delaunay triangulation [4] of all terminals and obstacle corner nodes combined. For each triangle we add a Steiner point located in the triangle's centre unless this would cause a Steiner point positioned inside of a solid obstacle. The second part of the chromosome is set to a binary string with all zeros, i.e. no obstacle corner points are included. An initial type of Steiner tree for the instance example introduced in Figure 1 is shown in Figure 2.
(2) The second type of chromosome contains $n+k$ Steiner points with random positions, where n is the number of terminals and k is the number of obstacle corners. Again, placement inside of solid obstacle is excluded.
(3) The third one leaves the list of Steiner points empty and instead randomly flips some of the genes in the second binary part of the chromosome.

After filling part of the initial population with the three chromosome types described above, the remaining elements are added by creating offspring of the already included chromosomes through crossover and mutation. These operators are described in detail in the following paragraph.

3.1.2 Mutation and Crossover Operators. To ensure sufficient exploration of the fitness landscape three different types of mutations and one crossover operator are used to create offspring. Out of those chromosomes that were selected to evolve, two parent chromosomes are chosen randomly to create two children by using one-point crossover ($p_{cross} = 1.0$). The child chromosomes then undergo one of the three mutations with adapting probability. For the crossover operation the width of the problem instance, including terminals only, is calculated and a random point on the x-axis inside of this range is chosen. A vertical line on this point splits the parent chromosomes into two parts as shown in Figure 3. The first child inherits all Steiner points and all included obstacle corners on the left of the split line from the first parent and all Steiner and obstacle corner points on the right of the line from the second parent chromosome. The second child is created vice-versa.

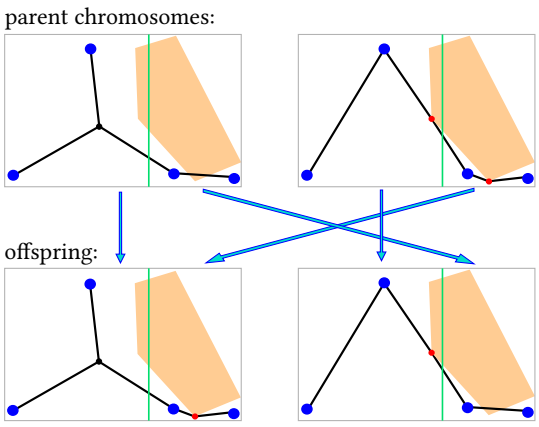


Figure 3: Crossover of StOBGA

Having produced the offspring, each child chromosome performs one of three mutations (visualised in Figure 4). In a standard GA a mutation typically consists of flipping each gene in the chromosome with a small probability $p_{gene} < 1.0$. In our case a gene is either one of the Steiner points or one of the bits in the binary string. Considering a chromosome with s Steiner points for a problem instance with k obstacle corners, there are $s+k$ genes. The problem-specific mutations are described as follows:

- The *flipMove* mutation: Going through each gene and with probability $p_{gene} = 1/(s+k)$ either moving it, if it is a Steiner point, or flipping it, if it is a bit of the second chromosome part. The move action alters the Steiner point coordinates by small values $\pm\, x_{move}$ and $\pm\, y_{move}$, which are random values between 0 (exclusive) and a certain move range m_{range}. The range for the move action is initially set to the average Euclidean distance between terminals but decreases with increasing number of generations (no_{gen}) as follows:

$$m_{range} = avgDist_{terminals} * \max\left\{1 - \frac{no_{gen}}{1000},\ 0.01\right\} \qquad (2)$$

- The *addSteiner* mutation: For this mutation we need to calculate the weighted minimum spanning tree of the current chromosome. Then each node in the tree is checked for an angle that is less than $2\pi/3$. One randomly selected node is determined including the two neighbours building the small angle. The Steiner point position between the three adjacent nodes is calculated and the new Steiner point is added to the offspring. If there is no small angle in the graph, a Steiner point at a random position (avoiding placement inside of solid obstacles) is created and added.
- The *removeSteiner* mutation: Throughout the algorithm some Steiner points might not be needed anymore and have moved close to a straight line. Since unnecessary Steiner points in the chromosome will slow down the convergence of the algorithm the third mutation removes a Steiner point of degree two, such that the two adjacent nodes can be connected directly (see Figure 4c).

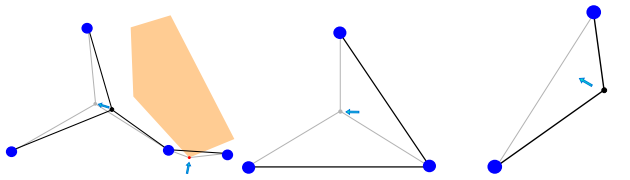


Figure 4: Mutations of StOBGA

The decreasing move range for the *flipMove* mutation follows the idea that the Steiner points move closer to their optimal position over time. Initially, a Steiner point move might need to be large enough to change the topology of the graph and get to its correct place. Once it is close to its optimal position the small move range is helpful to find the exact location.

Just like the move range the probabilities for choosing each mutation ($p_{flipMove}$, p_{add}, and p_{remove}) adapt over time as well.

The probability of the first mutation is initially set high (p_{max}) while the other two mutations are initially performed with only a small probability. Since one of the mutations must be performed and to reduce the number of parameters the probabilities are determined in the following way:

$$p_{flipMove} = \max\left\{p_{max} * \left(1 - \frac{no_{gen}}{1000}\right),\ p_{min}\right\} \quad (3)$$

$$p_{add} = p_{remove} = 1 - \frac{p_{flipMove}}{2} \quad (4)$$

The reason for adapting the probabilities is that the different types of mutations are more or less valuable depending on how much of a potential optimal solution is already determined. For instance, removing Steiner points of degree two too early might cause the removal of too many Steiner points that through the evolutionary process would have become necessary Steiner points in a different position. Using a high probability $p_{flipMove}$ for the *flipMove* mutation initially and then increasing the probabilities for the other mutation operators have shown to be most successful. Table 1 lists the parameter values for the StOBGA that were chosen to be fixed for the range of different problem instances. These values were found by performing a grid search over different parameter values.

Table 1: Parameters used for the StOBGA

parameter	value
population size μ	500
offspring size λ	$\frac{1}{3} * \mu$
tournament size	5
p_{max} for *flipMove* mutation	0.99
p_{min} for *flipMove* mutation	0.60

Finally, when the StOBGA has converged one additional chromosome is added to the final population. This last step performs only fine adjustments to the location of some Steiner points. Given three fixed coinciding nodes, finding the exact position of the Steiner point becomes the *Fermat-Torricelli* problem. The exact position can be calculated using *Simpson lines* (see [19], [12]). Therefore, the lastly added chromosome has each Steiner point of degree three moved to its calculated exact position assuming its three neighbours are fixed. When added to the final population the chromosome is compared to the previously best solution and the best result is returned.

3.2 Iterative Insertion of Steiner points

A simplified iterative algorithm was implemented for a baseline comparison to the StOBGA results. The basic idea can be described as follows: Starting with a solid obstacle-avoiding spanning tree, Steiner points are added where two edges meet with an angle of less than $2\pi/3$ if this can reduce the total length of the tree. The aim was to build a relatively fast method that creates feasible solutions avoiding solid obstacles and considering the intersection of soft obstacles when favourable. The spanning tree is determined again by using a combination of Prim's and Dijkstra's shortest path algorithms to determine the weights. This method has also been described by Garrote *et al.* [9]. The possible Steiner point locations are then calculated as above for each occurrence of three nodes building an angle of less than $2\pi/3$. Once the coordinates are calculated there are four possible cases:

- case 1: The Steiner point is added at its calculated position and the two edges are removed.
- case 2: The angle occurs at a Steiner point of degree three. Instead of adding another Steiner point the position of this one is adjusted using the three adjacent nodes to calculate the Fermat-Torricelli coordinates.
- case 3: The angle occurs at a Steiner point of degree three. The Steiner point and its edges are removed and the two edges are added to connect the three adjacent nodes directly.
- case 4: The current state remains unchanged.

The outcome for each of these cases is determined and the one with the lowest weighted distance is performed. This step is repeated until no changes would improve the total weighted distance.

4 EXPERIMENTAL RESULTS

The StOBGA was tested on a total number of 43 instances with soft obstacles (or a mix of soft and solid obstacles) and 40 instances with solid obstacles only. The instances contain a range of different obstacle shapes and combinations that will be discussed in greater detail later on. Additionally, the instances include a range of different sizes, *i.e.* various number of terminals that need to be connected. While the above mentioned paper [9] showed results of their heuristic method on small problem instances with up to 30 terminals and with a maximum of around 40 obstacle corner points, the StOBGA has also been tested on some larger instances with 500 and up to 1000 terminals as well as some test cases with a larger number of obstacle corner points, up to 150. The 18 problem instances with solid obstacles from Zachariasen *et al.* [21] that were used as a comparison by Garrote *et al.* [9] could be approximated and the StOBGA was run 30 times on each instance. It could on average outperform the previous heuristic method for soft obstacles showing an average error to the exact solution of 0.044% over all runs and all instances. For most problem cases all of the runs could approximate the exact solution within three decimal places and only three instances showed a worst run error of more than 1%, the overall worst run error on one instance being just under 4%. The example in Figure 5 shows the solution computed by the StOBGA to one of the 18 instances used in the paper discussed above. While the heuristic method developed by Garrote *et al.* leads to an error of around 10% on this instance the StOBGA has an average error of less than 0.2% and a worst case error of 4% over 30 runs on this instance.

The 83 problem instances that we created to test the StOBGA can be used to test with other methods that solve the ESTP with soft (and solid) obstacles[1]. To make results comparable the original instance numbers are used throughout this paper.

4.1 Solid Obstacles

As the StOBGA can be applied to both, cases with soft and solid obstacles, a range of problem instances with solid obstacles in various numbers and shapes have been constructed. The results computed

[1] https://github.com/ManouRosenberg/SteinerTreeProblemWithSoftObstacles

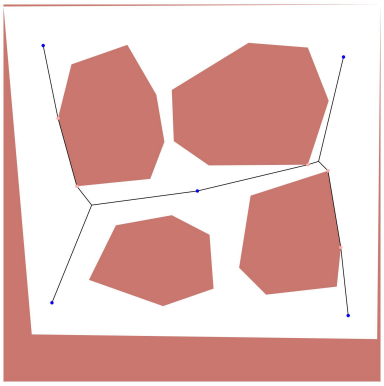
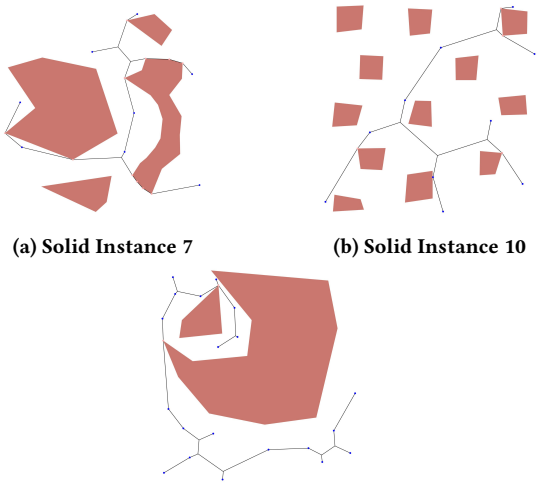


Figure 5: Solution computed by the StOBGA to one of the instances used in [9].

by the StOBGA for some of these are exemplified in Figure 6. The solved instances demonstrate that the algorithm effectively finds Steiner point locations and uses obstacle corners to avoid crossing any part of the solid obstacles.



(a) Solid Instance 7 **(b) Solid Instance 10**

(c) Solid Instance 20

Figure 6: Some examples with solid obstacles solved by the StOBGA

Figure 7 illustrates how the algorithm evolves over the generations. The three lines display the total distance of the best solution of each generation for the three problem instances in Figure 6. Since the chosen test cases include different number of obstacles and terminals the algorithm obviously results in solutions of differently weighted length but the behaviour for each instance is similar. In the first 50 generations the total distance decreases rapidly while the later generations show almost no improvement, meaning the algorithm has converged to a minimised solution. The results in numbers are presented in Table 2.

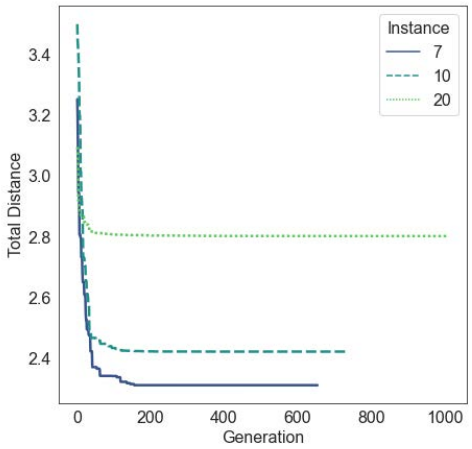


Figure 7: Convergence of the StOBGA for some solid problem instances

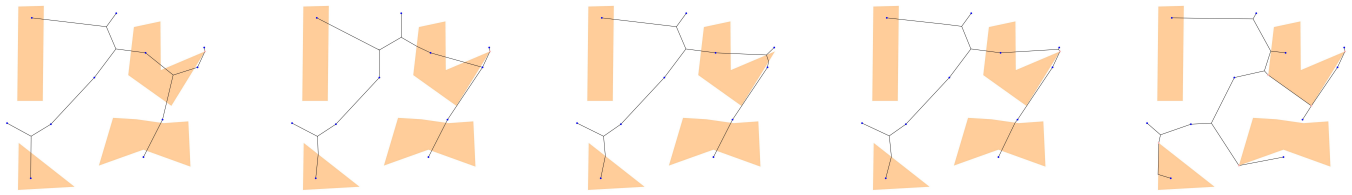
Table 2: The STOBGA on some example instances with solid obstacles

	inst. 7	inst. 10	inst. 20
# terminals	8	10	20
# obstacle corners	37	48	15
Total Distance (best out of 30 runs)	2.3102	2.4211	2.8023
# Steiner points used	3	4	7
# obstacle corners used	11	4	2

4.1.1 Comparison to iterative approach. The iterative method that was described in Section 3.2 is a simplified heuristic approach for the ESTP with soft and solid obstacles. As Figure 9 shows, a solution found by the iterative method still avoids crossing solid obstacles but can clearly be further optimised in several parts by finding better Steiner point locations and integrating more obstacle corners. In fact, on almost all of the tested instances the StOBGA could determine a Steiner tree that had a shorter total length than the solution provided by the iterative method. In only some simple problem cases both algorithms could reach the same solution. However, the brief comparison outlined in Table 3 indicates that the iterative method is significantly faster than the StOBGA and for this example case the difference in the total length is less than 5%. Therefore, the iterative method might be an alternative when a solution must be found in a timely manner and lower quality solutions can be accepted.

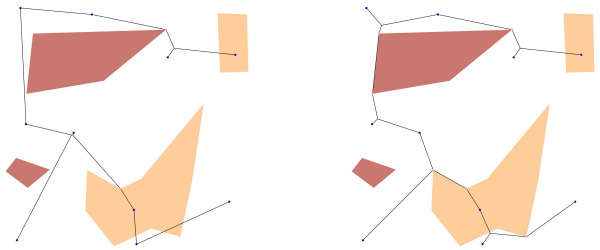
4.2 Soft Obstacles

Similar to the problem instances with solid obstacles we have developed a number of different test cases with soft obstacles only and a



(a) weight: 1.1 **(b) weight:** 1.25 **(c) weight:** 1.5 **(d) weight:** 1.75 **(e) weight:** 5.0

Figure 8: The StOBGA for soft obstacles with varying obstacle weights



(a) Iterative method **(b) StOBGA result**

Figure 9: Comparison of the StOBGA and the iterative method

Table 3: Example case with 10 terminals and 20 obstacle Corners

	Iterative Method	StOBGA
# Steiner points	3	4
# included obstacle corners	2	6
Total Distance	2.7540	2.6261
Time [s]	0.066	33.29

mix of both, soft and solid obstacles. Furthermore, we examined different Steiner tree solutions as a result of varying obstacle weights while the number and position of terminals and obstacles are the same. A problem instance with four soft obstacles with increasing crossing weight ranging from 1.1 to 5.0 has been used. The five different solutions developed by the StOBGA are shown in Figure 8. In the case displayed in Figure 8a, which has the lowest obstacle crossing weight, the resulting Steiner tree crosses the obstacles with a large part of its edges only avoiding an obstacle when the distance added by the way around the obstacle is small. In case of a high crossing weight, as used for the obstacles in Figure 8e, the resulting Steiner tree only intersects an obstacle where a terminal inside of the obstacle must be connected. Figure 10 outlines how the algorithm evolves for the different obstacle crossing weights. While all of the curves look similar, the different lengths of the lines indicate that the StOBGA needs less generations to find its final solution when the obstacle weights are either low or high. Low obstacle weights allow the algorithm to partly ignore the obstacles, while high obstacle weights causes the algorithm to avoid potential intersections with the obstacle. In both cases, this limits the number of possibilities that need to be considered by the StOBGA.

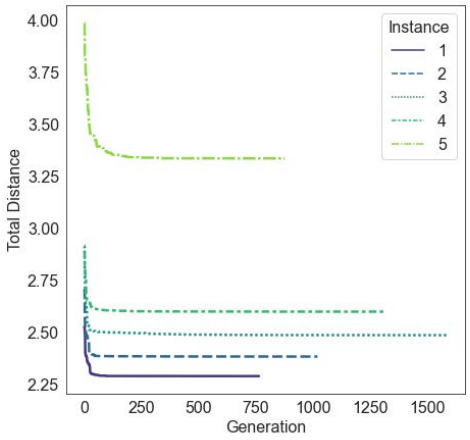
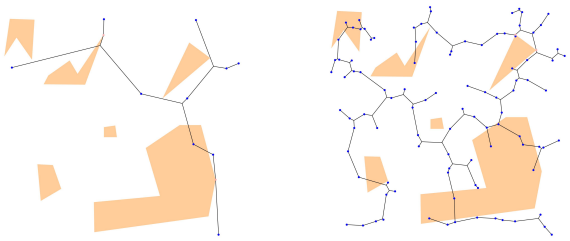


Figure 10: Similar rate of convergence for different obstacle weights

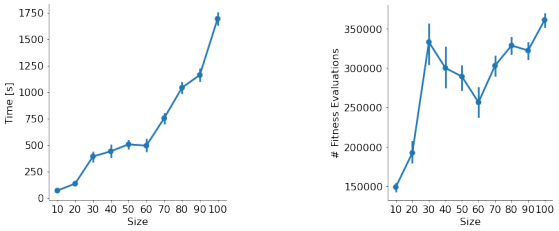
4.3 Instances of Different Sizes

Finally, the proposed algorithm is also tested on instances of varying size. The size of an instance of the ESTP with obstacles comprises two aspects: the number of terminals and the number of obstacle corners. To be able to scale a problem instance but also keep it comparable, we choose a fixed set of obstacles and consider the behaviour of the StOBGA over an increasing number of terminals ranging from 10 to 100. The test scenarios were developed for solid and soft obstacles separately but to avoid repetition we illustrate the results on the soft obstacle cases only. The optimised Steiner tree for the first and last instance are shown in Figure 11. For each of the 10 instances the algorithm was run 30 times recording the total length of the final chromosome, the time and the number of fitness evaluations of each run. While the time that the algorithm needs before it converges to a final solution appears to increase exponentially with the size of the problem instance, the number of fitness evaluations fluctuates. The length of the error bars included in Figure 12 also indicate more diverse values within the 30 runs on each instance.



(a) 10 terminals (b) 100 terminals

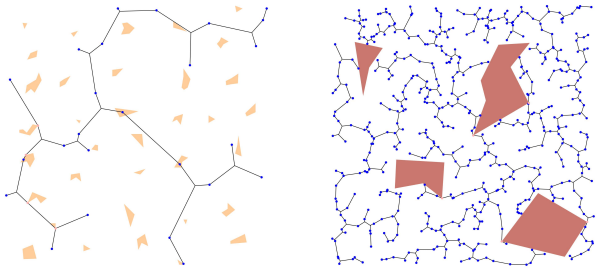
Figure 11: Increasing number of terminals with fixed soft obstacles



(a) Time [s] (b) Number of fitness evaluations

Figure 12: Increasing number of terminals with fixed soft obstacles

To show that the StOBGA also generates feasible solutions for larger problem instances, Figure 13 demonstrates the solutions to an instance with soft obstacles including 150 corner nodes and a solution to a solid obstacle instance with 500 terminals. While the computed solution for the latter does not intersect any of the obstacles' boundaty edges, Figure 14 outlines that the algorithm converges significantly slower in the case of such a large instance. Depending on the application, this might not necessarily be a problem. When computing a Steiner minimum tree with obstacles for planning an electricity network (as in [6]) or for example when constructing a communication network (see [9]) the planning phase can take several years, so the algorithm's slower run time on large instances does not pose a restriction.



(a) Instance with 150 obstacle corners (b) Instance no. 501 with 500 terminals

Figure 13: Examples with large numbers of terminals or obstacle corners

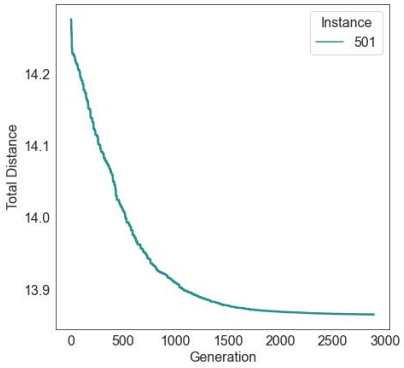


Figure 14: Slower convergence for larger instance

5 CONCLUSIONS

The Euclidean Steiner tree problem with soft and solid obstacles has several potential real-life applications. Despite its useful properties the problem has not been approached much yet. In the 2019 paper by Garrote *et al.* [9], the authors claim that they are the first to attempt this problem. While there are some approaches attempting to solve the Euclidean Steiner tree problem with solid obstacles the literature on this problem for soft obstacles indeed remains sparse. The complexity of the problem comes with increased difficulty for deterministic methods to obtain good solutions. In cases like this evolutionary algorithms have been applied successfully in the past due to their adaptive nature and flexibility. Therefore, a genetic algorithm (StOBGA) with problem-specific representation and operators was introduced to solve the ESTP with soft and solid obstacles. We have shown that the proposed algorithm performs well on a range of different scenarios with soft and solid obstacles. It has also successfully been applied to larger problem instances with up to 1000 terminals. A versatile range of test instances has been created to allow future approaches to this problem to be tested and compared. This set of test instances could potentially be extended to also contain problem instances of obstacles with non-homogeneous weight. These obstacles could for example represent different landscapes in an infrastructure network planning problem. Testing and potentially adapting the StOBGA on these problems with more complex soft obstacles may be interesting to investigate in the future.

ACKNOWLEDGMENTS

The author Manou Rosenberg would like to thank the Forrest Research Foundation for their funding and support through the Forrest Research scholarship.

REFERENCES

[1] Gaurav Ajwani, Chris Chu, and Wai Kei Mak. 2011. FOARS: FLUTE based obstacle-avoiding rectilinear steiner tree construction. In *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, Vol. 30. 194–204. https://doi.org/10.1109/TCAD.2010.2096571

[2] M. Brazil and M. Zachariasen. 2015. *Optimal Interconnection Trees in the Plane: Theory, Algorithms and Applications* (1st ed.). Springer International Publishing.

[3] Chris S. Coulston. 2003. Steiner minimal trees in a hexagonally partitioned space. *International Journal of Smart Engineering System Design* 5, 4 (oct 2003), 341–346. https://doi.org/10.1080/10255810390224099

[4] B. Delaunay. 1934. Sur la sphere vide. A la memoire de Georges Voronoi. *jour Bulletin de l'Academie des Sciences de l'URSS. Classe des sciences mathematiques et na* 6 (1934), 793–800.

[5] Gang Duan and Yixin Yu. 2003. Power distribution system optimization by an algorithm for capacitated Steiner tree problems with complex-flows and arbitrary cost functions. *International Journal of Electrical Power & Energy Systems* 25, 7 (sep 2003), 515–523. https://doi.org/10.1016/S0142-0615(02)00128-X

[6] James Fletcher, Tyrone Fernando, Herbert H C Iu, Mark Reynolds, and Shervin Fani. 2018. Spatial Optimization for Sparse Distribution Networks in Geographically Constrained Areas. *IEEE Transactions on Power Systems* 33, 6 (2018), 6686–6695. https://doi.org/10.1109/TPWRS.2018.2846407

[7] Ian Frommer and Bruce Golden. 2007. A Genetic Algorithm for Solving the Euclidean Non-Uniform Steiner Tree Problem. In *Extending the Horizons: Advances in Computing, Optimization, and Decision Technologies*. Springer US, Boston, MA, 31–48. https://doi.org/10.1007/978-0-387-48793-9_3

[8] M. R. Garey, R. L. Graham, and D. S. Johnson. 1977. The Complexity of Computing Steiner Minimal Trees. *SIAM J. Appl. Math.* 32, 4 (1977), 835–859.

[9] Luis Garrote, Lucia Martins, Urbano J. Nunes, and Martin Zachariasen. 2019. Weighted Euclidean Steiner Trees for Disaster-Aware Network Design. In *15th International Conference on the Design of Reliable Communication Networks, DRCN 2019*. IEEE, 138–145. https://doi.org/10.1109/DRCN.2019.8713664

[10] Joseph Diaz Gergonne. 1811. Solutions purement géométriques des problèmes de minimis. *Annales de Mathématiques pures et appliquées* 1, 1 (1811), 375 – 384.

[11] Frederick C. Harris and Rakhi Motwani. 2013. Steiner minimal trees: An introduction, parallel computation, and future work. *Handbook of Combinatorial Optimization* 5-5 (2013), 3133–3177. https://doi.org/10.1007/978-1-4419-7997-1_56

[12] Franz Heinen. 1834. Ueber Systeme von Kraeften. *G. D. Baedeker* (1834).

[13] Borhan Kazimipour, Xiaodong Li, and A. K. Qin. 2014. A review of population initialization techniques for evolutionary algorithms. In *Proceedings of the 2014 IEEE Congress on Evolutionary Computation, CEC 2014*. Institute of Electrical and Electronics Engineers Inc., 2585–2592. https://doi.org/10.1109/CEC.2014.6900618

[14] Z. A. Melzak. 1961. On the Problem of Steiner. *Canad. Math. Bull.* 4 (1961), 143—-148.

[15] B. Miller and D. Goldberg. 1995. Genetic Algorithms, Tournament Selection, and the Effects of Noise. *Complex Syst.* 9 (1995).

[16] Matthias Müller-Hannemann and Siamak Tazari. 2010. A near linear time approximation scheme for Steiner tree among obstacles in the plane. *Computational Geometry: Theory and Applications* 43, 4 (may 2010), 395–409. https://doi.org/10.1016/j.comgeo.2009.01.011

[17] R. C. Prim. 1957. Shortest Connection Networks And Some Generalizations. *Bell System Technical Journal* 36, 6 (1957), 1389–1401.

[18] J. Scott Provan. 1988. An approximation scheme for finding Steiner trees with obstacles. *SIAM J. Comput.* 17, 5 (1988), 920–934. https://doi.org/10.1137/0217057

[19] Thomas Simpson. 1805. *The doctrine and application of fluxions: containing (besides what is common on the subject) a number of new improvements in the theory, and the solutions of a variety of new and very interesting problems in different branches of the mathematics*. CUNY Academic Works, London. https://academicworks.cuny.edu/cc{_}arch{_}text/1

[20] J F Weng and J Macgregor Smith. 2001. Steiner Minimal Trees with One Polygonal Obstacle 1. *Algoritmica* 29 (2001), 638–648. https://doi.org/10.1007/s00453-001-0002-1

[21] Martin Zachariasen and Pawel Winter. 1999. Obstacle-Avoiding Euclidean Steiner Trees in the Plane: An Exact Algorithm. In *Algorithm Engineering and Experimentation. ALENEX 1999. Lecture Notes in Computer Science*, Michael T. Goodrich and Catherine C. McGeoch (Eds.). Vol. 1619. Springer, Berlin, Heidelberg, 286–299. https://doi.org/10.1007/3-540-48518-X_17