Solving Euclidean Steiner Tree Problems with Multi Swarm Optimization *

Tom Decroos
KU Leuven
Dept. of Computer Science

Patrick De Causmaecker
KU Leuven
Dept. of Computer Science

Bart Demoen
KU Leuven
Dept. of Computer Science

ABSTRACT

A new iterative heuristic algorithm, based on Multi Swarm Optimization, is presented for Steiner Tree Problems (STP) in the 2-dimensional Euclidean plane. The basic algorithm is made practical for large instances by applying a result from graph theory, and a well-informed approximation. The algorithm's performance is compared to perfect solutions for the classic Steiner Tree Problem and to a deterministic heuristic for the k-bottleneck STP, a variant of STP. The algorithm often produces near optimal solutions with limited resources. The approach can be applied to higher dimensions and to other variants of the Steiner Tree Problem.

Categories and Subject Descriptors

I.2.8 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Heuristic methods*

1. INTRODUCTION

The Steiner Tree Problem is a problem in combinatorial optimization. It asks for the best possible network interconnecting n given points, also named terminal points. Extra points called *Steiner points* can be added to improve the network.

The problem can be formulated both in the plane (or in spaces with higher dimensions) and in the context of weighted graphs where we are given an edge-weighted graph $G = (V, E, w)$ and a subset $S \subseteq V$ of required vertices. STP has many variants with applications in VLSI design, network routing, wireless communication, computational biology, etc. Almost all these variants are NP-hard.

We have developed and applied a new heuristic to two specific instances of STP: (1) the classic Steiner Tree Problem in the Euclidean plane which asks for a Steiner tree of minimal total length; it is abbreviated as ESTP; (2) the k-bottleneck STP which asks for a Steiner tree that minimizes the longest edge. A maximum of k Steiner points can be added to the network to shorten the bottleneck.

We achieve near optimal results for ESTP with little computational effort. Moreover, our approach improves significantly on a recent heuristic for the k-bottleneck STP as devised by Marcus Brazil et al. [1].

We assume the reader to be familiar with Particle Swarm Optimization and (Multi) Swarm Optimization [2].

2. THE STP-MSO ALGORITHM

Our application of Multi Swarm Optimization is unconventional: we let a set of sub-swarms each search for their own optimal placement of one Steiner point. Each subswarm contributes to a global optimal placement of Steiner points by being aware of the position of the Steiner points of the other sub-swarms.

Our algorithm performs a number of iterations, either predefined or until some criterium is met. Each iteration evolves all the particles in a subswarm in the direction of its current best particle, taking into account the best particle in the other sub-swarms, and with some random factor. At each iteration, a new sub-swarm (i.e a new Steiner point) is injected in the system, possibly subject to conditions that limit the number of sub-swarms or their fitness, and with some random element in its generation. The evolution of particles is quite naturally based on their current position and velocity, as we consider the placement of Steiner points in the Euclidean plane. Sub-swarms are deleted when they do not contribute to a better current solution.

Visualization of the algorithm.

The progress of the sub-swarms during an execution of the algorithm for the two dimensional ESTP can be visualized conveniently: Figure 2 shows three snapshots from an animation generated by our implementation. This suggests the possibility of an interactive environment in which the user can manipulate Steiner points directly in coordination with the running program. The decision at which precision to stop the computation could also be made by the user.

3. IMPROVING EFFICIENCY

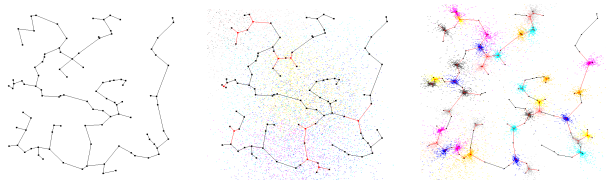
STP-MSO updates the best Steiner Tree by repeatedly comparing alternative spanning trees and selecting the best one. This is the most important step in the algorithm and also the most resource intensive. To compare Steiner trees,

*This research was performed by the first author in partial fulfilment of the requirements for the Bachelor's degree in Informatics at the KULAK, KU Leuven, under the supervision of the other two authors. This work is supported by the Belgian Science Policy Office (BELSPO) in the Interuniversity Attraction Pole COMEX. (http://comex.ulb.ac.be)

GECCO '15 July 11-15, 2015, Madrid, Spain

ACM ISBN 978-1-4503-3488-4/15/07.

DOI: http://dx.doi.org/10.1145/2739482.2764676



(a) (b) (c)

Figure 1: (a) the initial MST without any additional Steiner point; (b) 10 sub-swarms and their best representative as Steiner points: the sub-swarms are still dispersed; (c) the sub-swarms converge: the current solution is close to optimal.

	optimal solution		STP-MSO	
Problem size	length (%)	k	length (%)	k
10	95.70	4	95.70	4
100	96.76	44	96.86	41
1000	n/a	n/a	97.02	389

Table 1: Optimal solutions vs STP-MSO. *length (%)* is the total length of the Steiner tree relative to the length of the minimal spanning tree without Steiner points; k is the number of Steiner points used to construct the Steiner tree. STP-MSO was given 500 iterations and 500 particles/sub-swarm for all tests.

we first construct them as the minimal spanning tree (MST) of the set of all points (terminal points and Steiner points) using Kruskal's algorithm. This computation is incremental, as new Steiner points are injected or the current best Steiner point in a sub-swarm changes. We rely on two observations to improve the efficiency of the computation of a new minimal spanning tree: (1) a theorem stating that one can compute an MST for a graph G with an additional vertex v by considering only the edges in an already computed MST for G and the edges between v and vertices in G; (2) an approximation using the intuition that long edges from the additional vertex v to any vertex of G are improbable to contribute to the new MST. Using these observations, our algorithm performs well also on large numbers of terminals.

4. RESULTS

We used benchmarks provided by the *OR library*[1].

Benchmarks of size 10 to 100 have been solved to optimality for the classic STP by an algorithm called *geosteiner96* in [4]. Table 1 shows how well STP-MSO approximates the optimal solutions for a small subset of our experiments. [2] We have not made a thorough performance comparison, however, it is worth noting that *geosteiner96* took several minutes even for small benchmarks whereas STP-MSO produces results in a matter of seconds. STP-MSO generates solutions that are very close or equal to the optimal solution. This leads us to believe that STP-MSO can also generate close to optimal solutions for other variants of the Steiner Tree Problem and for problem instances for which the optimal solution is not yet known.

[1]http://people.brunel.ac.uk/~mastjjb/jeb/info.html

[2]Optimal solutions seem to exist for the instances with more than 100 terminal nodes, but we were unable to obtain them.

		MSTH	STP-MSO
Problem size	k	bottleneck (%)	bottleneck (%)
10	5	75.31	63.34
20	10	56.35	46.08
50	20	47.05	40.33
100	50	32.07	27.94

Table 2: MSTH vs STP-MSO. *bottleneck (%)* shows the length of the bottleneck, relative to the minimal spanning tree without Steiner points; k is the number of Steiner points. STP-MSO was given 500 iterations and 500 particles per sub-swarm.

We tested the effectiveness of STP-MSO on the k-bottleneck Steiner Tree Problem by comparing it with the Minimal Spanning Tree Heuristic (MSTH) [3]. To our knowledge, this is currently still one of the best deterministic heuristics in literature and its approximation ratio is bounded above by 2. The results can be seen in Table 2. Our tests show that STP-MSO outperforms MSTH by a significant margin: the Steiner trees constructed by STP-MSO have a bottleneck that is, on average, 11% shorter than the bottleneck of a Steiner Tree constructed by MSTH, with peaks of 17% improvement over MSTH. We would have liked to compare directly with the newer approximation algorithm devised by Marcus Brazil et al. [1], but were unable to find an implementation. [1] performs up to 8% better than MSTH, so our heuristics seem to outperform it as well.

5. CONCLUSION

Our method based on Swarm Intelligence is very effective and not complex. Indeed, unlike other heuristics, it does not rely on any mathematical properties of the problem. It is also easily adaptable to other constraints and it has a powerful visualization that can be of great help while using the algorithm interactively. Our STP-MSO produces near optimal results for the classic Steiner Tree Problem and largely outperforms a deterministic heuristic on the k-bottleneck STP. Our STP-MSO implementation in Java can be found at https://github.com/TomDecroos/stp-mso. A longer version of this paper is at https://lirias.kuleuven.be/handle/123456789/473764.

Acknowledgements.

We thank Daan Seynaeve for his valuable comments to improve this paper.

6. REFERENCES

[1] Marcus Brazil, Charl Ras, and Doreen Thomas. A new algorithm for the euclidean k-bottleneck steiner problem. In *Proceedings of the 19th International Symposium on Mathematical Theory of Networks and Systems–MTNS*, volume 5, 2010.

[2] James Kennedy. Particle swarm optimization. In *Encyclopedia of Machine Learning*, pages 760–766. Springer, 2010.

[3] L. Wang and D.-Z. Du. Approximations for a bottleneck steiner tree problem. *Algorithmica*, 32(4):554–561, 2002.

[4] P. Winter and M. Zachariasen. *Large Euclidean Steiner Minimum Trees in an Hour*. Rapport (Københavns universitet. Datalogisk institut). Datalogisk Institut, Københavns Universitet, 1996.