Using ACO metaheuristic for MWT problem

Maria Gisela Dorzán
Edilma Olinda Gagliardi
Mario Guillermo Leguizamón
Facultad de Ciencias Físico Matemáticas y Naturales
Universidad Nacional de San Luis
San Luis, Argentina
{*mgdorzan, oli, legui*}*@unsl.edu.ar*

Gregorio Hernández Peñalver
Universidad Politécnica de Madrid
Facultad de Informática
Madrid, España
gregorio@fi.upm.es

***Abstract*—Globally optimal triangulations are difficult to be found by deterministic methods as, for most type of criteria, no polynomial algorithm is known. In this work, we consider the Minimum Weight Triangulation (MWT) problem of a given set of n points in the plane. This paper shows how the Ant Colony Optimization (ACO) metaheuristic can be used to find high quality triangulations. For the experimental study we have created a set of instances for MWT problem since no reference to benchmarks for these problems were found in the literature. Through the experimental evaluation, we assess the applicability of the ACO metaheuristic for MWT problem.**

***Keywords*-Triangulation, Minimum Weight, Computational Geometry, ACO Metaheuristic.**

I. Introduction

In Computational Geometry there are many optimization problems that either are NP-hard or no polynomial algorithms are known to solve them. Examples of these optimization problems are those related to special geometric configurations, such as *triangulations*, and are interesting to research due to their use in many fields of application. Triangulations are planar partitions, which received considerable attention mainly due to their applications, e.g. computer graphics, scientific visualization, robotics, computer vision, and image synthesis, as well as in mathematical and natural science.

Minimum Weight Triangulation problem (MWT) minimizes the sum of the edge lengths, providing a quality measure for determining how good is a structure. The complexity of computing a minimum weight triangulation has been one of the most longstanding open problems in Computational Geometry, introduced by Garey and Johnson [11] in their open problems list, and various approximation algorithms were proposed over time. Mulzer and Rote [19] recently showed that MWT problem is NP-hard. Considering the inherent difficulty of this problem, the approximate algorithms arise as alternative candidates. These algorithms can obtain approximate solutions to the optimal ones, and they can be specific for a particular problem or they can be part of a general applicable strategy in the resolution of different problems. A metaheuristic methods satisfy these properties. This kind of technique is an iterative generation process that guides the search of solutions intelligently combining different concepts of diverse fields as artificial intelligence [20], biological evolution [2], swarm intelligence [14], among others. These algorithms have a simple implementation and they can efficiently find good solutions for NP-hard optimization problems [18]. In this work we present the *Ant Colony Optimization* (ACO) metaheuristic. The family of algorithms derived from the ACO metaheuristic embodies a set of simple agents that compose a complex system capable of timely building solutions of high quality. The agents obey simple rules and act independently. However, they cooperate sporadically in a indirect form to conform a distributed process in which all the agents work to carry out a common aim.

According to the current state-of-the-art with respect to theoretical results about the MWT problem, we adopted to solve them using a metaheuristic technique as the more appropriate approach to find near optimal solutions. Previous works about approximations on MWT problem using metaheuristic, were presented in [6] and [9], where we described the design of the ACO algorithm and gave the first steps in this research. To the best knowledge of the authors, there are no reports in literature of extensive experimental evaluations using metaheuristics techniques. More precisely, there are some limited experiments using metaheuristic techniques which do not represent a real challenge [3][16][21][22][24].

This paper is organized as follows. In the next two sections, the theoretical aspects of MWT problem is presented. In Section III, we present the general overview of the ACO metaheuristic and the proposed ACO algorithm for the MWT problem, namely ACO-MWT. In Section IV, we describe the MWT problem instance used, and the details and results of the experimental study in which we analyze the sensitivity of some important parameters on the performance of the proposed ACO algorithm. Finally, in Section V, we show the statical analysis for observing the behavior of the ACO-MWT algorithm and we show the runtimes of the compared algorithms. Last section is reserved for the conclusions and future vision.

II. Minimum Weight Triangulation Problem

Let *S* be a set of points in the plane. A triangulation of *S* is a partition of the convex hull of *S* into triangles whose set of vertices is exactly *S*. The weight of a triangulation *T* is the sum of the Euclidean lengths of all the edges of *T*. The triangulation that minimizes this sum is named a *Minimum Weight Triangulation* of *S* and it is denoted by $MWT(S)$.

MWT problem has a long and rich history, dating back to the 1970s. As far as the knowledge of the authors, the MWT problem was first considered by Düppe and Gottschalk [10] who proposed a greedy algorithm which always adds the shortest edge to the triangulation. Later, Shamos and Hoey [23] suggested using the Delaunay triangulation as a minimum weight triangulation. Lloyd [17] provided examples which show that both proposed algorithms usually do not compute the MWT. Similarly, Gilbert [12] and Klincsek [15] independently showed how to compute a minimum weight triangulation of a simple polygon in $O(n^3)$ time by dynamic programming.

From the point of view of metaheuristics, many papers present solutions to problems in the field of Graphical Computation. In 1992, Sen and Zheng [22] proposed an algorithm to approximate the minimum weight triangulation using Simulated Annealing but in many proofs they obtain solutions "near" to the ideal ones. In 1993, Wu and Wainwright [24] approximated the minimum weight triangulation using a genetic algorithm where the recombination and mutation operators are the same, such that both of them make a flip to obtain the neighbors. Qin et al. [21] also use a genetic algorithm and they proposed new operators for recombination and mutation. Capp and Julstrom [3] present a new weight codification of the triangulations to use it in a genetic algorithm. In 2001, Kolingerova and Ferko [16] presented a genetic optimization, which recombination operator is named DeWall and the mutation operator makes a flip in the selected individual. In the previous mentioned works, the experimental evaluation is rather poor and they do not describe the quality of the obtained solutions.

The complexity of the computation was one of the more interesting opened problems in Geometry Computacional until Mulzer and Rote demonstrated in 2006 that MWT is a NP-hard problem [19].

III. Ant Colony Optimization Metaheuristic - ACO

Ant Colony Optimization [5] is a metaheuristic approach for solving hard combinatorial optimization problems. ACO is based on the indirect communication of a colony of simple agents, called ants, mediated by pheromone trails. The pheromone trails in ACO serve as a distributed, numerical information which the ants use to probabilistically construct solutions to the problem being solved and which the ants adapt during the algorithm's execution to reflect their search experience.

ACO is an iterative distributed algorithm where each ant builds a solution by walking from vertex to vertex on the construction graph with the constraint of not visiting any vertex that it has already visited in her walk. At each construction step of a solution, an ant selects the next vertex to be visited according to a stochastic mechanism that is biased by the pheromone: at vertex i, the next vertex is selected stochastically among the previously unvisited ones. In particular, if j has not been previously visited, it can be selected with a probability that is proportional to the pheromone associated with edge (i, j). At the end of an iteration, considering the solution quality, the pheromone values are modified in order to bias ants in future iterations for constructing solutions similar to the best ones. The probabilistic transition rule is based on two parameters, called pheromone trails and heuristic information. The pheromone trail, denoted by τ_{ij}, encodes a long-term memory about the entire ant search process, and is updated by the ants themselves. The heuristic information, denoted by η_{ij}, represents *a priori* information about the problem instance information provided by a source different from the ants. In many cases η is the cost, or an estimate of the cost, of adding the component or connection to the solution under construction.

Furthermore, an ACO algorithm includes two adicional mechanisms: trail evaporation and, optionally, daemon actions. Trail evaporation decreases all trail values over time, in order to avoid unlimited accumulation of trails over some component. Daemon actions can be used to implement centralized actions which cannot be performed by single ants, such as the invocation of a local optimization procedure, or the update of global information to be used to decide whether to bias the search process from a non-local perspective.

In the following, the ACO-MWT algorithm and a description of its main components are presented.

Algorithm ACO-MWT

Initialize()
for $c \in \{1, \ldots, C\}$ **do**
  for $k \in \{1, \ldots, K\}$ **do**
    /*Begin of BuildSolutionk process*/
    $S_k \leftarrow \emptyset$
    $i \leftarrow$ *SelectInitialPoint*(S)
    while S is not triangulated **do**
      $\mathrm{F}_i \leftarrow$ *FeasiblePoints*$(i,\ S_k)$
      if $\mathrm{F}_i = \emptyset$ **then**
        $i \leftarrow$ *SelectPoint*$(S,\ S_k)$
        $\mathrm{F}_i \leftarrow$ *FeasiblePoints*$(i,\ S_k)$
      end if
      $j \leftarrow$ *SelectPointProb*(F_i)
      if not *IntersectSolution*(i, j, S_k) **then**
        $S_k \leftarrow S_k \cup (i, j)$
        $i \leftarrow j$

```
    end if
    UpdateFeasiblePoints(i, j)
  end while
  /*End of BuildSolutionk process*/
  EvaluateSolution()
 end for
 SaveBestSolutionSoFar()
 UpdateTrails()
end for
ReturnBestSolution()
```

Main components:

Initialize(): τ_0, initial trail of pheromone associated to each edge; *K*, quantity of ants of the colony; α and β, proportion in which they will affect the pheromone trails and heuristic information in the probabilistic transition rule; and *C*, quantity of cycles.

BuildSolutionk: each ant builds a triangulation of a given set or instance S, starting from an initial random point. At each step, the algorithm adds a new edge (i, j) if there is no intersection between (i, j) and the edges of the partial solution S_k. In this case, i is a feasible point for j and vice versa.

In this work, probabilistic transition rule is based on the following probabilistic model:

$$P_{ij} = \begin{cases} \frac{\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}}{\sum_{h \in F(i)} \tau_{ih}^{\alpha}.\eta_{ih}^{\beta}}, & j \in F(i); \\ 0, & \text{otherwise.} \end{cases} \quad (1)$$

- $F(i)$ is the set of feasible points for point i.
- τ_{ij} is the pheromone value associated to edge (i, j).
- η_{ij} is the heuristic value associated to edge (i, j).
- α and β are positives values for determining the relative importance of the pheromone with respect to the heuristic information.

If the current point has no feasible points, it selects the next reference point according to one of the following criteria: i) random selection; ii) select the point with the largest quantity of feasible points; or, iii) select the point with the lowest quantity of feasible points.

SelectInitialPoint(S): returns a point $p \in S$, randomly selected.

FeasiblePoints(i, S_k): returns a set P of points ($P \subseteq S$), such that the edge (i, p), with $p \in P$, could not intersect with the edges of the solution S_k. Note that this function may return points that are no feasible for *p*.

SelectPoint(S, S_k): returns a point $p \in S$, such that $FeasiblePoints(p, S_k)$ has at least a point. p is selected according to one of the criteria mentioned previously.

SelectPointProb(F_i): returns a point $j \in F_i$ chosen according to Equation 1, where η_{ij} is $1/d_{ij}$, and d_{ij} is the Euclidean distance between i and j.

IntersectSolution(i, j, S_k): returns true if the edge (i, j) intersects at least one edge of the solution S_k; returns false, otherwise.

UpdateFeasiblePoints(i, j): updates the feasible points of i and j, i.e., the points i and j are not more feasible with respect to each other.

UpdateTrails(): increases the pheromone level in the promising paths, and is decreased otherwise. The following equation is used:

$$\tau_{ij} = (1 - \rho)\tau_{ij} + \Delta\tau_{ij} \quad (2)$$

- $\rho \in (0, 1]$ is the factor of persistence of the trail.
- $\Delta\tau_{ij} = \sum_{k=1}^{K} \Delta^k\tau_{ij}$ is the accumulation of trail, proportional to the quality of the solutions.
- $\Delta^k\tau_{ij} = \begin{cases} Q/L_k, & \text{when ant } k \text{ used edge } (i, j); \\ 0, & \text{in other case.} \end{cases}$
- Q is a constant depending of the problem; it usually set to 1.
- L_k is the weight of the triangulation *k*.

In this work, the pheromone trail update can be done according to one of the following criteria: *elitist* and *not elitist*. With the *elitist* criterion, the best found solution is used to give an additional reinforcement to the levels of pheromone. The *not elitist* one uses the solutions found by all the ants to give an additional reinforcement to the levels of pheromone.

IV. Experimental Evaluation

In this work we present an ACO algorithm for the MWT problem which is represented by an Ant System (AS), a particular instance of the class of ACO algorithms. We try to find an acceptable combination of parameter values for the ACO-MWT algorithm in order to obtain triangulations of small weight, closer to the minimum.

To the best knowledge of the authors, there not exist collections of instances in the literature for MWT problem. Consequently, no benchmarking data are publicly available that allow us to compare our proposal with some other algorithm previously studied. According to that, we design an *instance generator* and we have generated respectively a collection of 10 instances of size 40/80/120/160/200; i.e., a total of 50 problem instances. Each instance is called LDn-i where n denotes the size of the i-instance, with $1 \leq i \leq 10$. The instance generator uses different functions of CGAL Library [1]. The points are randomly generated, uniformly distributed and for each point (x, y), the coordinates $x, y \in [0, 1000]$. For implementation purposes, we assume that there are non collinear points.

The ACO-MWT algorithm was implemented in C language and run on BACO parallel cluster, under CONDOR batch queuing system. It must be remarked that the current experimental study is devoted to analyze the performance of

the algorithms with respect the quality of the solutions found considering different combination of parameter values.

We used the following parameter values: $\alpha = 1$; $\beta = 1$ and 5; and $\rho = 0.10$, 0.25, and 0.50. *elit* = 1 and 0, where 1 means that the trail is updated with the *elitist* criterion; in other case, the updating is done with the *not elitist* criterion. *criterion* = 1, 2, and 3, is used for selecting a point in the *SelectPoint*(S, S_k) procedure in ACO-MWT algorithm. For $criterion = 1$ the point is chosen randomly; for $criterion = 2$, the chosen point has the largest quantity of feasible points; and for $criterion = 3$, the chosen point has the lowest quantity of feasible points. $C = 1000$. $K = 50$.

For each parameter setting given below, 30 runs were performed by using different random seeds. For each problem instance, the experiment were done with the twelve parameter settings, according to combinations of parameters before detailed. We obtain average, median, best, and standard deviation values, considering the objective function (*weight*). We show the results according to the four best parameter settings considering the smaller weights, i.e., the four *Best* values. We only show the results for the best four parameter settings since the results for the remaining ones were in general of lower quality with respect to the best found values.

Next, we show the experimental results, considering the above presented settings. Each parameter setting is denoted by ($instance$-β-ρ-$elit$). α is not shown because it is the same for all the cases ($\alpha = 1$). We show the results for $criterion = 1$ because in the most of the instances (more than 80%) we obtained better results choosing the next reference point in a random way. The decimal numbers are not showed because they are not significant.

In this work, we analyze the performance of the ACO-MWT algorithm over four instances of 40, 80, 120, 160, and 200 points. In Tables V, VI, VII, VIII, and IX (see Appendix) we show the results according to the four best parameter settings with respect to the smaller weights (*Best* values). The Table I is a summary of the previous tables and shows that the best weights are obtained using configurations with $\beta = 5$, $elit = 1$, and ρ between 0.1 and 0.5, i.e., we obtained better results giving more relevance to the heuristic information and updating the trails with the elitist criterion.

Table I: ACO-MWT: Summary of results for four instances of 40, 80, 120, 160, and 200 points with respect to the best values.

β	ρ	$elit$
1 (29.4%)	0.10 (31.7%)	0 (4.7%)
5 (70.6%)	0.25 (36.6%)	1 (95.3%)
	0.50 (31.7%)	

We compare ACO-MWT algorithm among the Delaunay Triangulation (DT). The Table II shows the best and median values for ACO-MWT algorithm and the weight of DT. In addition, the fifth column shows the percentage differences between the weight of DT and the best weight found with ACO-MWT algorithm. In the displayed results it can seen that the ACO-MWT algorithm found the smaller weights for all cases. ACO-MWT algorithm managed to reduce (as seen in column "diff.%") the weights between 0.5% and 5% with regard to the DT strategy, but for LD40-4 instance achieved a reduction larger than 8%.

Table II: ACO-MWT: Comparing results between ACO-MWT and DT.

Instance	*ACO-MWT* Best values	*ACO-MWT* Median values	*DT*	*diff.%*
LD40-1	5493047	5502010	5666348	-3,06
LD40-2	4661242	4664817	4722381	-1,29
LD40-3	5502567	5519777	5663032	-2,83
LD40-4	5745772	5750729	6289829	-8,65
LD80-1	6242505	6273781	6462038	-3,40
LD80-2	7605383	7638408	8081573	-5,89
LD80-3	5836037	5866309	6143637	-5,01
LD80-4	6217040	6283664	6460311	-3,77
LD120-1	9325984	9361368	9581142	-2,66
LD120-2	5962099	6020394	6149825	-3,05
LD120-3	8632306	8661549	8948084	-3,53
LD120-4	7762612	7804348	8111182	-4,30
LD160-1	7489134	7515852	7837804	-4,45
LD160-2	7057185	7089629	7144975	-1,23
LD160-3	8748156	8803769	8891459	-1,61
LD160-4	6184695	6225358	6315497	-2,07
LD200-1	6629218	6684062	6990407	-5,17
LD200-2	7657541	7704917	8140057	-5,93
LD200-3	8804317	8845121	9358523	-5,92
LD200-4	7971440	8019717	8015613	-0,55

V. Statistical Analysis

Considering the parameter values mentioned above, we obtain 12 parameter settings. These settings are labeled and listed in Table III. The parameter settings were obtained empirically after performing several experiments.

We show a statistical analysis to establish the parameter effect on the reliability of the ACO method.

We performed the Kolmogorov-Smirnov test to determine if these samples are normally distributed, which resulted that the samples have not the normal distribution.

As the values do not not follow a normal distribution we apply the Kruskal-Wallis test (a nonparametric statistical test) to perform the median comparison in order to determine the sensitivity of the parameters, using the parameter settings given in Table III. The null hypothesis considered is: there is not a significative difference among the found results and if there are differences, they are due to random effects. Then we apply the Tukey test in order to determine the experimental conditions where exist significative differences.

Table III: Parameter settings and their identifiers (ID).

ID	β	ρ	$elit$
1	1	0.10	0
2	1	0.25	0
3	1	0.50	0
4	5	0.10	0
5	5	0.25	0
6	5	0.50	0
7	1	0.10	1
8	1	0.25	1
9	1	0.50	1
10	5	0.10	1
11	5	0.25	1
12	5	0.50	1

We also use the boxplot method to show the distribution of weights for each environment.

We present an statical analysis of the ACO-MWT algorithm over five groups of four instances of size 40, 80, 120, 160, and 200 respectively. Figures 1, 2, 3, 4, and 5 show the results obtained by the Tukey method (see Appendix). The $y\text{-}axis$ represents the identifier (ID) for each parameter setting shown in Table III. We can infer that: the algorithm is sensitive to the $elit$ parameter because there are significative difference in the results when we change its value; fixed the α, β, and $elit$ parameters the algorithm is not sensitive to the ρ parameter because there are not significative difference between the results; if $elit = 0$, then β has influence to the results, but β has no influence otherwise.

Figures 6, 7, 8, 9, and 10 show the boxplots of the weights obtained for the 30 seeds for four instances for 40, 80, 120, 160, and 200 points for the twelve configurations (see Appendix). The $x\text{-}axis$ represents the identifier (ID) for each parameter setting shown in Table III and the $y\text{-}axis$ represents the weight. The medians are similar for ρ between 0.10, 0.25, and 0.50. The algorithm is more robust when $elit = 1$ because the 50% of the values (values between the first and third quartile) are very closed around the median value. We obtained better results with $\beta = 5$ and $elit = 1$ (see parameter settings 7 to 12).

A. Analysis of runtimes

In order to compare the computational effort of the treated algorithms we compare the runtimes of both algorithms (ACO-MWT and DT). It should be remarked that our proposed algorithm is based on a metaheuristic technique, which is an iterative and stochastic population-based process. Therefore, it is very different from the DT algorithm because it is a deterministic algorithm that builds a single solution in $O(nlogn)$ time. So, this leads to large differences when comparing the runtimes of both algorithms. This difference in the use of computational resources is despicable because ACO-MWT algorithm gets higher quality solutions. In Computational Geometry, high quality solutions are very important in certain applications in relation to the MWT problem [13] [25] [17] [4]. Consequently, more complex and time-consumer algorithms need to be used to reach solutions of higher quality as usually expected. Nevertheless, we are aware that for some Computational Geometry applications the Delaunay Triangulation could be a simple and direct alternative when solutions of medium or low quality are acceptable.

In Table IV we show the execution times of ACO-MWT algorithm considering the parameter setting that yields the best results. The α, β and ρ parameters do not substantially affect the runtime because they only modify the probability distribution and the extent of amount of pheromone laid on the edges. Therefore, as the consumed time does not significantly vary for different values of α, β, and ρ we particularly show the runtimes for $\alpha = 1$, $\beta = 1$, and $\rho = 0.1$ using one instance of each one of the three different problem sizes studied here. We show the execution time for $elit = 1$ because we obtain better results following an elitist criterion.

Table IV: MWT: Average runtimes for ACO-MWT algorithm (for 30 seeds) and the runtime for Delaunay Triangulation (in milliseconds).

# points	*ACO-MWT*	*DT*
40	417463	13
80	2672815	18
120	7416141	29
160	13729977	31
200	24944603	54

As previously analyzed and expected, Table IV shows the runtimes of the algorithms applied. It can clearly be observed the increment of the computational cost of the ACO-MWT algorithm with respect to DT. These results show that solutions of higher quality can be found by applying a metaheuristic technique but with a higher cost.

VI. Conclusion

In this work, we present the design of approximation algorithm for solving the Minimum Weight Triangulation problem for sets of points in the plane. The proposed ACO algorithm is respectively represented by an Ant System (AS), a particular instance of the class of ACO algorithms.

We also detailed the generation of instances for the experimental evaluation, being this another contribution of this paper, since there are not available instances with special properties for building triangulations. At the moment, we have a large collection of instances for future experimentations, where the obtained results could be used as benchmarks.

Future work will address us to go further in our research in order to design either an improved version of the ACO algorithm studied here and also, to propose alternative

metaheuristics that reduce the computation time required to reach high quality solutions. In addition, we aim at comparing the proposed algorithms against other metaheuristics. Currently we are conducting an experimental study involving Simulated Annealing [7] [8] which will help us to compare the performance of both techniques in regards of quality of solutions and runtimes.

ACKNOWLEDGMENT

The authors would like to thank to CONICET *Consejo Nacional de Investigaciones Científicas y Técnicas*; Research Project *Tecnologías Avanzadas de Bases de Datos* (22/F014) financed by Universidad Nacional de San Luis, Argentina; Research Project MTM2008-05043 financed by *Ministerio de Ciencia e Innovación de España* and, Instituto de Física Aplicada (INFAP) UNSL-CONICET, Argentina, to allow us to use the cluster. Likewise, to the colleagues who contributed with opinions and technical advice.

REFERENCES

[1] Cgal, computational geometry algorithms library.

[2] T. Bäck, D. Fogel, and Z. Michalewicz. *Handbook of evolutionary computation*. Oxford Univ. Press, 1997.

[3] K. Capp and B. Julstrom. A weight-coded genetic algorithm for the minimum weight triangulation problem. In *SAC*, pages 327–331, 1998.

[4] J. Davis and M. McCullagh. Display and analysis of spatial data, wiley, 1975.

[5] M. Dorigo and T. Sttzle. *Ant Colony Optimization*. MIT Press, Cambridge, MA, 2004.

[6] M. Dorzán, E. Gagliardi, Leguizamón, and G. M., Hernández Peñalver. Algoritmo aco aplicado a la obtención aproximada de triangulaciones de peso mínimo. In *XXXV Conferencia Latinoamericana de Informática*, 2009.

[7] M. Dorzán, E. Gagliardi, M. Leguizamón, and G. Hernández Peñalver. Simulated annealing aplicado a triangulaciones y pseudotriangulaciones de peso mínimo. In *XVI Congreso Argentino de Ciencias de la Computación*, 2010.

[8] M. Dorzán, E. Gagliardi, M. Leguizamón, and G. Hernández Peñalver. Triangulaciones y pseudotriangulaciones de peso mínimo: resolución aproximada con simulated annealing. In *VII Jornadas de Matemática Discreta y Algorítmica. España*, 2010.

[9] M. Dorzán, E. Gagliardi, M. Leguizamón, M. Taranilla, and G. Hernández Peñalver. Algoritmos aco aplicados a problemas geométricos de optimización. In *XIII Encuentros de Geometría Computacional*, 2009.

[10] R. Düppe and H. Gottschalk. Automatische interpolation von isolinien bei willkürlichen stützpunkten. *Allgemeine Vermessungsnachrichten*, 77:423–426, 1970.

[11] M. Garey and D. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[12] P. Gilbert. New results in planar triangulations. Technical Report R–850, Univ. Illinois Coordinated Science Lab., 1979.

[13] L. Gray, L. Martha, and A. IngraLea. Hypersingular intergrals in boundary element fracture analysis, 1990.

[14] J. Kennedy and R. Eberhart. *Swarm Intelligence (The Morgan Kaufmann Series in Artificial Intelligence)*. Morgan Kaufmann, 1st edition, April 2001.

[15] G. Klincsek. Minimal triangulations of polygonal domains. *Ann. Disc. Math.*, 9:121–123, 1980.

[16] I. Kolingerová and A. Ferko. Multicriteria-optimized triangulations. *The Visual Computer*, 17(6):380–395, 2001.

[17] E. Lloyd. On triangulations of a set of points in the plane. In *"Proc. 18th IEEE Symp. Found. Comp. Sci.*, pages 228–240, 1977.

[18] Z. Michalewicz and D. Fogel. *How to Solve It: Modern Heuristics*. Springer, 2004.

[19] W. Mulzer and G. Rote. Minimum weight triangulation is np-hard. In *In Proc. 22nd Annu. ACM Sympos. Comput. Geom*, pages 1–10. ACM Press, 2006.

[20] I. Osman and J. Kelly. *Meta-heuristics: Theory and Applications*. Kluwer academic publishers, 1996.

[21] K. Qin, W. Wang, and M. Gong. A genetic algorithm for the minimum weight triangulation. In *In Proceedings of 1997 IEEE International Conference on Evolutionary Computation*, 1997.

[22] S. Sen and S. Zheng. Near-optimal triangulation of a point set by simulated annealing. In *SAC '92: Proceedings of the 1992 ACM/SIGAPP symposium on Applied computing*, pages 1000–1008, New York, NY, USA, 1992. ACM.

[23] M. Shamos and D. Hoey. Closest-point problems. In *Proc. 16th IEEE Symp. Foundations of Comp. Science*, pages 151–162, 1975.

[24] Y. Wu and R. Wainwright. Near-optimal triangulation of a point set using genetic algorithms. In *Proceedings of the Seventh Oklahoma Conference on AI.*, 1993.

[25] P. Yoeli. Compilation of data for computer-assisted relief cartography, 1975.

APPENDIX

Table V: MWT: Results for four instances of 40 points.

Par. Setting	Average	Median	Best	Std. Dev.
LD401-1-0.25-1	5497920	5499201	5493047	3093
LD401-1-0.50-1	5500288	5501497	5493047	4543
LD401-5-0.10-1	5501427	5502009	5493047	4890
LD401-5-0.25-1	5502441	5502009	5493047	4547
LD401-5-0.50-1	5500754	5501988	5493047	5518
LD402-1-0.25-1	4666083	4666261	4661242	2511
LD402-1-0.10-1	4665869	4665657	4660495	2830
LD402-5-0.50-1	4664708	4664817	4659553	2927
LD402-1-0.50-1	4666420	4666984	4659553	3191
LD402-5-0.25-1	4665475	4664817	4659553	3693
LD402-5-0.10-1	4665988	4665789	4659553	3812
LD403-5-0.25-1	5519150	5519777	5502567	6516
LD403-1-0.10-1	5520802	5521320	5503301	6966
LD403-5-0.10-1	5519544	5519625	5510241	5353
LD403-5-0.50-1	5517745	5519181	5510241	5657
LD404-1-0.25-1	5748259	5747745	5745772	2316
LD404-1-0.50-1	5748852	5748473	5745772	1946
LD404-5-0.50-1	5751695	5750729	5745772	4002
LD404-1-0.10-1	5749372	5748757	5747725	1950
LD404-5-0.10-1	5751877	5750729	5747725	3170
LD404-5-0.25-1	5751976	5750729	5747725	5157

Table VI: MWT: Results for four instances of 80 points.

Par. Setting	Average	Median	Best	Std. Dev.
LD801-5-0.50-1	6271586	6273781	6242505	14337
LD801-5-0.25-1	6271507	6275369	6249124	13911
LD801-1-0.25-1	6287660	6289344	6256190	14223
LD801-5-0.25-0	6312084	6313977	6257491	15609
LD802-5-0.25-1	7640159	7643473	7605383	13945
LD802-5-0.50-1	7637904	7638408	7607462	15751
LD802-5-0.10-1	7640725	7642497	7610007	16196
LD802-1-0.10-1	7648258	7645077	7611405	22608
LD803-5-0.10-1	5863919	5865538	5836037	13482
LD803-5-0.50-1	5867149	5866309	5843634	14250
LD803-1-0.50-1	5880349	5884690	5845840	15361
LD803-1-0.25-1	5879002	5882230	5848638	16061
LD804-5-0.50-1	6277069	6283664	6217040	23328
LD804-1-0.50-1	6273397	6271736	6221908	23681
LD804-1-0.10-1	6274697	6275648	6225424	23752
LD804-1-0.25-1	6268067	6270581	6228084	17899

Table VII: MWT: Results for four instances of 120 points.

Par. Setting	Average	Median	Best	Std. Dev.
LD1201-5-0.25-1	9361401	9361368	9325984	18424
LD1201-5-0.50-1	9364442	9361221	9331139	22122
LD1201-5-0.10-1	9366316	9361576	9333488	20569
LD1201-1-0.50-1	9393130	9398060	9345181	22710
LD1202-5-0.10-1	6019316	6020394	5962099	23256
LD1202-5-0.25-1	6022598	6027282	5979832	20284
LD1202-5-0.50-1	6026150	6030549	5995484	21177
LD1202-1-0.10-1	6052288	6059251	5996347	25249
LD1203-5-0.10-1	8661456	8661549	8632306	16552
LD1203-5-0.25-1	8658617	8659753	8632574	11813
LD1203-5-0.50-1	8663020	8668620	8633526	17104
LD1203-1-0.10-1	8704510	8706670	8658672	21258
LD1204-5-0.50-1	7802093	7804348	7762612	18435
LD1204-5-0.10-1	7797742	7798414	7766328	14877
LD1204-1-0.10-1	7831163	7832325	7774480	23019
LD1204-5-0.25-1	7798003	7794526	7776160	13279

Table VIII: MWT: Results for four instances of 160 points.

Par. Setting	Average	Median	Best	Std. Dev.
LD1601-5-0.10-1	7515805	7515852	7489134	16875
LD1601-5-0.50-1	7527522	7527589	7495482	17124
LD1601-5-0.25-1	7518991	7516890	7496947	15200
LD1601-1-0.50-1	7588850	7588794	7510242	24740
LD1602-5-0.50-1	7091344	7089629	7057185	18160
LD1602-5-0.25-0	7132276	7136565	7057845	18387
LD1602-5-0.25-1	7095758	7095337	7066699	17190
LD1602-5-0.10-1	7097824	7098349	7073680	13596
LD1603-5-0.50-1	8797618	8803769	8748156	19596
LD1603-5-0.10-1	8797039	8796674	8750828	21039
LD1603-5-0.25-1	8809061	8810073	8770606	17587
LD1603-1-0.10-1	8844513	8843003	8809559	24466
LD1604-5-0.50-1	6223382	6225358	6184695	19044
LD1604-5-0.10-1	6223522	6224625	6194364	15100
LD1604-5-0.25-1	6225932	6225323	6197952	16147
LD1604-1-0.25-1	6261122	6263084	6216992	20906

Table IX: MWT: Results for four instances of 200 points.

Par. Setting	Average	Median	Best	Std. Dev.
LD2001-5-0.10-1	6682331	6684062	6629218	25296
LD2001-5-0.25-1	6685181	6687311	6630057	26345
LD2001-5-0.50-1	6696450	6700165	6653791	29302
LD2001-5-0.25-0	6750588	6756529	6678866	20179
LD2002-5-0.10-1	7700693	7704917	7657541	21096
LD2002-5-0.25-1	7710332	7709907	7661699	20577
LD2002-5-0.50-1	7703475	7703086	7668286	15855
LD2002-5-0.25-0	7751571	7753143	7723551	15152
LD2003-5-0.10-1	8842181	8845121	8804317	22157
LD2003-5-0.25-1	8855700	8859233	8810797	20206
LD2003-5-0.50-1	8855016	8858442	8815348	22871
LD2003-1-0.25-1	8928666	8927215	8829864	32197
LD2004-5-0.50-1	8020927	8019717	7971440	26202
LD2004-5-0.25-1	8026796	8029992	7973472	22964
LD2004-5-0.10-1	8012032	8014495	7978078	21018
LD2004-1-0.25-1	8078588	8082376	8017105	29478

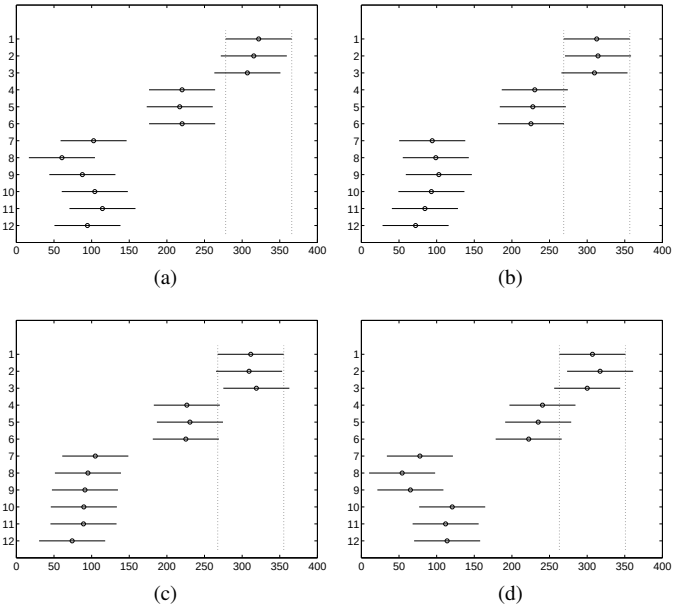


Figure 1: MWT: Multi-comparison Tukey test: (a) LD40-1, (b) LD40-2, (c) LD40-3, and (d) LD40-4.

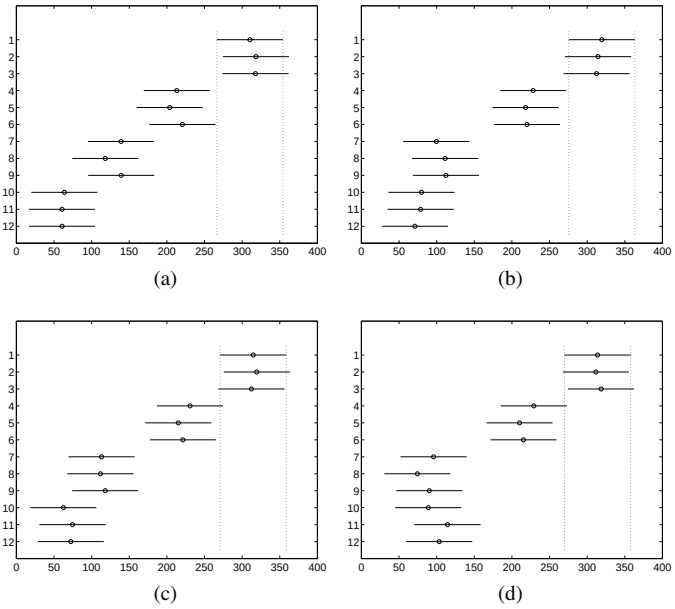


Figure 2: MWT: Multi-comparison Tukey test: (a) LD80-1, (b) LD80-2, (c) LD80-3, and (d) LD80-4.

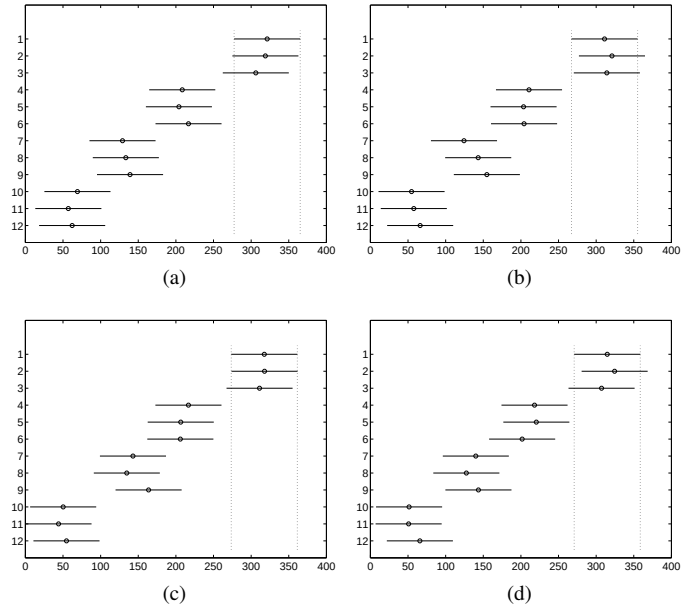


Figure 3: MWT: Multi-comparison Tukey test: (a) LD120-1, (b) LD120-2, (c) LD120-3, and (d) LD120-4.

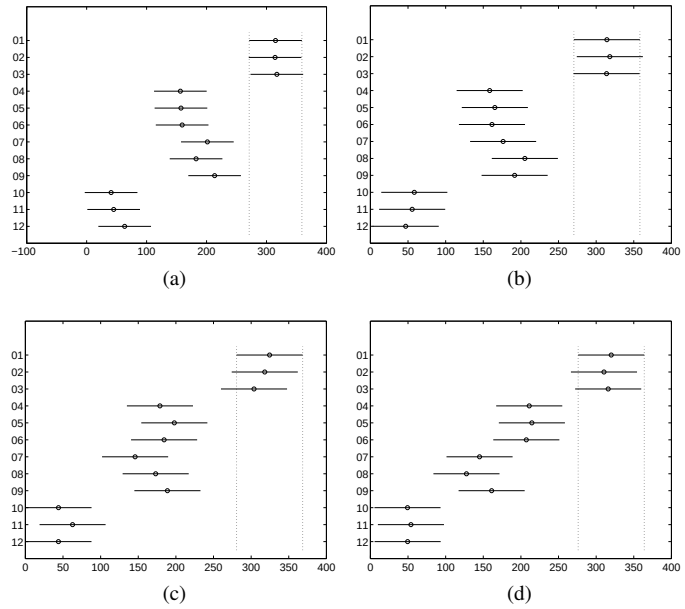


Figure 4: MWT: Multi-comparison Tukey test: (a) LD160-1, (b) LD160-2, (c) LD160-3, and (d) LD160-4.

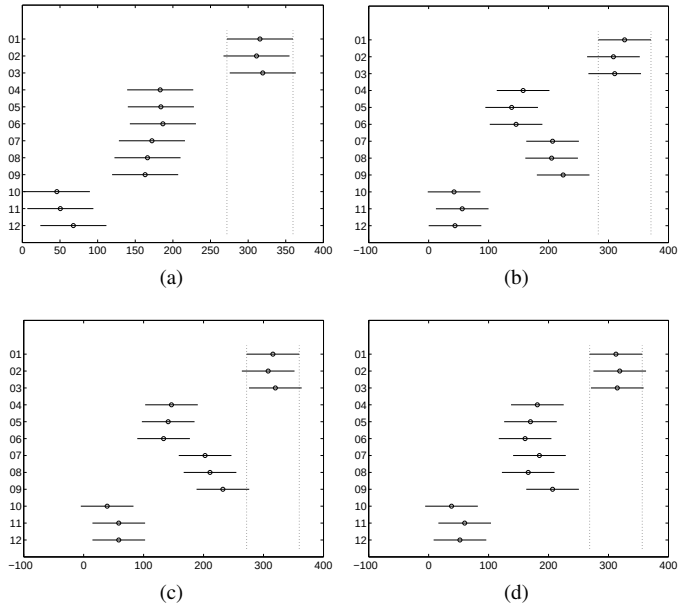


Figure 5: MWT: Multi-comparison Tukey test: (a) LD200-1, (b) LD200-2, (c) LD200-3, and (d) LD200-4.

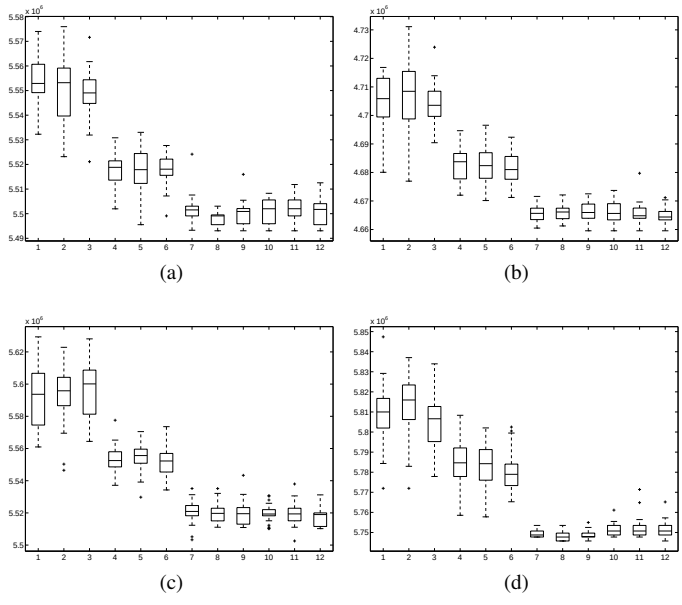


Figure 6: MWT: Boxplots for (a) LD40-1, (b) LD40-2, (c) LD40-3, and (d) LD40-4.

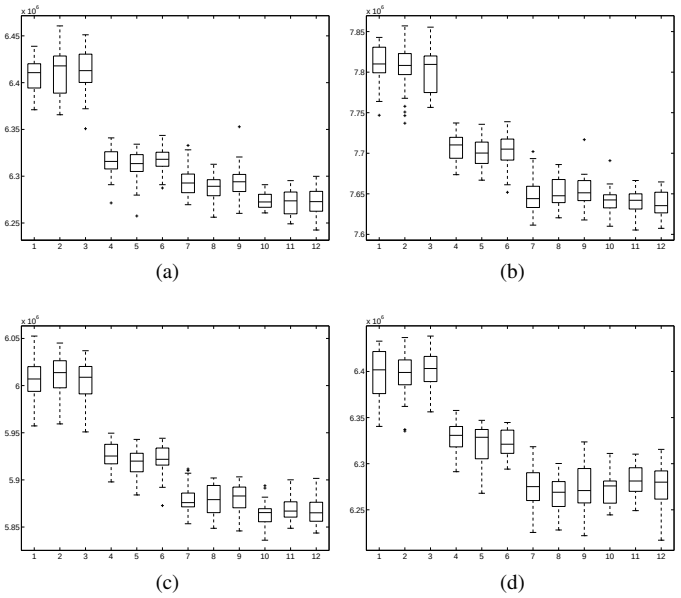


Figure 7: MWT: Boxplots for (a) LD80-1, (b) LD80-2, (c) LD80-3, and (d) LD80-4.

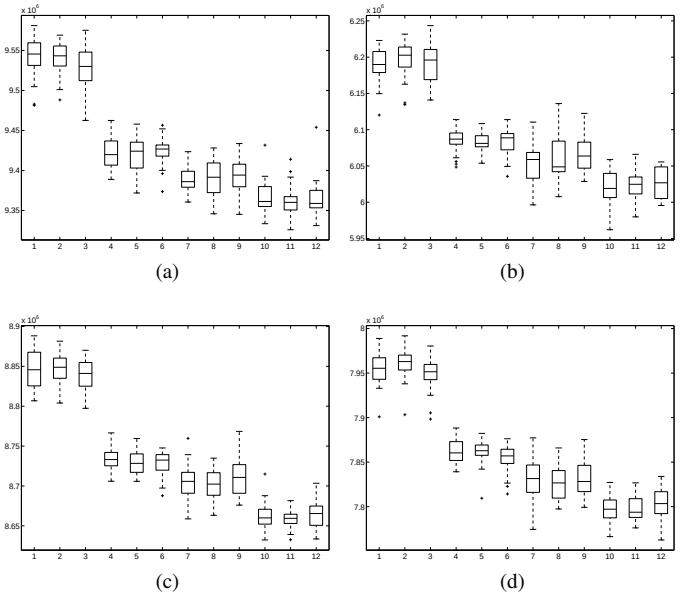
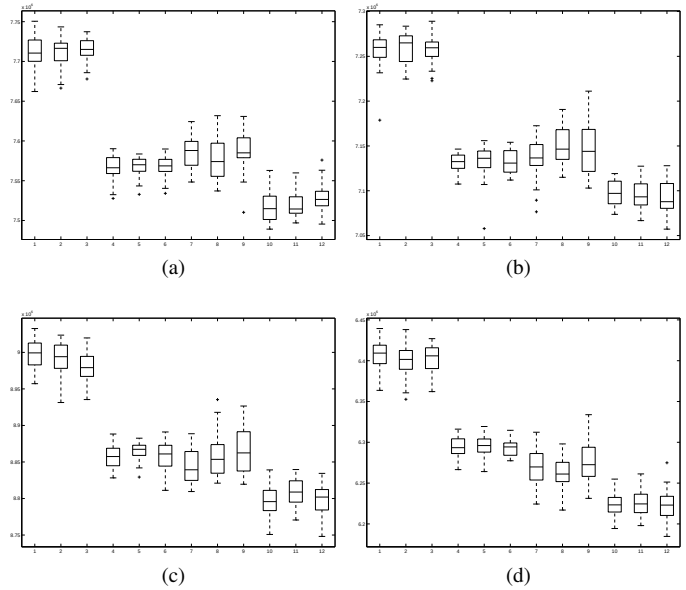
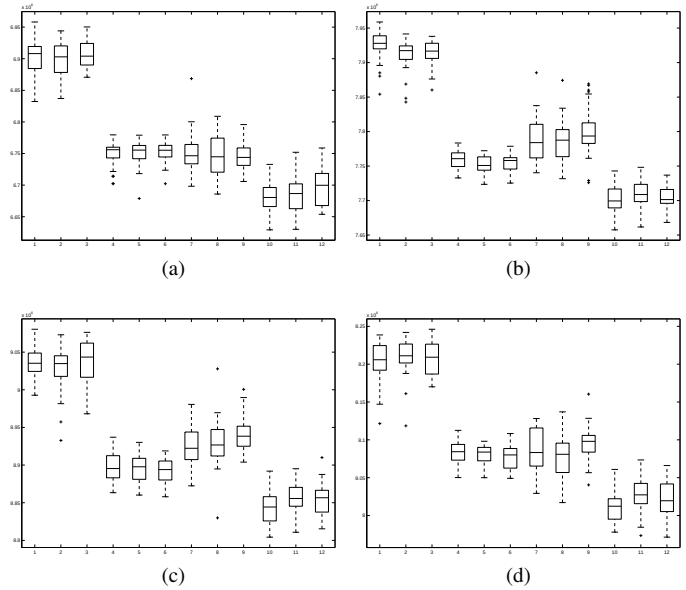


Figure 8: MWT: Boxplots for (a) LD120-1, (b) LD120-2, (c) LD120-3, and (d) LD120-4.



(a) (b) (c) (d)

Figure 9: MWT: Boxplots for (a) LD160-1, (b) LD160-2, (c) LD160-3, and (d) LD10-4.



(a) (b) (c) (d)

Figure 10: MWT: Boxplots for (a) LD200-1, (b) LD200-2, (c) LD200-3, and (d) LD200-4.