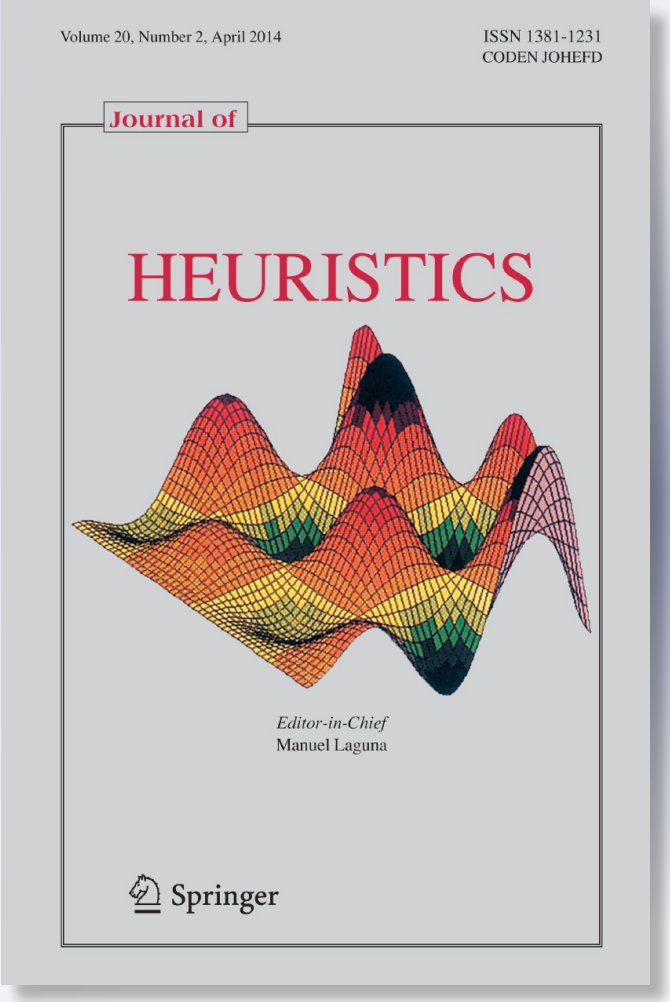
Approximated algorithms for the minimum dilation triangulation problem

Maria Gisela Dorzán, Mario Guillermo Leguizamón, Efrén Mezura-Montes & Gregorio Hernández-Peñalver

Journal of Heuristics

ISSN 1381-1231
Volume 20
Number 2

J Heuristics (2014) 20:189-209
DOI 10.1007/s10732-014-9237-2



Springer

J Heuristics (2014) 20:189–209
DOI 10.1007/s10732-014-9237-2

Approximated algorithms for the minimum dilation triangulation problem

Maria Gisela Dorzán · Mario Guillermo Leguizamón · Efrén Mezura-Montes · Gregorio Hernández-Peñalver

Received: 16 November 2012 / Revised: 15 October 2013 / Accepted: 25 January 2014 /
Published online: 9 February 2014

Abstract The complexity status of the minimum dilation triangulation (MDT) problem for a general point set is unknown. Therefore, we focus on the development of approximated algorithms to find high quality triangulations of minimum dilation. For an initial approach, we design a greedy strategy able to obtain approximate solutions to the optimal ones in a simple way. We also propose an operator to generate the neighborhood which is used in different algorithms: Local Search, Iterated Local Search, and Simulated Annealing. Besides, we present an algorithm called Random Local Search where good and bad solutions are accepted using the previous mentioned operator. For the experimental study we have created a set of problem instances since no reference to benchmarks for these problems were found in the literature. We use the sequential parameter optimization toolbox for tuning the parameters of the SA algorithm. We compare our results with those obtained by the *OV-MDT* algorithm that uses the obstacle value to sort the edges in the constructive process. This is the only available

M. G. Dorzán (✉) · M. G. Leguizamón
Universidad Nacional de San Luis, Ejército de Los Andes 950,
D5700HHW San Luis, Argentina
e-mail: mgdorzan@unsl.edu.ar

M. G. Leguizamón
e-mail: legui@unsl.edu.ar

E. Mezura-Montes
Departamento de Inteligencia Artificial, Universidad Veracruzana,
Sebastián Camacho 5, Centro, 91000 Xalapa, Veracruz, Mexico
e-mail: emezura@uv.mx

G. Hernández-Peñalver
Facultad de Informática, Universidad Politécnica de Madrid,
Campus de Montegancedo Boadilla del Monte, 28660 Madrid, Spain
e-mail: gregorio@fi.upm.es

algorithm found in the literature. Through the experimental evaluation and statistical analysis, we assess the performance of the proposed algorithms using this operator.

Keywords Computational geometry · Metaheuristics · Triangulation · Minimum dilation

1 Introduction

Triangulations are applied in different areas such as wireless sensor networks (Zhou et al. 2011), computational geometry (de Berg et al. 2000), computer graphics (Kolingerová and Ferko 2001), scientific visualization (Nielson 1997), robotics (Bokka and Gurla 1998), computer vision (Vite-Silva et al. 2007), image synthesis (Chen and Medioni 1997), as well as in mathematical and natural science. Different measures are adopted to design optimal triangulations. The most popular are the weight, stabbing number,[1] area, and dilation. Although the usage of approximated approaches to solve complex optimization problems is common nowadays, this last measure has not been used as selection criteria when optimizing triangulations by using metaheuristic algorithms (Michalewicz and Fogel 2004).

The dilation of a triangulation can be defined as follows: Let S be a finite planar point set, T a triangulation of S and u, v two points in S. There are two distance metrics: the Euclidean distance between u and v, $|uv|$, and the shortest path distance between u and v with respect to T, $dist(u, v)$. The shortest path distance represents the minimum distance we must cover in order to travel from u to v when we are only allowed to use the edges in T. The dilation between u and v with respect to T is the ratio between the shortest path and the Euclidean distances between u and v, and is defined as:

$$\Delta_T(u, v) = \begin{cases} 1, & u = v \\ \frac{dist(u,v)}{|uv|}, & u \neq v \end{cases} \tag{1}$$

Intuitively, the dilation measures the quality of the connection between u and v in T. If the dilation is large, this means that we have to travel a long way along the edges in T in order to reach v from u when the direct route would be much shorter.

The maximum over all the dilations between pairs of vertices in T is called the dilation of T and is represented by $\Delta(T)$. It measures the quality of the connection between any two vertices in T. The best possible dilation of any triangulation of S is the dilation of S and is denoted by $\Delta(S)$. Thus, we have

$$\Delta(T) = \max_{u,v \in S} \Delta_T(u, v) \tag{2}$$

and

$$\Delta(S) = \min_{T \text{ of } S} \Delta(T) \tag{3}$$

[1] For a given triangulation, the stabbing number is the maximum number of edges that are encountered (in their interior or at an endpoint) by any infinite line.

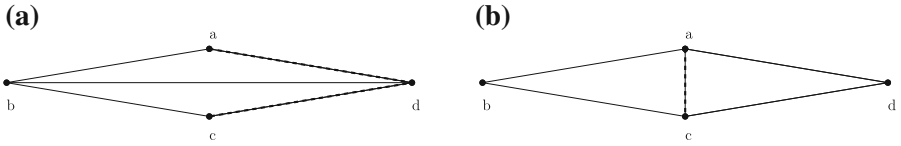


Fig. 1 A set of points $\{a, b, c, d\}$ and two possible triangulations. In triangulation (**a**), the dilation between points a and c is very high, whereas in triangulation (**b**) achieves the lowest possible dilation. The *bold dashed lines* represent the shortest path between a and c

The triangulation T^* which dilation is the dilation of S, i.e. $\Delta(T^*) = \Delta(S)$, is called the minimum dilation triangulation of S. In Fig. 1, we show an example of dilation for a set of points and two possible triangulations.

The rest of this paper is organized as follows. First, we give an overview of the related work and background, describing previous attempts at solving the MDT problem. In the next section we present the approaches developed in our research. Greedy, Local Search, Iterated Local Search, Simulated Annealing algorithms, and the new proposed algorithm Random Local Search are described including their respective operators and parameters. Section 3 describes the experimental framework used to evaluate the performance of the proposed algorithms giving a summary of the results obtained. We compare the results obtained by the proposed algorithms against those obtained by the existing algorithm in literature (*OV-MDT*). The use of different statistical tests has become necessary to confirm which of the proposed methods offers a significant improvement over the others. Therefore, we present a statistical analysis of the results to decide which algorithm has the best performance. The last section is deserved for the conclusions and future vision.

2 Related work

In 2006 several authors considered the question, whether computing certain graphs of minimum dilation is NP-hard. The graph dilation is also called stretch factor (Narasimhan and Smid 1999). A geometric graph G is called a t-spanner if $\Delta(G) \leq t$, and it is said to t-approximate the complete graph (Keil and Gutwin 1992). A good overview of the results dealing with graph dilation known until 2000 is given by Eppstein's survey (Eppstein 1997). Even more recent results have been included in (Narasimhan and Smid 2007). Klein and Kutz (2006) proved that the problem to compute a minimum dilation graph $G = (S, E)$ with less than $\frac{5920}{5919}n - \frac{7624}{5919}$ edges for a given set $S \subset R^2$ of n points is NP-hard.

Cheong et al. (2007) showed that it is NP-hard to determine whether a spanning tree of a set of point S with dilation at most $\Delta(S)$ exists. These points have integer coordinates in the plane and a rational dilation $\Delta(S) > 1$. Giannopoulos et al. (2007) proved that computing a Hamilton circuit or path of minimum dilation for a given set of points in the plane is NP-hard.

There are special cases of problem instances that can be solved efficiently. Eppstein and Wortman (2005) showed that, given a set of terminals, choosing a point in the plane and connecting it to all the given terminals such that a star graph is created of minimum dilation can be solved in $O(n \log n)$ time. However, if one of the given terminals has

to be chosen as the center of the star, then the problem becomes considerably harder. A randomized algorithm is presented to solve this problem in $O(n2^{\alpha(n)} \log^2 n)$ time, where $\alpha(n)$ is the inverse Ackermann function. Knauer and Mulzer (2005b) computed a simple lower bound for the dilation of the regular n-gon and use this bound in order to derive an efficient approximation algorithm that computes a triangulation whose dilation is within a factor of $1 + O(1/\sqrt{\log n})$ of the optimum.

The research on minimum dilation triangulations is scarce, even though there has been some work on estimating the dilation of certain types of triangulations that had already been studied in other contexts. Chew (1986) showed that the rectilinear Delaunay triangulation has dilation at most $\sqrt{10}$. Bonichon et al. (2012) improved this bound to $\sqrt{4+2\sqrt{2}} \approx 2.61$. A similar result for the Euclidean Delaunay triangulation was given by Dobkin et al. (1990) where they showed that the dilation of the Euclidean Delaunay triangulation is bounded above by $((1+\sqrt{5})/2)\pi \approx 5.08$. This bound was further improved to $2\pi/(3\cos(\pi/6)) \approx 2.42$ by Keil and Gutwin (1989). Xia (2011) proved that the dilation of the Delaunay triangulation of a set of points in the plane is less than $\rho = 1.998$, improving the previous best upper bound of 2.42. Xia and Zhang (2010) presented an improved lower bound of 1.5932 for the dilation of the Delaunay triangulation.

When considering optimal triangulations, it is instructive to look at local properties of these triangulations, since local properties improve our understanding of the structure of global optimal triangulations and sometimes lead to efficient algorithms to compute them. One important class of local properties is constituted by exclusion regions. Exclusion regions give us a necessary condition for the inclusion of an edge into an optimal triangulation. Let u and v be two points in a planar point set S, then the edge $e = uv$ can only be contained in an optimal triangulation of S if no other points of S lie in certain parts of the exclusion region of S. Knauer and Mulzer (2005a) showed that an edge e can only be included in the minimum dilation triangulation of S, if at least one of the two half circles with radius $\alpha|e|$ whose center is the center of e is empty. Here α denotes any constant such that $0 < \alpha < 3\cos(\pi/6)/(4\pi) \approx 0.2067$.

Usually, exclusion regions are applied as an initial filter of algorithms that compute optimal triangulations (Beirouti and Snoeyink 1998; Drysdale et al. 1995). The remaining edges are processed using some other local properties that give sufficient conditions for the inclusion of an edge. For the minimum dilation triangulation, however, it is not yet clear what to do with the remaining edges, since no other useful local properties are known that could be used in further processing steps. Finding such local properties remains an open problem.

The upper bound of (Keil and Gutwin 1989) can also be used to obtain a sufficient condition for the inclusion of an edge. Let p and q be two points of S, $\gamma = 2\pi/(3\cos(\pi/6))$ and the ellipsoid

$$E_{p,q,\gamma} = \left\{ x \in R^2 / |px| + |qx| \leq \gamma |pq| \right\} \tag{4}$$

with foci p and q. If $E_{p,q,\gamma}$ is empty, then the edge pq has to be included in the minimum dilation triangulation of S, since otherwise the dilation between p and q would be would be higher than γ.

Klein (2006a) in his Master thesis, proposed a greedy algorithm for approximating the minimum dilation triangulation for small sets of points. In fact, in his experimental evaluation only results for sets of up to 10 points were shown. This algorithm is based on the obstacle value of the edges. The obstacle value of an edge is the maximum dilation that its presence causes. The existence of that edge in the triangulation can block the direct path between two points and cause a detour. In the algorithm the edges are sorted in ascendent order considering the obstacle value. Then the edge with the lower obstacle value is inserted if it does not intersect with the edges previously added and if a complete triangulation is not achieved.

Computing the minimum dilation triangulation for a set of points in the plane is listed as an open problem in Eppstein's survey (Eppstein 2000). There are no polynomial algorithms reported in the literature which can build it. Moreover, it has not been proved that the problem is NP-hard. Therefore, one approach is to use metaheuristic techniques for obtaining approximate solutions to the optimum. In this work we show results of several techniques: Greedy (Edmonds 1971), Local Search (Aarts and Lenstra 1997; Papadimitriou 1976), Iterated Local Search (Martin et al. 1991), Simulated Annealing (Černý 1985; Kirkpatrick et al. 1983), and Random Local Search.

3 Metaheuristics applied to the MDT problem

Computing optimal solutions is intractable for many optimization problems of industrial and scientific importance. In practice, we are usually satisfied with "good" solutions, which are obtained by heuristic or metaheuristic algorithms.

Due to the fact that there are no works in the literature where the MDT problem is approached using metaheuristics and also that the complexity of the MDT problem is unknown, we propose a set of simple techniques as a first approach to approximate the minimum dilation triangulation. In the rest of this section, we briefly describe a greedy algorithm and the metaheuristic techniques considered in this work. We present the general overview of the studied algorithms: Greedy, Local Search (LS), Iterated Local Search (ILS), Simulated Annealing (SA), and Random Local Search (RLS) describing their features and parameters.

Each optimization problem consists of a solution space and objective function. We describe these concepts related to the MDT problem which are considered by all the algorithms presented in this work (except for the greedy algorithm as it is a constructive algorithm). The solution space $\mathcal{E}$ is represented by all possible triangulations of a set S of n points in the plane. An $n \times n$ matrix of ones and zeros is used to represent a possible solution. A 1 at position (i, j) means that there is an edge connecting points i and j, otherwise a 0 is placed. The objective function, $f : \mathcal{E} \rightarrow R$, assigns to each element of $\mathcal{E}$ a real value. For each $E \in \mathcal{E}$, the function f is defined as the maximum dilation between all pairs of points in E.

3.1 Greedy Algorithm

Greedy algorithms start from an empty solution and construct, in a deterministic way, a solution by assigning values to one decision variable at a time, until a complete

solution is generated. At each step, a local heuristic is used to select the new element to be included in the solution. In general, this heuristic chooses the best element from the current list in terms of its contribution in minimizing locally the objective function. Once an element is selected to be part of the solution, it is never replaced by another element. There is no backtracking of the already taken decisions.

Particularly, for the MDT problem, we present a greedy algorithm *G-MDT* which starts with an empty solution *Sol* and inserts edges until a triangulation for a given set of point S is generated.

Let A be the set of all possible edges that have not been inserted in Sol, $u, v \in S$ and $e = uv \in A$. If the dilation of the points u and v determines the dilation of the solution, i.e., $\Delta_{Sol}(u, v) = \Delta(Sol)$, then the edge e is inserted in the solution Sol. This reduces the dilation of the points u and v to 1 because $dist(u, v) = |uv|$ and consequently decreases the dilation of the solution Sol. The insertion is performed whenever the new edge to insert produces no intersections with the edges already inserted.

Algorithm 1 describes how the greedy technique works for the MDT problem. The main functions used are *allEdges(S)* and *largestDilationEdge(A, Sol)*. *allEdges(S)* returns the set of all possible edges considering the set of points S, and *largestDilationEdge(A, Sol)* returns the edge $(u, v) \in A$ where $\Delta_{Sol}(u, v) = \Delta(Sol)$.

Algorithm 1 G-MDT

```
Sol ← Ø
A ← allEdges(S)
while Sol is not a triangulation do
  e ← largestDilationEdge(A, Sol)
  if e do not intersect edges of Sol then
    Sol ← Sol ∪ {e}
  end if
  A ← A \ {e}
end while
return Sol
```

3.2 Local Search Algorithm

Local search is likely the oldest and simplest metaheuristic method (Aarts and Lenstra 1997; Papadimitriou 1976). It starts at a given initial solution. At each iteration, the current solution is replaced by a neighbor that improves the objective function. The search stops when all candidate neighbors are worse than the current solution, meaning that a local optimum is reached. For large neighborhoods, the candidate solutions may be a subset of the neighborhood. The main objective of this restricted neighborhood strategy is to speed up the search.

The proposed local search algorithm *LS-MDT* starts from a random initial solution and then iteratively moves to a neighbor solution. An operator is applied to such solution so as to generate a new one which is expected to improve its quality. If a better solution is found, it replaces the current solution by the new one and it continues the process until it can not improve the current solution. When no improved solutions are present in the neighborhood, local search is stuck at a local optimal solution.

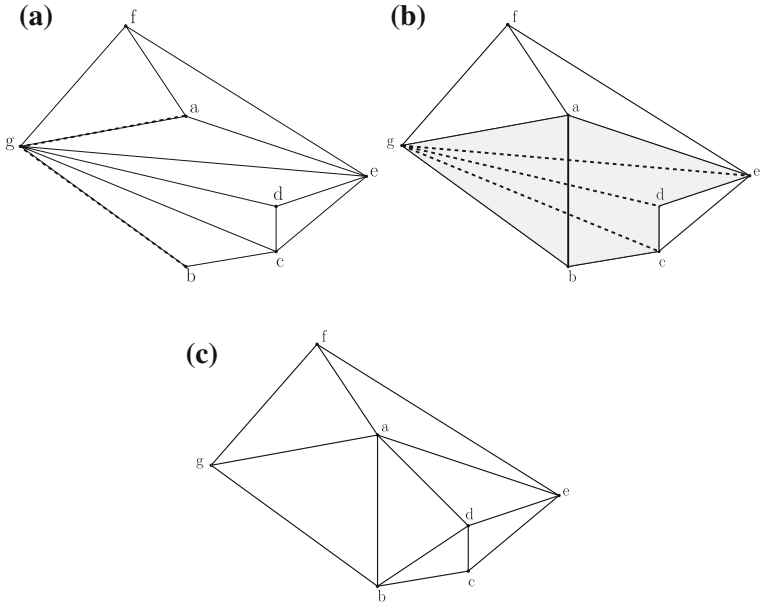


Fig. 2 *retriang()* operator. In (**a**), $\Delta_{Sol}(a, b) = \Delta(Sol)$ is shown in *dashed style*. In (**b**), the edge ab is added and the dashed edges intersected by ab are deleted. The *grey region* is retriangulated in a greedy way (**c**)

The moves in the search space are performed using the *retriang()* operator that works as follows. Let $a, b \in S$, if the dilation of the points a and b determines the dilation of the current solution Sol, i.e., $\Delta_{Sol}(a, b) = \Delta(Sol)$ (see Fig. 2a), then a and b are joined by an edge. The edges intersected by ab are deleted and the region delimited by these edges (see Fig. 2b) is retriangulated in a greedy way. The edge whose points have the higher dilation is inserted at each step if it does not intersect with the edges previously added and if a complete triangulation is not achieved (see Fig. 2c). Therefore the current solution is improved because its dilation has been reduced. Due to the behavior of this operator only a single neighbor is obtained.

Algorithm 2 describes how the *LS-MDT* algorithm works. We use the *retriang()* operator to obtain the neighbor of a solution.

Algorithm 2 LS-MDT

Generate a random initial solution $Sol \in \mathcal{E}$
$Sol' \leftarrow retriang(Sol)$
while Sol' improves Sol **do**
  $Sol \leftarrow Sol'$
  $Sol' \leftarrow retriang(Sol)$
end while
return Sol

3.3 Iterated Local Search Algorithm

The quality of the local optimum obtained by a local search method depends on the initial solution. As we can generate local optimum with high variability, iterated local search may be used to improve the quality of successive local optimum. This kind of strategy has been applied first in (Martin et al. 1991) and then generalized in (Lourenco et al. 2002).

Local search methods build a trajectory in the search space which leads from an initial solution to a local optimum, where the local search stops. If the neighbors of that local optimum do not improve its objective function, local search needs to be modified to escape from a local optimum and continue the search beyond local optimality. A simple modification consists of iterating calls to the local search routine, each time starting from a different initial configuration. Iterated Local Search is based on building a sequence of locally optimal solutions by perturbing the current local minimum and applying local search after starting from the modified solution. The perturbation strength has to be sufficient to lead the trajectory to a different attraction basin leading to a different local optimum.

The proposed algorithm *ILS-MDT* works as follows. First, a local search is applied to an initial random solution and yields a local optimum. We use the *LS-MDT* algorithm as given in Algorithm 2. At each iteration, a perturbation of *Sol* is carried out and then, a local search is applied to the perturbed solution Sol'. This process iterates until the number of perturbations is less than 50 and the current solution is improved, always keeping the best solution found so far Sol_{best}.

Complete details of *ILS-MDT* can be found in Algorithm 3. The *perturb(Sol)* procedure is performed by the perturbation operator where $n/5$ random local retriangulations over random selected points of S are performed worsening the current solution, i.e., a point $b \in S$ is randomly chosen and all the points adjacent to b form the grey region that has to be retriangulated (see Fig. 3a). The edges belonging to the region are deleted and then this region is retriangulated in a random way. The edges of the region are inserted at random if they do not intersect with the edges previously added and if a complete triangulation is not achieved (see Fig. 3b).

Algorithm 3 ILS-MDT

Generate a random initial solution $Sol \in \mathcal{E}$
$Sol \leftarrow LS\text{-}MDT(Sol)$
repeat
  $Sol' \leftarrow perturb(Sol)$
  $Sol \leftarrow LS\text{-}MDT(Sol')$
until termination criterion is reached
return Sol_{best}

3.4 Simulated Annealing Algorithm

Simulated Annealing (SA) applied to optimization problems emerges from the work of Kirkpatrick et al. (1983) and Černý (1985). SA is based on the principles of statistical mechanics whereby the annealing process requires heating and then slowly cooling a

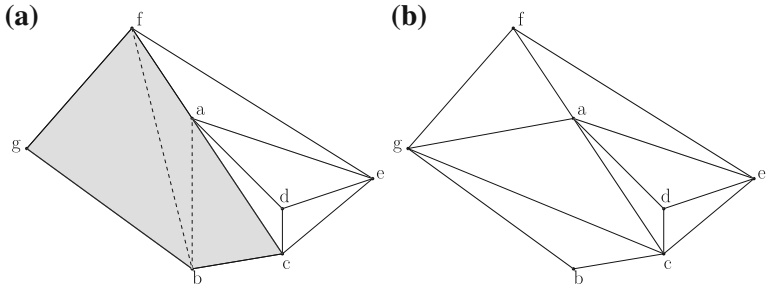


Fig. 3 In (**a**), the *grey region* formed by the adjacent points of b (g, f, a, c) is shown. In (**b**), random edges are added to that region

substance to obtain a strong crystalline structure. The strength of the structure depends on the rate of cooling metals. If the initial temperature is not sufficiently high or a fast cooling is applied, imperfections (metastable states) are obtained. In this case, the cooling solid will not attain thermal equilibrium at each temperature. Strong crystals are grown from careful and slow cooling. SA simulates the energy changes in a system subjected to a cooling process until it converges to an equilibrium state.

The algorithm *SA-MDT* (see Algorithm 4) works as follows. From an initial solution, it executes a number of iterations where a neighbor of the current solution *Sol* is generated in each one of them. The moves that improve the cost function are always accepted. Otherwise, the neighbor Sol' is selected with a given probability that depends on the current temperature T. This probability is usually called *acceptance function* and it is evaluated according to

$$p\left(T, Sol, Sol'\right) = e^{-\frac{\delta}{T}} \tag{5}$$

where $\delta = f(Sol') - f(Sol)$.

$M(T_k)$ moves are performed for each temperature T_k. Finally, the value of T_k is decreased at each algorithm iteration k. The algorithm continues this way until the termination condition is met. Once an equilibrium state is reached, the temperature is gradually decreased according to a cooling schedule such that few nonimproving solutions are accepted at the end of the search. The best solution found since the beginning of the search is stored.

Considering the MDT problem, some issues must be considered in the design of SA algorithm:

- *Initial solution*: a random triangulation is used.
- *Initial temperature* T_0: different initial temperatures are considered: i) $1000 \times (n/4)$; ii) $1000 \times (n/2)$, and iii) the maximum length of the paths of the initial solution.
- *Temperature decrement rule* $\mathcal{R}$: we use Geometric Decrease ($T_{k+1} = \alpha T_k$ with $\alpha = 0.95$).
- *Number of moves at each temperature* $M(T_k)$ *(Markov chain)*: we use (i) $M(T_k) = \lceil T_k \rceil$ to ensure that the amount of moves is directly proportional to the actual temperature. When the temperature T_k is less or equal than the double of the number of edges, it performs the double of the number of edges moves in order

Algorithm 4 SA-MDT

```
Generate an initial solution Sol ∈ E
k ← 0
while termination condition not met do
  i ← 0
  while i < M(T_k) do
    Sol' ← N(Sol)
    δ ← f(Sol') − f(Sol)
    if δ < 0 then
      Sol ← Sol'
    else
      Sol ← Sol' with probability p(T, Sol, Sol') //see Equation (5)
    end if
    i ← i + 1
  end while
  T_{k+1} ← R(T_k)
  k ← k + 1
end while
```

not to make few moves when the temperature is low. Therefore, when $T_k \leq 2 \times e$, then $M(T_k) = 2 \times e$, where e is the number of edges of the initial solution; and (ii) $M(T_k) = e$.

- *Termination condition*: the search process is finished when the temperature is less than or equal to 0.005, i.e., $T_f = 0.005$.
- *Neighborhood of a solution* $\mathcal{N}(x)$: two operators are used.
  1. Local random retriangulation (RR) as used in the perturbation operator in *ILS-MDT* algorithm.
  2. Local greedy retriangulation (GR) is similar to the RR described above, but the edges are inserted in a greedy way. A point $b \in S$ is randomly chosen and all the points adjacent to b are recovered (see Fig. 4a). Then the grey region that forms the adjacent points to b is retriangulated using at each step the edges whose points have the higher dilation (if it does not intersect with that previously added (see Fig. 4b).

  The sequence of application of these operators is based on the fact that the GR operator can lead to the same solution to which it applies (if the region to retriangulate is the same). Therefore, it is interleaved with the RR operator to better explore the search space. The application scheduling for these operators is as follows. GR is performed g times and RR is performed r times. We considered $g \in \{4, 6, 8\}$ and $r = 2$ as will be described later in the next section.

3.5 Random Local Search Algorithm

Initially, the proposed SA algorithm used the operator *retriang()* (described in Sect. 3.2) to move in the solution space. We observed that such algorithm converged too early to suboptimal regions (the best solution was achieved during the first temperature). Therefore we proposed another operator for the SA algorithm to avoid this condition of premature convergence.

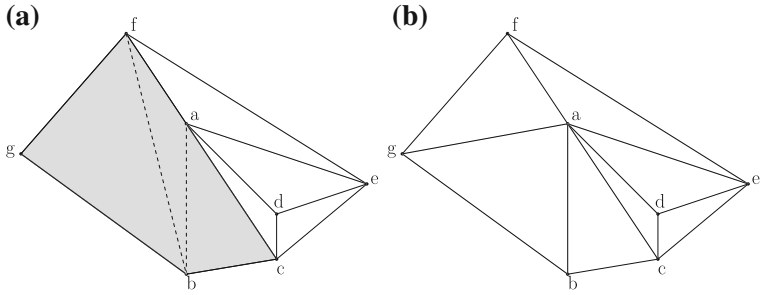


Fig. 4 In (**a**), the *grey region* formed by the adjacent points of b (g, f, a, c) is shown. In (**b**), the edges are added in a greedy way

On the other hand, the results obtained with operator *retriang()* were encouraging with the LS and ILS algorithms (as it will be detailed in the next section of the paper). Therefore, we propose a new algorithm; we called it RLS algorithm, which starts with a random initial solution. $maxEvals = 2n$ evaluations are performed, accepting good and bad solutions. This allows to explore and exploit the search space, always keeping the best solution found so far Sol_{best}. We consider the operator *retriang()* but the retriangulation of the region is calculated in a random manner because better results were obtained.

The algorithm *RLS-MDT* is showed in Algorithm 5 remaining that the operator *retriang()* returns a single neighbor.

Algorithm 5 RLS-MDT

Generate an initial solution $Sol \in \mathcal{E}$
$k \leftarrow 0$
while $k < maxEvals$ **do**
  $Sol \leftarrow retriang(Sol)$
  $k \leftarrow k + 1$
end while
return Sol_{best}

4 Experimental Evaluation and Statistical Analysis

After reviewing the literature on the presented issues we found no collections of instances for the MDT problem. Therefore, no results are available for comparing our proposed algorithms with some others previously studied. The collections of problem instances were designed by the authors, using an *instance generator* with different functions of Computational Geometry Algorithms Library.[2] A collection of 10 instances respectively of size 40/80/120/160/200 were generated; i.e., a total of 50 problem instances, each one is called LDn-i, the size of the instance i, $1 \leq i \leq 10$, is denoted by n. The points are randomly generated, uniformly distributed with coordi-

[2] http://www.cgal.org.

nates $x, y \in [0, 1000]$. For implementation purposes, there are non collinear points. The proposed algorithms were implemented in C language.

This paper shows the results obtained for instances *LDn-i*, with $n = 40, 80, 120, 160, 200$ and $1 \leq i \leq 4$. For stochastic algorithms (*LS-MDT*, *ILS-MDT*, *SA-MDT*, and *RLS-MDT*) twenty-five runs were carried out with different initial seeds and different triangulations for each run.

The statistical validation of results is necessary to establish a conclusion on an experimental analysis. Statistical tests allow us to determine whether the differences observed in results obtained are significant with respect to the choices taken and whether the conclusions remarked are supported by the experimentation carried out. We can find different studies that propose methods for conducting comparisons among various approaches (Demšar 2006; Markatou et al. 2005).

We performed the Kolmogorov-Smirnov test to determine if the samples follow a Normal distribution. As all samples have a non-Normal distribution we used non-parametric statistical tests to evaluate the algorithms. In order to carry out a comparison which involves more than two algorithms Kruskal–Wallis test was used for multiple comparisons with independent samples (Derrac et al. 2011). The idea of this test is to rank all the results from all algorithms together and then apply a one-way ANOVA to the ranks rather than to the original results. In order to compare the performance of the algorithms on the whole set of instances, we applied the Friedman test for related samples. These tests help to determine if there are significant differences among the results obtained by the algorithms but do not show which algorithms have significant differences in their results. We used the post-hoc Tukey test to find which algorithms' results are significantly different from one another. These tests are well-known and they are usually included in standard statistics packages (such as *Matlab*, *R*, etc.).

Due to the complexity involved in tuning the parameters of metaheuristic techniques, we used Sequential Parameter Optimization (SPO) (Bartz-Beielstein 2006) for tuning the parameters required by SA. SPO is a framework for tuning and understanding of algorithms by active experimentation and employs methods from error statistics to obtain reliable results. SPO is a heuristic that combines classical and modern statistical techniques. The Sequential Parameter Optimization Toolbox (SPOT) (Bartz-Beielstein 2010) is an implementation of the SPO framework. It has been successfully applied to numerous heuristics for practical and theoretical optimization problems. SPOT provides tools for tuning many parameters simultaneously and it is well-suited for optimization problems and it can reach good results with only a few model-building experiments since it builds a surrogate model during its sequence of runs. This is constantly refined as the tuning progresses. SPOT has been made available as an R-package.[3] SPOT employs a sequentially improved model to estimate the relationship between algorithm input variables and its output. Therefore SPOT looks for two goals: one goal is to enable determining good parameter settings, thus SPOT may be used as a tuner. Secondly, variable interactions can be revealed for helping in understanding how the tested algorithm works when confronted

[3] http://CRAN.R-project.org/package=SPOT.

Table 1 Tunable parameters and their ROI

	ROI	Best
$M(T_k)$	[1, 2]	1
$\mathcal{N}(x)$	[1, 3]	3
T_0	[1, 3]	3

The best tuning results are shown in column "Best"

with a specific problem or how changes in the problem influence the algorithm's performance.

All SPOT-tuning experiments for the SA-MDT algorithm were performed with the following settings (see Bartz-Beielstein 2010 for further details): 50 sequence steps, five new design points in each step, up to two repeats per design point, and seven initial design points. Random Forest was used as a fast surrogate model building tool. Latin hypercube sampling was chosen as the generator of design points.

The Region Of Interest (ROI) contains the interesting value ranges of the algorithm parameters that should be optimized. These ranges are specified by two design points (high and low limit value). Categorical variables are encoded as numerical values. The ROI of the tunable parameters of *SA-MDT* is composed as follows.

- Number of moves at each temperature $M(T_k)$: (1) T_k and (2) number of edges of the initial solution.
- Neighborhood of a solution $\mathcal{N}(x)$: local retriangulation interleaving with (1) $r = 4$, $g = 2$, (2) $r = 6$, $g = 2$, and (3) $r = 8$, $g = 2$.
- Initial temperature (T_0): (1) $1000 \times (n/4)$, (2) $1000 \times (n/2)$, and (3) the maximum length of the paths in the initial solution.

In Table 1 the SPOT results are shown considering the tunable parameters. We obtained better results with T_k moves at each temperature ($M(T_k) = T_k$), using local retriangulation interleaving with $r = 8$ and $g = 2$ and with a initial temperature equal the maximum length of the paths in the initial solution.

4.1 Comparing best results of the proposed algorithms

In this section we present the best results obtained by the proposed algorithms. It should be noticed that *G-MDT* is a deterministic algorithm and therefore yields a single result per instance. Table 2 shows the best values obtained with the proposed algorithms (the lowest dilations are bolded): *G-MDT*, *LS-MDT*, *ILS-MDT*, *SA-MDT*, *SA-MDT*$_{+LS}$, and *RLS-MDT*. *SA-MDT*$_{+LS}$ is an improvement of the *SA-MDT* algorithm by applying local search (*LS-MDT*) to the best solution obtained for each instance. In the second column, the dilation of the Delaunay triangulation is shown in order to compare our results with a known triangulation that have a upper bound of 1.998 (Xia 2011).

In the third column we show the results obtained with the greedy algorithm proposed in (Klein 2006a). We called it *OV-MDT* because it computes the Obstacle Value of the edges for approximating the minimum dilation triangulation. We use the Java-applet available at (Klein 2006b) to obtain the results. It should be noted that the used applet is a time-consuming process and the instances considered in (Klein 2006b) do not

Table 2 Best results of *DT*, *OV-MDT*, *G-MDT*, *LS-MDT*, *ILS-MDT*, *SA-MDT*, *SA-MDT*$_{+LS}$, and *RLS-MDT* algorithms

Instance LD	*DT*	*OV-MDT*	*G-MDT*	*LS-MDT*	*ILS-MDT*	*SA-MDT*	*SA-MDT*$_{+LS}$	*RLS-MDT*
40-1	1.31871	1.32863	1.37203	1.30959	**1.29467**	**1.29467**	**1.29467**	**1.29467**
40-2	1.37491	**1.36881**	1.37491	**1.36881**	**1.36881**	**1.36881**	**1.36881**	**1.36881**
40-3	**1.32268**	**1.32268**	1.36026	**1.32268**	**1.32268**	**1.32268**	**1.32268**	**1.32268**
40-4	1.35391	**1.32330**	1.34919	**1.32330**	**1.32330**	1.32557	**1.32330**	**1.32330**
80-1	1.33034	**1.32363**	1.39394	**1.32363**	**1.32363**	1.40844	**1.32363**	**1.32363**
80-2	1.40500	1.35181	1.38199	1.35339	**1.32418**	1.50331	**1.32418**	**1.32418**
80-3	1.37791	**1.30519**	1.36180	1.34065	1.31457	1.37791	1.33739	**1.30519**
80-4	1.42547	1.34239	1.39362	1.33176	**1.31583**	1.47561	**1.31583**	**1.31583**
120-1	1.37428	**1.34442**	1.42386	1.35398	1.34825	1.72082	**1.34442**	**1.34442**
120-2	1.33973	1.31194	1.37778	1.31194	**1.30366**	1.65921	1.31194	1.31194
120-3	1.37724	1.34016	1.43027	1.40314	1.31985	1.74912	1.34664	**1.29786**
120-4	1.38529	**1.33600**	1.39972	1.35152	1.34272	1.68163	1.34918	**1.33600**
160-1	1.34365	**1.31050**	1.43076	1.35236	1.33384	1.95530	1.34834	**1.31050**
160-2	1.35409	NA	**1.33260**	1.36463	**1.33260**	1.97896	**1.33260**	**1.33260**
160-3	1.35797	1.33278	1.37723	1.37178	1.36352	1.87574	1.36352	**1.32995**
160-4	1.37922	**1.34941**	1.37922	1.37922	1.35037	1.87033	1.35943	**1.34941**
200-1	**1.35751**	NA	1.42220	1.41867	**1.35751**	2.08064	1.37162	**1.35751**
200-2	1.39512	NA	1.39512	1.36451	**1.36350**	2.06324	1.37118	**1.36350**
200-3	1.41962	NA	1.45763	1.40645	**1.40645**	2.09818	**1.40645**	**1.40645**
200-4	1.36965	NA	1.40765	1.35499	**1.35251**	2.00110	1.37509	**1.35251**

The lowest dilations are bolded

exceed ten points. The word *NA* means that the applet did not produce a result (the applet halted due to an error condition).

No results of other algorithms were found in the literature by the authors against which we can compare our results.

We observed that all the results obtained by the proposed algorithms are less than 1.998 and the Delaunay triangulation never obtains better results for the instances considered. The *G-MDT* algorithm obtained solutions of poor quality with respect to those of the other algorithms, even for the smallest instances. The *RLS-MDT* algorithm obtained the lowest (best) dilations for all problem instances (except for LD-120-2). We observed that the *OV-MDT* algorithm did not obtain better results than *RLS-MDT*. Although in some instances equal best values were obtained with some algorithms, the *RLS-MDT* algorithm is less complex and faster than the other algorithms, as it will be shown later in this section.

4.2 Comparing the performance of the proposed algorithms

In Table 3 we present the median values (the lowest are bolded) obtained by the proposed algorithms: *LS-MDT*, *ILS-MDT*, *SA-MDT*, *SA-MDT*$_{+LS}$, and *RLS-MDT*.

Table 3 Median values for *LS-MDT*, *ILS-MDT*, *SA-MDT*, *SA-MDT*$_{+LS}$, and *RLS-MDT* algorithms

Instance LD	*LS-MDT* (1)	*ILS-MDT* (2)	*SA-MDT* (3)	*SA-MDT*$_{+LS}$ (4)	*RLS-MDT* (5)
40-1	1.36786	**1.33142**	1.35891	**1.33142**	**1.33142**
40-2	**1.36881**	**1.36881**	**1.36881**	**1.36881**	**1.36881**
40-3	1.36333	**1.32268**	**1.32268**	**1.32268**	**1.32268**
40-4	1.34919	1.32557	1.34919	**1.32330**	**1.32330**
80-1	1.38273	**1.32363**	1.65695	1.36237	**1.32363**
80-2	1.38199	**1.32418**	1.62768	1.37217	1.35630
80-3	1.38183	**1.34662**	1.56403	1.37791	1.34745
80-4	1.38201	1.33176	1.62907	1.34413	**1.31583**
120-1	1.41295	**1.35398**	1.90133	1.42386	**1.35398**
120-2	1.37696	**1.31194**	1.84388	1.38851	**1.31194**
120-3	1.48917	1.35736	1.88084	1.41905	**1.31182**
120-4	1.44444	1.35868	1.86884	1.40879	**1.33600**
160-1	1.41974	1.36817	2.11985	1.41974	**1.31452**
160-2	1.41918	1.36463	2.13222	1.42867	**1.33260**
160-3	1.44772	1.37597	2.12003	1.43008	**1.32995**
160-4	1.42350	1.36432	2.13973	1.39009	**1.35943**
200-1	1.49990	1.38670	2.34612	1.44980	**1.35751**
200-2	1.40806	**1.38744**	2.28723	1.39112	**1.38744**
200-3	1.43856	**1.40645**	2.32721	1.44293	**1.40645**
200-4	1.46962	1.37217	2.35554	1.45886	**1.35251**

The lowest median values are bolded

We observed that the *RLS-MDT* algorithm obtained the lowest median values for all problem instances (except for LD-80-2 and LD-80-3).

Table 4 is used to show if there were significant differences between the proposed algorithms using the Kruskal–Wallis test. In the "KW test" column we show the p-value and, if the p-value is below 0.05, the last two columns show which algorithms' results had significant differences with respect to those obtained by *ILS-MDT* (2) and *RLS-MDT* (5) (because these algorithms obtained the best *Best* and *Median* values). For example, if the p-value associated to the instance i is lower than 0.05 and in the "Tukey test w.r.t. (5)" column is shown "1,3", this means that the results of the *RLS-MDT* algorithm (5) had significant differences with respect to those of the *LS-MDT* (1) and *SA-MDT* (3) algorithms for the instance i.

We observed that there was always a significant difference between the algorithms ($1.e{-}16 < p\text{-value} \leq 0.0015$). Smaller p-values were obtained while larger instances are considered, i.e., the difference was much larger when considering sets with more points. Considering the best and median values obtained by the *ILS-MDT* and *RLS-MDT* algorithms, we noted that they had similar performances for the whole set of instances. This situation was observed in the absence of significant differences between these two algorithms. Although both algorithms had similar performances, the *RLS-MDT* algorithm performed fewer evaluations than the others, as it can be seen in Fig. 5 where the convergence plots of the algorithms are shown, except for

Table 4 p-values of Kruskal–Wallis and Tukey tests for *LS-MDT*, *ILS-MDT*, *SA-MDT*, *SA-MDT*$_{+LS}$, and *RLS-MDT* algorithms

Instance LD	K–W test	Tukey test w.r.t. (5)	Tukey test w.r.t. (2)
40-1	5.4988e−008	1	1
40-2	0.0015	1	1
40-3	2.5823e−008	1	1
40-4	1.1102e−016	1, 2, 3	1, 3, 5
80-1	1.e−16	1, 3, 4	1, 3, 4
80-2	1.e−16	1, 2, 3	1, 3, 4, 5
80-3	1.e−16	1, 3	1, 3, 4
80-4	1.e−16	1, 3	1, 3
120-1	1.e−16	1, 3, 4	1, 3, 4
120-2	1.e−16	1, 3, 4	1, 3, 4
120-3	1.e−16	1, 3, 4	1, 3
120-4	1.e−16	1, 2, 3, 4	1, 3, 5
160-1	1.e−16	1, 2, 3, 4	1, 3, 4, 5
160-2	1.e−16	1, 3, 4	1, 3, 4
160-3	1.e−16	1, 2, 3, 4	1, 3, 5
160-4	1.e−16	1, 3, 4	1, 3, 4
200-1	1.e−16	1, 3, 4	1, 3
200-2	1.e−16	1, 3, 4	1, 3, 4
200-3	1.e−16	1, 3, 4	1, 3, 4
200-4	1.e−16	1, 3, 4	1, 3, 4

the *SA-MDT* algorithm because it performs many evaluations and this produces an unreadable plot. The x-axis represents the number of evaluations of the objective function and the y-axis represents the objective value for each evaluation. We plotted the data with logarithmic scale for the y-axis for a better display of the results. In each plot we magnified some interesting regions to observe the behavior of the algorithms. We observed that the *LS-MDT* and *ILS-MDT* algorithms behaved equal until the first perturbation of *ILS-MDT* (the first peak in the dotted function). The *LS-MDT* algorithm performed less evaluations while the *ILS-MDT* algorithm performed more evaluations, getting better results. This agrees with the theoretical aspects of both strategies. The *RLS-MDT* algorithm differs from others because it uses the RR operator instead of the GR operator (see magnified plots). *RLS-MDT* does not converge to a single value, it moves between suboptimal solutions until it consumes the given evaluations. Although the *RLS-MDT* algorithm accepts bad solutions to further explore the solution space, such solutions are not so bad (not very sharp peaks in the plots).

In Fig. 6 we present the results of the Tukey post-hoc test based on the Friedman test using two performance metrics (best and median) in order to compare the performance of the algorithms on the whole set of instances. Therefore we determined if there are significant differences between the algorithms and showed which algorithm behaved better than the others. In the x-axis, the confidence interval of mean ranks (given

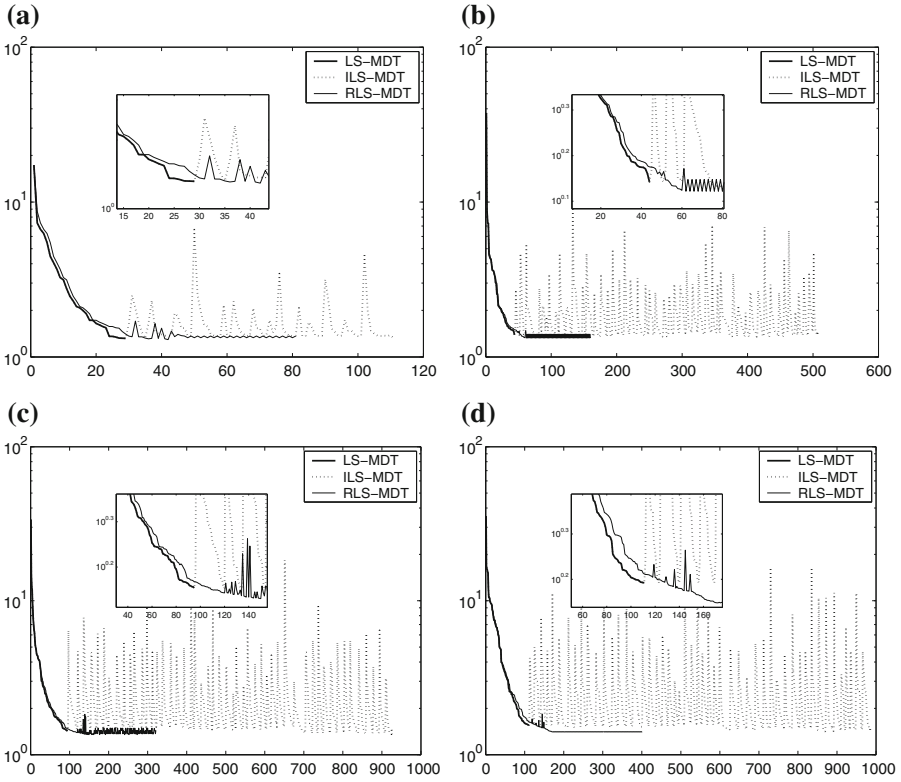


Fig. 5 Example of the behavior of the *LS-MDT*, *ILS-MDT*, and *RLS-MDT* algorithms for an instance of (**a**) 40, (**b**) 80, (**c**) 160, and (**d**) 200 points

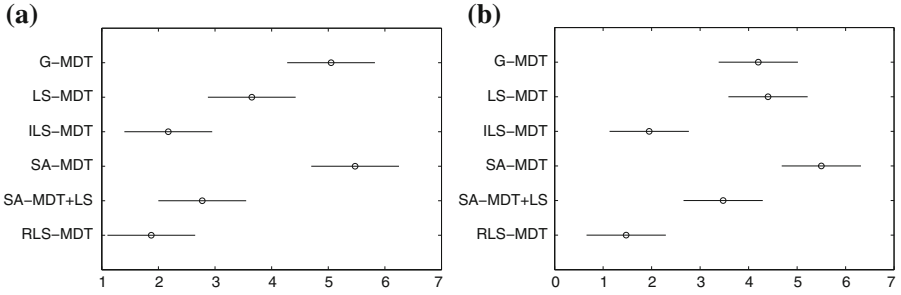


Fig. 6 Friedman test using (**a**) the best values and (**b**) the median values over the whole set of instances

by the Friedman test) is shown. As *G-MDT* is a deterministic algorithm, we use the single result as the best and median values. Using the best and median values for the remaining algorithms as well, the test yielded a p-value = 1.8919e−011 and a p-value = 4.2943e−013, respectively. Therefore, there were significative differences between the proposed algorithms. In both cases, we observed a better performance of *RLS-MDT* and *ILS-MDT* because they have the first rankings of the test.

Table 5 Runtimes of *DT*, *OV-MDT*, *G-MDT*, *LS-MDT*, *ILS-MDT*, *SA-MDT*, *SA-M DT*$_{+LS}$, and *RLS-MDT* algorithms

Instance LD	*DT*	*OV-MDT*	*G-MDT*	*LS-MDT*	*ILS-MDT*	*SA-MDT*	*SA-M DT*$_{+LS}$	*RLS-MDT*
40-1	0.0045	53.2532	0.439	0.032	0.737	863.163	863.196	0.250
40-2	0.0054	51.3404	0.191	0.032	0.471	998.381	998.413	0.227
40-3	0.0059	50.614	0.168	0.048	0.540	809.903	809.950	0.208
40-4	0.0063	66.7724	0.161	0.031	0.192	839.972	840.003	0.156
80-1	0.0108	1067.791	1.802	0.308	7.895	9931.709	9932.017	2.972
80-2	0.0113	854.6675	1.862	0.322	7.943	11046.521	11046.843	2.111
80-3	0.0114	1124.764	1.813	0.289	7.814	9646.330	9646.619	2.543
80-4	0.011	857.2655	1.846	0.367	7.947	15960.365	15960.732	2.744
120-1	0.0134	9196.828	7.952	1.615	43.575	69768.200	69769.815	7.888
120-2	0.0145	9223.472	7.929	1.521	30.039	66366.500	66368.021	7.990
120-3	0.0141	10816.831	8.150	1.777	30.877	76360.000	76361.777	9.332
120-4	0.0166	10360.759	8.503	1.554	29.988	66969.000	66970.554	7.644
160-1	0.0319	92250.185	23.005	5.556	86.127	209016.600	209022.156	26.306
160-2	0.0323	–	24.176	4.688	82.365	247275.400	247280.088	24.705
160-3	0.0311	61097.471	23.844	4.763	73.718	247970.600	247975.363	26.985
160-4	0.0333	64406.581	24.354	5.193	78.879	202503.600	202508.793	23.069
200-1	0.0497	–	54.947	11.136	176.614	479645.600	479656.736	52.146
200-2	0.0489	–	55.643	20.301	174.671	507958.400	507978.701	53.070
200-3	0.0495	–	56.711	12.638	173.478	556564.200	556576.838	60.524
200-4	0.0499	–	57.107	12.179	174.970	507628.200	507640.379	48.743

4.3 Runtime analysis

In this subsection we compare and analyze the computational effort of the studied algorithms applied to the MDT problem.

Table 5 shows the runtimes (in seconds) that each algorithm consumes to solve the problem instances considered. For a different point of view, Fig. 7 shows these data graphically. The x-axis and y-axis represent respectively, the problem instances studied in the experimental evaluation and the consumed time (in seconds and plotted in logarithmic scale).

The *SA-MDT* and *SA-M DT*$_{+LS}$ algorithms had similar runtimes due to the fact that *SA-M DT*$_{+LS}$ performs only a local search over the best solution found by break*SA-MDT*. Besides, both algorithms reach the highest runtimes since they perform more function evaluations with respect to the remaining proposed algorithms. Although the *DT* algorithm is the fastest one, its results are not competitive with respect to the results of the other methods. As expected, *ILS-MDT* is slower than *LS-MDT* because *ILS-MDT* repeatedly calls the local search process. The runtimes of *RLS-MDT* and *G-MDT* are similar; however, *RLS-MDT* obtains much better quality results. A runtime comparison between the proposed algorithms against *OV-MDT* could be unfair since this algorithm is implemented as an applet. Nevertheless, we also show the runtimes

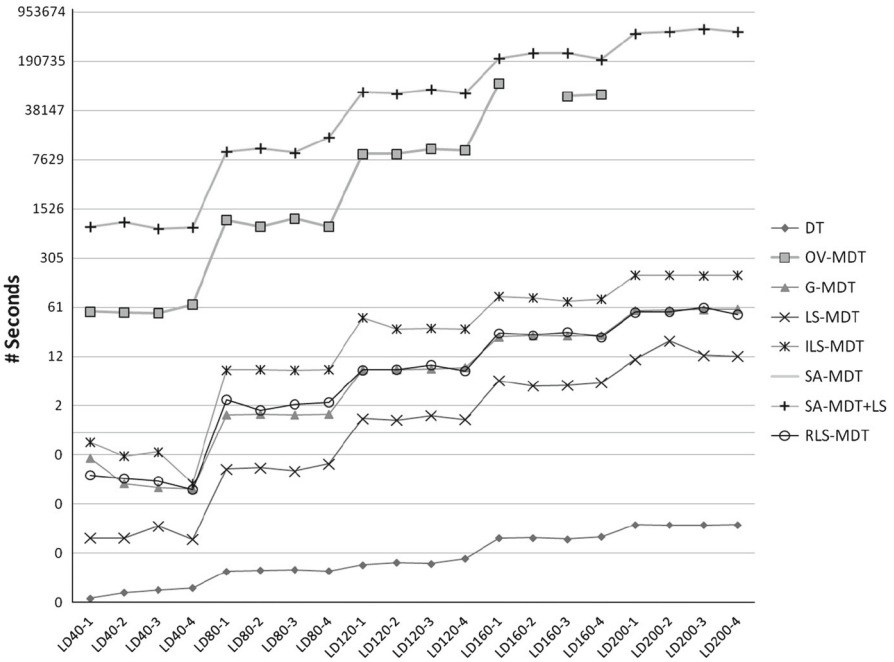


Fig. 7 Runtimes of *DT*, *OV-MDT*, *G-MDT*, *LS-MDT*, *ILS-MDT*, *SA-MDT*, $SA\text{-}MDT_{+LS}$, and *RLS-MDT* algorithms

for *OV-MDT* in order to include it in the overall runtime analysis. Considering this fact, *OV-MDT* is the second slowest algorithm. Even though *RLS-MDT* and *ILS-MDT* algorithms have similar performances, *ILS-MDT* is slower than *RLS-MDT* (by an average factor of 3.18).

5 Conclusions

The design of approximated algorithms for solving the Minimum Dilation Triangulation problem for sets of points in the plane and their respective experimental evaluation and statistical analysis were presented. In this paper we showed how some metaheuristic techniques can be used to find high quality triangulations of minimum dilation. We have created a set of instances for the experimental study since no reference to benchmarks for these problems were found in the literature. They are available at the research project site (http://www.dirinfo.unsl.edu.ar/bd2/GeometriaComp/).

Considering the experimental evaluation, we assessed the applicability of the LS, ILS, SA, and RLS algorithms for the MDT problem. The respective performances were analyzed with nonparametric statistical tests. The statistical analysis showed that *RLS-MDT* and *ILS-MDT* have similar performances and they are competitive with respect to the other algorithms in the problem instances considered. *RLS-MDT* obtained the best objective values for all the studied instances (except for LD120-2). The improvement achieved over the results of the greedy strategy resulted in a reduction on the dilation

values between 0.4 % and 9.2 %. Although *RLS-MDT* and *ILS-MDT* behave similarly, the *RLS-MDT* algorithm required less functions evaluations and it is faster than *ILS-MDT*. It should be noted that the existing algorithm (*OV-MDT*) yielded no results for four of the instances considered. Our best algorithm (*RLS-MDT*) outperformed 50 % of the instances considered and obtained the same results for the other instances.

The future work will address us to go further in our research in order to propose and design alternative metaheuristics such as genetic algorithms or ant colony optimization algorithms so as to analyze their performances with respect to the MDT problem.

Acknowledgments The authors would like to thank to Research Project Tecnologías Avanzadas de Bases de Datos 22/F014 financed by Universidad Nacional de San Luis, San Luis, Argentina; CONICET (Consejo Nacional de Investigaciones Científicas y Técnicas), Argentina; and ESF EUROCORES programme EuroGIGA, CRP ComPoSe: grant EUI-EURC-2011-4306, Spain. The authors acknowledge support from MINCyT-CONACyT through bilateral project No. MX/11/03 -164626.

References

Aarts, E., Lenstra, J.: Local Search in Combinatorial Optimization. Wiley, New York (1997)

Bartz-Beielstein, T.: Experimental Research in Evolutionary Computation: The New Experimentalism (Natural Computing Series). Springer-Verlag New York Inc., Secaucus (2006)

Bartz-Beielstein, T.: SPOT: an R package for automatic and interactive tuning of optimization algorithms by sequential parameter optimization. CoRR (2010)

Beirouti, R., Snoeyink, J.: Implementations of the LMT heuristic for minimum weight triangulation. In: Proceedings of the Fourteenth Annual Symposium on Computational Geometry, SCG '98, pp. 96–105. ACM, New York (1998)

Bokka, V., Gurla, H.: Constant-time algorithms for constrained triangulations on reconfigurable meshes. IEEE Trans. Parallel Distrib. Syst. **9**(11), 1057–1072 (1998)

Bonichon, N., Gavoille, C., Hanusse, N., Perkovic, L.: The stretch factor of l_1- and l_{inf}-delaunay triangulations. CoRR abs/1202.5127 (2012)

Černý, V.: Thermodynamical approach to the traveling salesman problem: an efficient simulation algorithm. J. Optim. Theory Appl. **45**, 41–51 (1985)

Chen, Q., Medioni, G.: Image synthesis from a sparse set of views. In: Proceedings of the 8th Conference on Visualization '97, VIS '97, p. 269-ff. IEEE Computer Society Press (1997)

Cheong, O., Haverkort, H., Lee, M.: Computing a minimum-dilation spanning tree is NP-hard. In: Proceedings of the Thirteenth Australasian Symposium on Theory of Computing, CATS '07, vol. 65, pp. 15–24 (2007)

Chew, P.: There is a planar graph almost as good as the complete graph. In: Proceedings of the Second Annual Symposium on Computational Geometry, SCG '86, pp. 169–177. ACM, New York (1986)

de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications, 2nd edn. Springer-Verlag, Heidelberg (2000)

Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**, 1–30 (2006)

Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evolut. Comput. **1**(1), 3–18 (2011)

Dobkin, D., Friedman, S., Supowit, K.: Delaunay graphs are almost as good as complete graphs. Discret. Comput. Geom. **5**(4), 399–407 (1990)

Drysdale, R., Rote, G., Aichholzer, O.: A simple linear time greedy triangulation algorithm for uniformly distributed points. In: Report IIG-408, Institutes for Information Processing, Technische Universit at Graz (1995)

Edmonds, J.: Matroids and the greedy algorithm. Math. Progr. **1**(1), 127–136 (1971)

Eppstein, D.: Dilation-free planar graphs. http://www.ics.uci.edu/~eppstein/junkyard/dilation-free/ (1997). Accessed 4 Feb 2014

Eppstein, D.: Spanning trees and spanners. In: Sack, J.R., Urrutia, J. (eds.) Handbook of Computational Geometry, pp. 425–461. Elsevier, Amsterdam (2000)

Eppstein, D., Wortman, K.: Minimum dilation stars. In: Proceedings of the 21st ACM Symposium on Computational Geometry, pp. 321–326, cs.CG/0412025. ACM (2005)
Giannopoulos, P., Knauer, C., Marx, D.: Minimum-dilation tour (and path) is NP-hard. In: EWCG 07: Proceedings of the 23rd European Workshop on Computational Geometry, pp. 18–21 (2007)
Keil, J., Gutwin, C.: The delaunay triangulation closely approximates the complete euclidean graph. In: Dehne, F., Sack, J.R., Santoro, N. (eds.) LNCS, vol. 382, pp. 47–56. Springer, Berlin (1989)
Keil, J., Gutwin, C.: Classes of graphs which approximate the complete euclidean graph. Discret. Comput. Geom. **7**(1), 13–28 (1992)
Kirkpatrick, S., Gelatt, C., Vecchi, M.: Optimization by simulated annealing. Science **220**, 671–680 (1983)
Klein, A.: Effiziente berechnung einer dilationsminimalen triangulierung. PhD thesis (2006a)
Klein, A.: Minimum dilation triangulation. Geometry Lab. http://www.geometrylab.de/ (2006b). Accessed 4 Feb 2014
Klein, R., Kutz, M.: Computing geometric minimum-dilation graphs is NP-hard. In: Kaufmann, M., Wagner, D. (eds.) Graph Drawing, pp. 556–567. Springer, Berlin (2006)
Knauer, C., Mulzer, W.: An exclusion region for minimum dilation triangulations. In: EuroCG, Technische Universiteit Eindhoven, pp. 33–36 (2005a)
Knauer, C., Mulzer, W.: Minimum dilation triangulations. In: Freie Universitt Berlin, Fachbereich Mathematik und Informatik, Technical Report (2005b)
Kolingerová, I., Ferko, A.: Multicriteria-optimized triangulations. Vis. Comput. **17**(6), 380–395 (2001)
Lourenco, H., Martin, O., Stützle, T.: Iterated Local Search, pp. 321–353. Kluwer Academic Publishers, Norwell (2002)
Markatou, M., Tian, H., Biswas, S., Hripcsak, G.: Analysis of variance of cross-validation estimators of the generalization error. J. Mach. Learn. Res. **6**, 1127–1168 (2005)
Martin, O., Otto, S., Felten, E.: Large-step Markov chains for the traveling salesman problem. Complex Syst. **5**, 299–326 (1991)
Michalewicz, Z., Fogel, D.: How to Solve It: Modern Heuristics. Springer, Berlin (2004)
Narasimhan, G., Smid, M.: Approximating the stretch factor of euclidean graphs. SIAM J. Comput. **30**, 2000 (1999)
Narasimhan, G., Smid, M.: Geometric Spanner Networks. Cambridge University Press, New York (2007)
Nielson, G.: Tools for triangulations and tetrahedrizations. In: Scientific Visualization, Overviews, Methodologies, and Techniques, IEEE Computer Society, pp. 429–525 (1997)
Papadimitriou, C.: The complexity of combinatorial optimization problems. PhD thesis, Princeton, NJ, aAI7704795 (1976)
Vite-Silva, I., Cruz-Cortés, N., Toscano-Pulido, G., Fraga, L.: Optimal triangulation in 3d computer vision using a multi-objective evolutionary algorithm. In: Giacobini, M. (ed.) Applications of Evolutionary Computing. Lecture Notes in Computer Science, vol. 4448, pp. 330–339. Springer, Berlin (2007)
Xia, G.: Improved upper bound on the stretch factor of delaunay triangulations. In: Proceedings of the 27th Annual ACM Symposium on Computational Geometry, SoCG '11, pp. 264–273. ACM, New York (2011)
Xia, G., Zhang, L.: Improved lower bound for the stretch factor of delaunay triangulations manuscript. Poster in fwcg10: 20th Annual Fall Workshop on Computational Geometry (2010)
Zhou, H., Wu, H., Xia, S., Jin, M., Ding, N.: A distributed triangulation algorithm for wireless sensor networks on 2d and 3d surface. In: INFOCOM, 2011 Proceedings IEEE, pp. 1053–1061 (2011)