Minimum Dilation Triangulation: Reaching Optimality Efficiently

Alex F. Brandt* Miguel M. Gaiowski* Cid C. de Souza* Pedro J. de Rezende*

Abstract

Let $G(P) = (P, E)$ be the geometric graph associated to a given set P of points in the plane, i.e., the complete graph whose vertex set is P and whose edges have weights defined by the Euclidean distance between their endpoints. On a planar triangulation of P, $T \subseteq G(P)$, the dilation of a pair of points $i, j \in P$ is the ratio between the length of the shortest path between i and j in T and their Euclidean distance. The dilation of T is the maximum dilation between the pairs of points in P. The *Minimum Dilation Triangulation Problem* (MDTP) asks for a triangulation of P with smallest dilation. We developed an exact algorithm that combines an efficient heuristic, a set of preprocessing routines that exploit geometric properties of the problem (using the primal bound given by the heuristic) and an integer programming model for the MDTP. We report on computational experiments in which, for the first time, instances of up to 70 points have been solved to proven optimality. The impact of the heuristic and the preprocessing on the algorithm's performance are demonstrated through a careful analysis of the results.

1 Introduction

Given a set P of points in the plane, consider the complete graph $G(P) = (P, E)$ whose vertex set is P and whose edges have weights defined by the Euclidean distance between their endpoints.

Dilation. On any spanning subgraph G' of $G(P)$, the *dilation* $\rho_{G'}(i, j)$ of a given pair of distinct vertices $i, j \in P$ is the ratio between the length of a shortest path connecting them in G', $\pi_{G'}(i, j)$, and their Euclidean distance, d_{ij}: $\frac{|\pi_{G'}(i,j)|}{d_{ij}}$. Non connected pairs of vertices are said to have infinite dilation. The *dilation*[1] *of* G' is the maximum dilation between pairs of points in P:

$$\rho(G') = \max_{\substack{i,j \in P \\ i \neq j}} \left(\frac{|\pi_{G'}(i,j)|}{d_{ij}} \right) \tag{1}$$

The *Minimum Dilation Triangulation Problem* (MDTP) asks for a triangulation of P with minimum dilation. While the complexity of this problem remains open, it is believed to be $\mathbb{NP}$-hard. This intuition derives from studying the following closely related problems. Let $t > 1$ be a real number. A *t-spanner* is a spanning subgraph of $G(P)$ with dilation at most t. Klein and Kutz, in [8], show that computing a t-spanner with the minimum number of edges is $\mathbb{NP}$-hard. On the other hand, computing the spanning tree with minimum dilation, even when edge crosses are not allowed, is also $\mathbb{NP}$-hard, as established by Cheong *et al.*, in [3]. Further complexity results are given in [6].

Related work. As far as settling tight bounds for the dilation of triangulations, an interesting result was shown by Xia, in [15], who proved that the Delaunay triangulation has a stretch factor of at most 1.998 for any point set in the plane. While no one has been able to guarantee a better upper bound for the minimum dilation of triangulations, a lower bound on the worst case might provide some useful information. According to [11], every triangulation of the vertices of a regular 21-gon has spanning ratio at least $\sqrt{2.005367532} \approx 1.41611$. It is also of interest to investigate lower bounds for the Delaunay triangulation given its significance. Bose and Devroye, in [1], construct a point set in convex position, for which the Delaunay triangulation has dilation at least 1.581. For points not in convex position, the corresponding lower bound is 1.5932 [16].

In regard to efforts to solve the MDTP, Mulzer presents, in [11], a $1 + O(1/\sqrt{\log n})$ approximation algorithm for the MDTP for the vertices of an n-gon. An interesting exclusion region rule, i.e., a local property that guarantees that an edge cannot be part of a minimum dilation triangulation is presented in [9].

Klein, in [7], developed an applet that implements two algorithms to enumerate all possible triangulations in order to find one of minimum dilation. Some strategies are available to be applied as tools to reduce the set of candidate triangulations and speed up the search. One of the available reductions is based on the exclusion region mentioned above. Although this applet is a nice visualization tool, the fact that it produces an (almost) complete enumeration limits its applicability to no more than a very small number of points. Recently, some metaheuristics and a simulated annealing algorithm were presented in [4] for solving the MDTP.

Motivation and Our Contributions. Extensive studies on general triangulations have been prompted by applications to areas as diverse as wireless networks, robotics and computer graphics (cf. [12]). Although significant effort has been made to establish theoretical results on geometric structures akin to triangulations, very little is known in regard to (sub) optimal solutions

*Institute of Computing, University of Campinas, Campinas, Brazil, {lexbrandt|miggaiowski}@gmail.com, {rezende|cid}@ic.unicamp.br

[1] *Stretch factor*, *spanning ratio* and *distortion* are alternative nomenclature for dilation (see [12] for a historical context).

of the MDTP. Our study is, therefore, motivated by the unknown complexity of MDTP and the belief that the search for optimal solutions and their properties is likely to lead to a better understanding of the problem and to provide insight for better algorithms.

This work presets an exact algorithm for the MDTP that is shown to be very efficient in practice. It relies on a powerful metaheuristic, geometric properties of the problem and a new mixed integer linear program (MILP) model. To the best of our knowledge, this is the first formulation for the MDTP in the literature. We show, through experimental tests, that the high quality solutions produced by the metaheuristic are essential to the efficiency of the method. The heuristic solution allows state-of-the-art MILP solvers to prune the enumeration tree of the branch-and-bound algorithm early and provides an incumbent solution to the solver.

However, the most expressive accomplishment comes from exploring the primal bound to exploit the geometry of the problem in order to critically reduce the model size using preprocessing routines. The results and analysis of several tests over a benchmark featuring 210 instances of up to 70 points are presented to assess the performance of our algorithm. Also, to allow for future comparisons with other works, our benchmark is made publicly available at [2]. Finally, we should notice that most MILP solvers do not support radical arithmetic. Thus, we consider the distances between points, rounded to a fixed precision, as part of the input and optimality is proven relative to them.

Text Organization. The next three sections are dedicated to describing each of the fundamental blocks of our exact algorithm: the problem reduction by preprocessing routines, the GRASP metaheuristic and the MILP formulation for the MDTP. In Section 5, the details of the benchmark, the computational environment and the tests performed are described, followed by their results and a detailed analysis. Section 6 summarizes our progress and points to future directions of our work.

2 Metaheuristic GRASP

Greedy Randomized Adaptive Search Procedure (GRASP) is a metaheuristic widely applied to combinatorial optimization problems. Since it is reported to produce high quality solutions within a short computing time [13] we employ it to find good primal bounds for the MDTP. We assume that the reader is familiar with the workings of a basic GRASP algorithm, which consists of a construction phase and a local search phase, repeated for an *a priori* fixed number of iterations, before the best solution found is returned. To explain its use for the MDTP, we limit ourselves to describing these two phases.

Despite its name, our construction phase resembles a *"destruction & repair"* procedure applied to the Delaunay triangulation (DT). Recall that, generically, for a minimization problem, the goal of the construction phase is to compute along all the iterations a set of (preferably) distinct solutions, all of which having small costs. In the context of GRASP, the diversity of the resulting set is obtained through some randomization of a greedy routine. In the MDTP case, our experiments showed that the DT often has a dilation that is very close to the optimum. So, at each iteration of GRASP, we generate new triangulations starting from the DT. The process consists of removing from it k edges at random, one at a time, followed by a randomized greedy strategy that iteratively inserts another k edges to the structure creating a new triangulation. Let us look into this process in more detail.

Let G' denote the graph associated to the current partial solution, initially representing the DT. Moreover, suppose that, throughout the iterations, for any two points i and j in P, the algorithm keeps track of the set of points in $P - \{i, j\}$ that belong to the ellipse of foci i and j and major axis equal to d_{ij} times the best primal bound available. We can now establish conditions for an edge of the DT to be a candidate for removal. Firstly, the removal must not disconnect G'. Secondly, the edge cannot be part of the convex hull of P. Lastly, the edge must not have been fixed by the *empty ellipse case* of the *edge fixing routine* presented later, in Section 4.

After k such edges have been removed from the triangulation, the next step is to insert k new edges into the current partial solution so as to obtain a new triangulation. The set of candidates for insertion is composed of the diagonals of all bounded faces of G'. The dilation resulting from the addition of each of the edges in this set to the current partial solution is computed using the algorithm discussed in Farshi et al. [5]. Finally, we introduce further randomization to this process by selecting, according to a uniform distribution, an edge that leads to a dilation that is at most α times larger than the best possible one. In our implementation of the construction phase, the values of k and α were set to $n/2$ and 0.8, respectively.

We now turn our attention to the local search phase. Consider two triangles abc and dbc in a triangulation T that share an edge bc. If the union of these triangles forms a convex polygon, we say that bc is *flippable*. A *flip* operation on T consists of replacing edge bc by ad, giving rise to a new triangulation. We say that two triangulations are *neighbors* if one can be obtained from the other by a single flip. The working of our local search phase should now be clear. Given the current triangulation T, the procedure traverses the edges of T checking whether they are flippable and, when this is the case, it computes the dilation of the triangulation T' obtained by flipping that edge[2]. In our implementation, we adopt the *first improvement strategy*, i.e., whenever the dilation is reduced, T is updated with T', and the local search is restarted with this new solution. The process is halted when T corresponds to a local minimum with respect to flip neighborhood.

[2]Note that there is a unique flip associated to a flippable edge.

3 Mixed integer formulation

Let a, b be any two vertices of P, from $G(P) = (P, E)$. Define a directed graph $\vec{G}_{ab} = (P, A_{ab})$, where, for every edge $\{i, j\} \in E$, both arcs (i, j) and (j, i) are in A_{ab}.

We now describe how we use these graphs to build a *directed multicommodity flow formulation* to model the MDTP. Given a pair of vertices $a, b \in P$, pick an arbitrary order, say, (a, b), and consider that an exclusive unit of flow (commodity) is sent from a to b. This unit of flow is allowed to go through any *directed arc* (i, j) in $\vec{G}_{ab}$. The binary (flow) variable x_{ij}^{ab} for an arc (i, j) of $\vec{G}_{ab}$ indicates whether the path from a to b uses the edge $\{i, j\}$ of G. In this way, we can identify a path on the triangulation for each pair of vertices, and determine its length by simply adding the costs of its edges. Also, notice that the edge $\{i, j\}$ is in the optimal planar triangulation computed by the model if and only if the variable x_{ij}^{ij} is set to one.

From Euler's formula, it follows that any triangulation of a set P of n points in the plane has $\phi(P) = 3n - 3 - h$ edges, where h is the number of vertices of the convex hull of P. Conversely, any set of $\phi(P)$ edges from the geometric graph that forms a *planar* graph determines a triangulation of P. Clearly, $\phi(P)$ can be efficiently computed *a priori*. Then, the mixed integer linear program below is a formulation for the MDTP:

$$\min \ \rho \tag{2}$$

$$\sum_{\{i,j\} \in E} x_{ij}^{ij} = \phi(P), \tag{3}$$

$$x_{ij}^{ij} + x_{kl}^{kl} \leq 1, \qquad \forall \{i, j\}, \{k, l\} \in E, \ \overline{ij} \cap \overline{kl} \neq \emptyset, \tag{4}$$

$$x_{ij}^{ab} + x_{ji}^{ab} - x_{ij}^{ij} \leq 0, \qquad \forall \{i, j\} \in E, \ \forall a, b \in P, \tag{5}$$

$$\sum_{i \in P \setminus \{j\}} x_{ij}^{ab} - \sum_{i \in P \setminus \{j\}} x_{ji}^{ab} = d_j, \qquad \forall j, a, b \in P, \tag{6}$$

$$\sum_{(i,j) \in A} \frac{d_{ij}}{d_{ab}} x_{ij}^{ab} - \rho \leq 0, \qquad \forall a, b \in P, \tag{7}$$

where in (6), d_j is set to 1, -1 or 0, when $j = a$, $j = b$ and $a \neq j \neq b$, respectively.

Inequalities (3) ensure that the number of edges included is necessary and sufficient to form a valid triangulation. Constraints (4) prevent edge crossings. The flow conservation constraints (6), enforce every commodity to flow through a path from the source node to the sink. This allows the calculation of the path length and a bound on its dilation with inequalities (7). Coupling the triangulation and the paths sub-models, constraints (5) force the necessary edges to connect each pair of vertices in the triangulation.

It is important to note that the the path joining a pair of vertices in an arbitrary integer solution might not be the shortest or even simple, since there is no restriction on the degrees of the vertices. The path connecting a pair of maximum dilation in an optimal solution must be simple and the shortest possible one in the triangulation, otherwise, it could be shortened and the dilation would be reduced.

4 Problem reduction

The major drawback of the MILP formulation given in Section 3 is that its size grows too fast with the number of points in P. So, it is important to find ways to eliminate variables and even constraints from this formulation, in order to facilitate the computation of the model. The routines described in this section are designed to achieve this goal. They all assume that a triangulation is known whose dilation is given by $\rho^{\mathrm{H}} \geq 1.0$.

Recall that the MILP model contains two sets of binary variables. Let a, b, i and j be four distinct vertices in $G(P)$. The first set describes which edges are present in an optimal triangulation T^* and correspond to variables of the form x_{ab}^{ab}. The second set of variables identifies the arcs that represent a path joining a and b in T^* and having dilation not greater than the optimum ρ. These are the variables of the form x_{ij}^{ab}, which may be interpreted in terms of the quantity of commodity ab that flows in arc ij. Essentially, given that $\rho \leq \rho^{\mathrm{H}}$, the preprocessing routines fix some of these binary variables to either 0 or 1 in an optimal solution and propagate this information to other x variables in an attempt to fix them, too. Initially, we identify four basic scenarios where this process occurs.

Firstly, consider what happens when a variable x_{ij}^{ij} is set to 0, meaning that the edge $\{i, j\}$ is discarded from an optimal triangulation. Due to constraints (5), this setting also forces all flow variables of the form x_{ij}^{ab} and x_{ji}^{ab}, for any commodity ab, to become null. Before we discuss the opposite case, where the edge $\{i, j\}$ is known to be in the optimal solution, consider what occurs when the arcs of a path connecting two vertices a and b are forced to take part of the optimal triangulation. Clearly, in this case, all flow variables of the form x_{ij}^{ab} associated to arcs not belonging to the path must be set to 0.

Now, we analyze the case where the flow x_{ij}^{ab} in an arc (i, j) is required to take value 1, i.e., we impose that the arc be on the path from a to b. By constraints (5), the commodity must flow in a single direction between any pair of vertices, therefore, the opposite arc (j, i) must be discarded from the optimal solution. As also enforced by these constraints, an arc can be used only if the respective edge is part of the solution. Thus, the corresponding edge variable x_{ij}^{ij} must be set to 1.

Finally, we examine the case where the edge $\{i, j\}$ is identified as being part of the solution. This gives rise to two types of propagation. The first comes from the observation that the path between i and j is completely defined and the associated propagation described above applies. The planarity restrictions (4) provide the second model reduction. Once an edge is included in the solution, no other edge crossing it, except possibly on

its extremities, can be used. As a consequence, the variables corresponding to these edges are set to 0 and, as seen before, this leads to further propagations.

We now describe the routines we use to identify the edges and arcs whose variables can be set to 1 or 0, triggering the propagations discussed above to expand the model reduction.

The ellipse-based elimination routine. Let T be a planar triangulation and suppose u is a vertex on the path $\pi_T(a,b)$, $a \neq u \neq b$, i.e., $\pi_T(a,b) = \pi_T(a,u) \cup \pi_T(u,b)$. The dilation of the pair $\{a,b\}$ in T is given by $\rho_T(a,b) = \frac{|\pi_T(a,b)|}{d_{ab}}$. From the triangle inequality, $\pi_T(a,u) \geq d_{au}$ and $\pi_T(u,b) \geq d_{ub}$ and $\rho_T(a,b) \geq \frac{d_{au}+d_{ub}}{d_{ab}}$. Therefore, if T has stretch factor at most ρ^{H}, the following constraint must be satisfied:

$$\frac{d_{au} + d_{ub}}{d_{ab}} \leq \rho_T(a,b) \leq \rho^{\mathrm{H}} \ , \tag{8}$$

which means that u must be in the ellipse with foci a and b having a major axis of length $\rho^{\mathrm{H}} \cdot d_{ab}$. In other words, given an upper bound ρ^{H} for the stretch factor and a pair of vertices a and b, only the points in the ellipse defined by (8) can be on the path joining a and b in a solution with dilation not exceeding ρ^{H}. This idea has also appeared in [8] and [10].

It is important to note consequences of these facts for the model reduction. Observe that all $O(n)$ arcs entering or leaving any discarded vertex are precluded from being on the path between a and b and may, therefore, be removed from the model, i.e., the corresponding variables may be set to 0.

The edge fixing routine. Our program applies two strategies to identify edges that must be present in an optimal triangulation. The first takes advantage of the maximality property of planar triangulations. Basically, if all the edges crossing the edge $\{i,j\}$ have been dropped from being in the optimal triangulation, $\{i,j\}$ must be part of it. Naturally, the propagations discussed earlier may result in further reductions.

The other strategy consists in examining whether the settings done so far during the process constrain the connection between two vertices to a unique path. This can be done by a simple procedure and, in the positive case, we apply the correspondent propagations described previously.

An important remark concerns the fixing of an edge caused by the arguments used in the ellipse-based elimination routine. To see that, consider the case where no vertices remain in the ellipse other than its foci, called the *empty ellipse case*. Obviously, the only way left to link the vertices representing the foci is through the direct arc joining them. Hence, this variable is set to 1.

5 Computational Results

This section presents the computation tests we performed to assess the proposed algorithm. The benchmark of instances and the computing environment are described in the next paragraphs. Later, we specify each of the configurations of the solver we ran, and analyze the results they yielded.

Instances. Our benchmark contains 210 instances, in total, divided into 7 groups of 30. Each group is formed by instances of a fixed size n, for $n \in \{10, 20, \ldots, 70\}$, whose elements are points with coordinates generated uniformly in a 10×10 square and rounded to the 6th decimal place. Since most state-of-the-art MILP solvers do not support arithmetic with radicals, we calculated, *a priori*, the distances between every two points, rounded to the 6th decimal place, *consider these values as part of the input*, and prove optimality relative to them. Hence, we are actually solving a rounded version of the MDTP. The entire benchmark is available at [2].

Computing Environment. The results presented in this section were obtained using three identical machines featuring Intel® Xeon® CPU E3-1230 V2 @ 3.30GHz (4 cores and 8 threads) processors and 32GB of RAM while running Ubuntu 12.04, g++ 4.6.3 (c++0x standard) and IBM® ILOG® CPLEX® Optimization Studio 12.5.1. The solver was allowed to use all 8 threads simultaneously for at most 30 minutes per test. The CGAL [14] and BOOST libraries provided, respectively, useful tools for geometric computation and graph representation.

Results. We begin by assessing the solutions delivered by our GRASP. The quality of such solutions is crucial for the performance of the exact algorithm as it affects the reduction of the MILP model and also the primal information given as input to the solver.

As mentioned in Section 2, GRASP uses the Delaunay triangulation as a starting point to produce random triangulations. The natural question is: on average, how close is the dilation (ρ^{DT}) of the DT to the optimum (ρ^*)? For an insight on this issue, consider the data in Table 1. There, we give the minimum, average, standard deviation and maximum values of the percentage gap between ρ^{DT} and ρ^* for each group of instances in our benchmark. The last column exhibits how many of the DTs (out of 30) are optimal, i.e., satisfy $\rho^{\mathrm{DT}} = \rho^*$.

Table 1: Statistics for Delaunay triangulation.

Size	Delaunay Triang. Gap (%)				# Opt
	Min	Avg	Std Dev	Max	Sols
10	0.00	1.66	3.08	11.88	17
20	0.00	1.50	1.99	5.79	12
30	0.00	1.63	2.09	6.61	13
40	0.00	1.12	1.53	6.28	12
50	0.00	1.65	2.12	7.96	12
60	0.00	2.12	1.90	5.98	7
70	0.00	1.76	2.02	6.95	8

As it can be seen, ρ^{DT} is usually very close to ρ^* and, therefore, a good upper bound for preprocessing. As a consequence, DT becomes a natural candidate for an incumbent solution for the MILP solver. From these results, one may wonder whether a more complicated and time consuming heuristic such as GRASP is needed.

To shed some light on this issue, we note that, with GRASP, all optima have been found whereas, for the DT, the fraction of optima found always stayed below 57% and, even worse, it decreases with the size of the instances. Hence, the advantage of using GRASP instead of the DT to produce primal information for the MILP solver is evident. Besides, GRASP clearly emerges as an excellent way to compute high quality MDTP solutions.

One of the main purposes of generating primal information is to obtain reductions on the MILP model to sizes that are manageable by the solver. Due to space limitations, we constrain our discussion about this aspect to the following observations. The percentage of variables removed from instances with 70 points reached 95.7% on average. Although at first glance this statistics looks impressive, the 4.3% remaining variables amounts to 505,096 of them, which still constitutes a huge model. It is also noticeable that, while the average number of removed variables increases with the number of points, the percentage of mandatory edges over the final number in the solution decreases from 90%, for instances with 10 points, to 60% for the ones with 70 vertices.

Next, we evaluate how much each piece of primal information contributes to improve the efficiency of our exact algorithm. To this end, six different variants of the MILP solver were tested. These variants are characterized by three aspects: *(i)* the method employed to produce a valid primal solution, if any; *(ii)* the application or not of the geometric preprocessing routines described in Section 4 and *(iii)* the usage or not of the primal information by the solver. Accordingly, the solver variants are identified by triples of characters, one for each of the aforementioned aspects. The first character identifies the primal method, where `G` stands for the GRASP solution, `D` for the DT and ␣ specifies that no primal solution is computed. When the preprocessing routines are applied, the second character is `P`, otherwise it is ␣. Finally, the third character is `S` whenever the MILP solver is fed with the primal solution described in the first element of the triple. Analogously, the ␣ character means that no solution is given to the solver.

It is well known that primal solutions and bounds may be used to speed up the running time of enumeration algorithms. In the case of MILP, the availability of such data allows state-of-the-art solvers, such as CPLEX, not only to prune the search tree but also to apply algebraic preprocessing routines to reduce the formulation. Two variants used in our experiments directly benefit from these ideas, namely, `G␣S` and `D␣S`. In contrast, we also tested the stand alone variant ␣␣␣ in which CPLEX receives nothing but the complete model of Section 3. By comparing ␣␣␣, `G␣S` and `D␣S`, we can measure to what extent the solver takes advantage of the knowledge brought by the GRASP solution and the DT. Notice that in `G␣S` and `D␣S`, only algebraic preprocessing is performed by CPLEX. For a better appreciation of our contribution, these variants must be compared to others where our geometric preprocessing is executed.

Table 2: Statistics for six variations of the MILP solver.

Size	Var	# Opt Sols	Avg Exec Time (s)	# Feas Sols	% Avg Gap	Wins
10	GPS	30	0.12	0	–	0
	GP␣	30	0.13	0	–	0
	DPS	30	0.03	0	–	30
	G␣S	30	0.39	0	–	0
	D␣S	30	0.29	0	–	0
	␣␣␣	30	0.36	0	–	0
20	GPS	30	1.18	0	–	4
	GP␣	30	1.25	0	–	0
	DPS	30	0.84	0	–	26
	G␣S	30	34.94	0	–	0
	D␣S	30	43.22	0	–	0
	␣␣␣	30	148.74	0	–	0
30	GPS	30	5.86	0	–	6
	GP␣	30	6.35	0	–	3
	DPS	30	6.17	0	–	21
	G␣S	7	276.06	23	7.3	0
	D␣S	2	306.13	28	8.3	0
	␣␣␣	0	–	3	98.5	0
40	GPS	30	24.26	0	–	10
	GP␣	30	29.36	0	–	8
	DPS	30	54.42	0	–	12
	G␣S	0	–	30	24.2	0
	D␣S	0	–	30	25.1	0
50	GPS	30	73.13	0	–	11
	GP␣	30	112.50	0	–	7
	DPS	30	207.24	0	–	12
60	GPS	30	244.42	0	–	18
	GP␣	29	584.75	1	75.7	5
	DPS	22	682.01	8	4.6	7
70	GPS	30	655.79	0	–	21
	GP␣	15	1278.36	9	59.3	2
	DPS	14	879.60	16	3.7	7

Hence, the remaining variants chosen to be part of our experiments are `GPS`, `DPS` and `GP␣`.

The results obtained by all MILP configurations are summarized in Table 2. A solver variant (second column) appears in a row corresponding to an instance size (first column) until it cannot prove optimality for any instance of this size (i.e., the value in the third column is null). The fourth column shows the average running time taken by the respective variant to achieve the optimal solutions reported in the previous cell (in this same row). The fifth column shows the number of instances for which, although optimality was not proved, the solver found a feasible solution (possibly the incumbent one). The average gaps of the latter solutions are displayed in the sixth column. The seventh column gives the total number of instances where the variant proved optimality quicker than its competitors.

Confirming the common knowledge, the configurations `G␣S` and `D␣S` largely outperform ␣␣␣. So, primal data are crucial to improve the solver's efficiency. As we have already seen in the beginning of this section, the dilation of the GRASP solution is typically smaller than ρ^{DT}. The data reported in the fourth column reveal how much the algorithm's running time benefits from the better information generated by GRASP.

To further emphasize the importance of our geometric processing we examine the results obtained by GP␣ and G␣S. Of course, the variant GP␣ is not really interesting in practice as it is not reasonable to produce primal information and not make it available to the solver. The GPS would be the obvious choice in this case. However, by comparing GP␣ and G␣S, one immediately sees that the preprocessing routines are more relevant to the algorithm's efficiency than the initial solution itself. In fact, GP␣ is able to solve instances which are at least twice as large as those treated by G␣S. Another evidence of the strength of the preprocessing appears when we observe that GP␣ solves almost as many instances as GPS.

The last lines on Table 2 suggest that GPS is the winner among all solver variants we experimented with, while DPS ranks second (recall that we consider GP␣ as an anomalous version of GPS). However, the results for the smaller instances demand a better analysis. Again, let n denote the instance size. Recall that, for the instance sizes considered here, the time consumed by CGAL to compute the DT is negligible. In this way, the time taken by GRASP, which also includes the computation of the DT, does not pay off for instances with 20 points or less. However, as the instances get larger ($n \geq 30$), the average running times of GPS are always smaller than those of DPS. The advantage of using GRASP becomes even more evident when we compare the numbers of optima found and of wins of GPS and DPS.

6 Conclusions and Future Directions

In this paper, we presented an association of three components that yields an efficient algorithm to compute exact solutions for the Minimum Dilation Triangulation Problem. They are: a GRASP metaheuristic, a powerful geometric preprocessing and a mixed integer linear program model. Computational tests showed that our method is a viable option to solve instances with up to 70 points to proven optimality in no more than 30 minutes on a standard computer. Essentially, this performance was due both to exploring relevant geometric information that culminates in a significant downsizing of the MILP model and to the employment of GRASP.

Improvements to the method are expected from future investigation of the following issues: *(a)* the use of GRASP and geometric preprocessing in all nodes of the enumeration tree and not only in the root node; *(b)* the customization of the choice of branch variable based on geometric properties; and *(c)* the use of an alternative MILP to model planar triangulations.

Acknowledgments. This research was supported by grants from CNPq #477692/2012-5, #302804/2010-2, #139107/2012-6, FAPESP #2012/17965-6 and FAEPEX/ UNICAMP.

References

[1] P. Bose, L. Devroye, M. Löffler, J. Snoeyink, and V. Verma. Almost all Delaunay triangulations have stretch factor greater than $\pi/2$. *Computational Geometry*, 44(2):121 – 127, 2011.

[2] A. F. Brandt, M. de M. Gaiowski, P. J. de Rezende, and C. C. de Souza. The Minimum Dilation Problem project, 2014. *www.ic.unicamp.br/∼cid/ Problem-instances/Dilation*.

[3] O. Cheong, H. Haverkort, and M. Lee. Computing a minimum-dilation spanning tree is NP-hard. *Comput. Geom.*, 41(3):188–205, Nov. 2008.

[4] M. Dorzán, M. Leguizamón, E. Mezura-Montes, and G. Hernández-Peñalver. Approximated algorithms for the minimum dilation triangulation problem. *J. of Heuristics*, 20(2):189–209, 2014.

[5] M. Farshi and J. Gudmundsson. Experimental study of geometric t-spanners. *J. Exp. Algorithmics*, 14:3:1.3–3:1.39, Jan. 2009.

[6] P. Giannopoulos, R. Klein, C. Knauer, M. Kutz, and D. Marx. Computing geometric minimum-dilation graphs is NP-hard. *Int. J. Comput. Geometry Appl.*, 20(2):147–173, 2010.

[7] A. Klein. Effiziente Berechnung einer dilationsminimalen Triangulierung, 2006. MSc., Univ. Bonn.

[8] R. Klein and M. Kutz. Computing geometric minimum-dilation graphs is NP-hard. In M. Kaufmann and D. Wagner, editors, *Graph Drawing*, volume 4372 of *LNCS*, pages 196–207. Springer, 2007.

[9] C. Knauer and W. Mulzer. An exclusion region for the minimum dilation triangulation. In *Proc. of the 21st European Workshop on Comput. Geometry*, pages 33–36, Eindhoven, Holland, 2005.

[10] C. Knauer and W. Mulzer. Minimum dilation triangulations, 2005. Tech. Rep., Freie Univ. Berlin.

[11] W. J. H. Mulzer. Minimum dilation triangulations for the regular n-gon, 2004. MSc., Fachbereich Mathematik und Informatik, Freie Univ. Berlin.

[12] G. Narasimhan and M. Smid. *Geometric Spanner Networks*. Cambridge Univ. Press, New York, 2007.

[13] M. Resende and C. Ribeiro. Greedy randomized adaptive search procedures: Advances, hybridizations, and applications. In *Handbook of Metaheuristics*, Int. Series in Oper. Res. & Man. Science, pages vol 57, 219–249. Springer, 2009.

[14] The CGAL Project. *CGAL User and Reference Manual*. CGAL Editorial Board, 4.4 edition, 2014.

[15] G. Xia. Improved upper bound on the stretch factor of Delaunay triangulations. In *Proc. of the 27th Annual Symposium on Computational Geometry*, SoCG '11, pages 264–273, New York, 2011.

[16] G. Xia and L. Zhang. Toward the Tight Bound of the Stretch Factor of Delaunay Triangulations. In *Proc. of the 23rd CCCG*, Toronto, 2011.