Approaching Minimum Area Polygonization *

Maria Teresa Taranilla[1], Edilma Olinda Gagliardi[1], and Gregorio Hernández Peñalver[2]

[1] Facultad de Ciencias Físico, Matemáticas y Naturales
Universidad Nacional de San Luis, Argentina
{tarani,oli}@unsl.edu.ar
[2] Facultad de Informática
Universidad Politécnica de Madrid, España
{gregorio}@fi.upm.es

Abstract. The problem of finding a minimum area polygonization for a given set of points in the plane, Minimum Area Polygonization (MAP) is NP-hard. Due to the complexity of the problem we aim at the development of algorithms to obtain approximate solutions. In this work, we suggest different strategies in order to minimize the polygonization area. We propose algorithms to search for approximate solutions for MAP problem. We present an experimental study for a set of instances for MAP problem.

Keywords: Simple Polygons, Minimum Area Polygonization, Computational Geometry, Geometric Structures Optimization

1 Introduction

Computational Geometry offers a formal framework to give solutions to geometric problems involving geometric structures. Geometric structures play important roles in several application areas, e.g., image processing, pattern recognition, computer graphics, geographic information system, computer-aided design and manufacturing (CAD/CAM), among others. In further relation to geometric structures, Computational Geometry includes NP-hard problems as well as problems for which efficient algorithms for their solution have not been found. In either case it is necessary, however, to provide efficient techniques to readily obtain good quality solutions. In particular, optimization problems related to some particular geometric structures, such as *polygonizations* are interesting for researchers due to their use in many fields of applications, e.g., pattern recognition and image reconstruction [1],[4].
A *polygonal chain* is an ordered sequence of points $p_0, p_1, \ldots, p_{n-1}$ with $n>3$ together with the set of line segments $e_0=\overline{p_0p_1}$, $e_1=\overline{p_1p_2}, \ldots, e_n=\overline{p_{n-2}p_{n-1}}$ denominated edges. The polygonal chain is *closed* if the first and the last point

* Supported by Research Project Tecnologías Avanzadas de Bases de Datos 22-F014, Universidad Nacional de San Luis, Argentina and Research Project MTM2008-05043, Ministerio de Ciencia e Innovación, España.

are connected by a line segment. The polygonal chain is *simple* if the only intersection of edges are those at common endpoints of consecutive edges. A simple closed polygonal chain divides the plane in two regions, an unbounded one denominated the exterior region, and a bounded one, the interior. A *simple polygon* is a simple closed polygonal chain together with its interior. A *simple polygonization* for a given set of points is a simple polygon whose vertices are precisely that set of points. A *simple polygonization* for a given set of points is a simple polygon whose vertices are precisely that set of points.
For a given set of points in the plane, a simple polygonization always exists unless all the points happen to be collinear. In general, the number of different simple polygonizations for a set of n points is known to be exponential in n [7]. However, the number of polygonizations to be obtained can be affected by the geometric properties of the set of points to be polygonized, e.g. if the points in the set are in convex position, the number of polygonization is only one, the convex hull.
A related problem to polygonizations is the paradigm of combinatorial optimization, the *Traveling Salesman Problem*(TSP) [9]. In the geometric version, the problem is to find the shorter closed route connecting a given set of points in the plane. This route must be a simple polygonization with shortest perimeter. While classic Euclidean TSP deals with finding a polygonization with the shorter perimeter, our concern is to find a simple polygonization minimizing another basic geometric measure: the enclosed area. This problem is denominated Minimum Area Polygonization (MAP). It is defined as follows.

Minimum Area Polygonization (MAP): *Given a finite set S of points in the Euclidean plane, find the simple polygonization with minimum area among all the simple polygonizations with vertices in S.*

In 2000, Fekete [8] proved that to find a minimum or maximum area polygonization for a given set of points in the plane is NP-hard. In the same article, it has been demonstrated: *i)* for a set P of n points a simple polygon of more than half the area of the convex hull of P CH(P), can be found in time O(n *log* n), and *ii)* for $0 < \varepsilon < 1/3$ it is NP-hard to decide whether a set P of points allows a simple polygon of area of more than $(2/3 + \varepsilon)$ CH(P).
In this work, we focused our attention in finding a simple polygonization with minimum area for a given set of points in the plane. Due to the complexity of the problem we aim at the development of algorithms to obtain approximate solutions. We propose different strategies in order to minimize the polygonization area. This paper is organized as follows. In section 2, the algorithm used for the construction of a polygonization for a given set of points is described. The strategies used in order to minimize the polygonization area are also introduced. In section 3, the proposed algorithms to search for approximate solutions for MAP problem are introduced. The experimental design and results are described in section 4. Finally, some conclusions are exposed in section 5.

2 Constructing Simple Polygonizations

In this section, we describe the algorithm used for the construction of a polygonization for a given set of points. The strategies used during the construction in order to minimize the polygonization area are also introduced.
The algorithm for the construction of a polygonization is based in the heuristic *SteadyGrowth*, proposed by Auer and Held for the generation of random polygons [2]. Given a set S of points in the plane, in the initial phase the algorithm selects three points p_i, p_j, $p_k \in S$, such that no other point of S lies within the triangle formed by p_i, p_j, p_k. After that, in each phase, the algorithm builds a polygon P_k, adding one feasible point $p \in S$. A point p is *feasible* if no remaining points of S - $\{p\}$ lie in the interior of the convex hull CH($P_k \cup \{p\}$) and there is at least one edge of P_k completely visible from p. In order to add the feasible point to the solution, an edge (p_j, p_k) from the set of visible edges must be selected and replaced with the edges (p_j, p) and (p, p_k). This algorithm is illustrated below.

Algorithm 1 ***BuildPolygonization***
$P_k \leftarrow$ *BuildInitialTriangle*(S)
S $\leftarrow$ S - {p / p $\in P_k$}
while (S $\neq \emptyset$) **do**
  FP $\leftarrow$ *FeasiblePoints*(S)
  p $\leftarrow$ *SelectFeasiblePoint*(FP)
  VE $\leftarrow$ *VisibleEdges*(p, P_k)
  (p_j, p_k) $\leftarrow$ *SelectEdge*(VE)
  AddPointSolution(p, (p_j,p_k), P_k)
  S $\leftarrow$ S - {p}
end while

The main components of the algorithm *BuildPolygonization* are described:

- *BuildInitialTriangle*(S): returns a triangle formed by p_i,p_j,$p_k \in S$.
- *FeasiblePoints*(S): returns the set of feasible points $p \in S$.
- *SelectFeasiblePoint*(FP): returns a feasible point $p \in S$.
- *VisibleEdges*(p,P_k): returns the set of edges of P_k completely visible from p.
- *SelectEdge*(VE): returns an edge (p_j,p_k)$\in VE$.

For the construction of the solutions of MAP problem we use the algorithm *BuildPolygonization*. In order to minimize the polygonization area, we apply different criteria for: the selection of the points in the initial triangle, the selection of the next feasible point to be added and the selection of the edge to be replaced.
The initial triangle points are selected according to either of the following criteria:

- *RandomTriangle*(RT): the three points are randomly selected.

– *RandomGreedyTriangle*(RGT): the first point is randomly selected and the other two are the nearest ones to the first point selected.

The feasible point is selected according to either of the following criteria:

– *RandomPoint*(RP): of the set of feasible points, one is randomly selected.
– *GreedyPoint*(GP): of the set of feasible points, the nearest one to the current polygon P_k is selected.

The edge is selected according to the next criteria:

– *GreedyEdge*(GE): of the set of visible edges, we select the edge together with the feasible point already chosen adding the minimum area.

An additional criterion involving the selection of a point and edge is proposed:

– *GreedyArea*(GAr): of the set of feasible points, the point together with the visible edge adding the minimum area are selected.

The combination of the different criteria presented above results in six different strategies:

– A_1: for the construction of the initial triangle RGT criterion is applied. GP is used for the selection of a point and GE for the selection of an edge.
– A_2: for the construction of the initial triangle RGT is applied. RP is used for the selection of a point and GE for the selection of an edge.
– A_3: for the construction of the initial triangle RGT is used. Then, the GAr criterion is applied.
– A_4: for the construction of the initial triangle RT is applied. GP is used for the selection of a point and GE for the selection of an edge.
– A_5: for the construction of the initial triangle the RT is applied. RP is used for the selection of a point and GE for the selection of an edge.
– A_6: for the construction of the initial triangle the RT is used. Then, GAr criterion is applied.

3 Searching Approximate Solutions for MAP Problem

In this section, the proposed algorithms to search for approximate solutions for MAP problem are introduced.

3.1 A Greedy Algorithm for MAP Problem (Greedy-MAP)

A greedy algorithm is an algorithm that follows the heuristic of making the locally optimal choice at each stage expecting to find the global optimum. We have implemented a greedy algorithm Greedy-MAP for the construction of polygonizations. We use *BuildPolygonization* algorithm according to the following greedy strategy: for the construction of the initial triangle, of all the feasible triangles formed by points in S, the triangle with minimum area is selected. In the next steps, of the set of feasible points, the point together with the visible edge adding the minimum area are selected.

3.2 Random Search (RS-MAP)

We propose a simple algorithm to search for approximate solutions for MAP problem. The algorithm RS-MAP is a random search over the space of feasible solutions. For every instance of each collection a certain number of solutions are constructed using *BuildPolygonization* algorithm with the six different strategies explained in section 2. In each case, the solution that minimizes the polygonization area is selected.

3.3 Ant Colony Optimization (ACO-MAP)

As exposed before, the MAP problem is NP-hard. Metaheuristics techniques are especially well suited for this kind of problems when approximate high quality solutions need to be obtained. A metaheuristic is a general algorithmic framework which can be adapted to different optimization problems with minor adjustments [3] [10]. *Ant Colony Optimization* (ACO) is a metaheuristic inspired by the behavior of real ants colonies. The ACO metaheuristic involves a family of algorithms based on a colony of artificial ants, that is simple computational agents that work cooperatively finding good solutions to difficult optimization problems [6]. ACO algorithms are essentially construction algorithms: in each iteration, every ant constructs a solution for the problem traveling on a construction graph. Every edge in the graph represents possible steps for the ant. An edge has associated two kinds of information to guide the ant movement: *Heuristic information* and *artificial pheromone trail information*. Heuristic information, η_{pq}, measures the heuristic preference of moving from node p to node q. This information is not modified by the ants during the algorithm. Artificial pheromone trail information, τ_{pq}, measures the "learned desirability" of the movement from node p to node q. This information is modified during the algorithm depending on the solutions found by the ants. Next, a general ACO algorithm is illustrated.

Algorithm 2 ***GeneralACO***
Initialize
for c=1 **until** C **do do**
  for k=1 **until** K **do do**
    BuildSolutionk
    EvaluateSolution
  end for
  SaveBestSolutionSoFar
  UpdateTrails
end for
ReturnBestSolution

The main components of the algorithm *GeneralACO* are described:

– *Initialize*: this process initializes the parameters of the algorithm. The initial trail of pheromone associated to each edge, τ_0 . It is an small positive

value, the same for all edges. The weights defining the proportion in which the heuristic information will be affected, β. The pheromone trails in the probabilistic transition rule, α. The quantity of ants of the colony K and the maximum number of cycles C .

– *BuildSolutionk*: this process begins with a partial solution. In each step of the construction, the solution is extended adding a feasible solution component selected from the current solution neighbors. The selection of a feasible neighbor is done in a probabilistic way, according to the ACO variant used. In this work, the selection rule for the solutions construction is based on the following probabilistic model:

$$P_{ij}(k) = \begin{cases} \frac{\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}}{\sum_{h \in F(p_i)} \tau_{ih}^{\alpha}.\eta_{ih}^{\beta}}, \text{j} \in \text{F}(\text{p}_i); \\ 0, \qquad \text{otherwise.} \end{cases} \tag{1}$$

- $\text{F}(\text{p}_i)$ is the set of feasible points for point p_i.
- τ_{ij} is the pheromone value associated to edge (p_i, p_j).
- $\eta_{ij} = 1/d_{ij}$ is the heuristic value associated to edge (p_i, p_j). d_{ij} is the euclidean distance between p_i and p_j.
- α y β are positives parameters. They define the relative importance of the pheromone with respect to the heuristic information.

– *EvaluateSolution*: evaluates and saves the best solution found by the ant k in the current cycle.

– *SaveBestSolutionSoFar*: saves the best solution found for all cycles so far.

– *UpdateTrails*: modifies the pheromone level in the promising paths. First, all the pheromone values are decreased by means of the process of evaporation. Then, the pheromone level is increased when good solutions appear. The following equation is used:

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

- $\rho \in (0, 1]$ is the factor of persistence of the trail.
- $\Delta\tau_{ij} = \sum_{k=1}^{NroAnts} \Delta^k\tau_{ij}$ is the accumulation trail, proportional to the quality of the solutions.
- $\Delta^k\tau_{ij} = \begin{cases} 1/L_k, \text{ if ant } k \text{ used edge } p_i, p_j; \\ 0, \quad \text{otherwise} \end{cases}$
- L_k is the objective value of the solution k .

Pheromone evaporation avoids a fast convergence of the algorithm. This way of forgetting allows the exploration of new areas of the search space. The update of the pheromone trail is done according to either of the following criteria: elitist and not elitist. In the elitist case, 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.
We propose ACO-MAP, an ACO algorithm to find approximate solutions for MAP problem. In the proposed ACO algorithm *BuildSolutionk* process in Algorithm 2 utilizes *BuildPolygonization* algorithm explained in the previous section. *BuildSolutionk* builds a solution according to the following criteria: the points of initial triangle and the next feasible point to be added are randomly selected. The edge to be replaced is chosen according to equation 1.

4 Experimental Design and Results

When evaluating approximate algorithms, it is usually necessary to consider a large amount of data to evaluate them. It is required to test their performance for a large number of practical and relevant data. For our experimental study, we need to consider large planar sets of points, in general position and randomly generated. As far as the authors are concerned, there do not exist collections of instances for benchmarking purposes for MAP problem. According to that, we have generated an input data set that includes six collections of general position set of points with 40, 80, 100, 120, 160 and 200 points respectively. Every collection contains 10 instances which results in a total of 60 instances of the problem. Each instance is called LPn_i where n denotes the size of the i-instance, with i between 1 and 10. The points of each instance are randomly generated and uniformly distributed. For each point (x,y), the coordinates x and y are in the interval [0, 9999]. In this work, we consider that a set of points in the plane is to be in general position only if no three of them lie on the same straight line. Therefore, all the instances were pre-processed for assuring non collinear points. The initial experimental phase is presented next. The experimental results obtained from the proposed algorithms Greedy-MAP, RS-MAP and ACO-MAP are showed. We compare their performance over four instances of the LP40, LP80 and LP100 collections.
First, we analyze the performance of the RS-MAP algorithm. For every instance of each collection 1000 solutions were constructed using different random seeds. We use *BuildPolygonization* algorithm with the six strategies presented in section 2. Table 1 shows the best areas obtained from each strategy. As we can observe, A_3 obtains the better solutions for all the instances of collections LP80 and LP100. With the exception of instance LP100_3. For the instances of collection LP40 the better areas are obtained from different strategies.
Next, we analyze the performance of the ACO-MAP algorithm. The ACO algorithm proposed are represented by an Ant System (AS), a particular instance of the class of ACO algorithms [5]. We used the following parameter values: α=1; β=1 and 5; ρ=0.10, 0.25, and 0.50; $elitism$=1 and 0, where 1 means that the trail is updated in an elitist way; in the other case, the updating is done in a not elitist way. The number of cycles, C is 1000; the number of ants, K is 50. For each parameter setting, 30 runs using different random seeds were performed. For each instance, the experiments were done with the twelve parameter setting, according to combinations of parameters presented. We obtain average, median, best,

Table 1. RS-MAP: best results for instances of collections LP40, LP80 and LP100

Instance	A_1	A_2	A_3	A_4	A_5	A_6
LP40_1	16913767	**14247604**	14970807	17033176	16142866	15809710
LP40_2	21718233	17397129	19308453	17638831	17833915	**16682123**
LP40_3	12724842	**12397361**	15467802	14463436	15590652	15786309
LP40_4	15732311	14107508	14637508	17588575	**13855953**	13858624
LP80_1	15334777	13861116	**12656857**	15397954	14046638	13681373
LP80_2	18525955	18303625	**15400620**	19120798	18140867	16007601
LP80_3	13765710	14063122	**12149490**	11796302	13146207	12873346
LP80_4	15827350	15496821	**14129195**	14348388	15841027	14608346
LP100_1	18839948	16843730	**14913968**	15795024	17983478	17140705
LP100_2	16977174	15003050	**14524855**	16201088	15467078	14763797
LP100_3	17660485	15242272	14841126	16442687	**14288550**	15238502
LP100_4	16188099	15313679	**14228908**	16902207	15472613	14565310

and standard deviation values, considering the objective function (area). Each parameter setting is denoted by (β-ρ-*elit*). For all cases α=1 so it is not shown. Table 2 shows the best results for instances of collection LP40, LP80 and LP100.

Table 2. ACO-MAP: best results for instances of collections LP40, LP80 and LP100

Instance	*Par.Setting*	*BestArea*	*Average*	*StdDev*
LP40_1	5-0.50-0	13407676	15772361	924860
LP40_2	5-0.50-0	16878114	18652494	813821
LP40_3	1-0.25-0	14191890	18111581	1047062
LP40_4	5-0.50-0	13782235	15312044	489332
LP80_1	5-0.50-1	15644003	17278058	549557
LP80_2	1-0.50-1	18267612	22007953	1009033
LP80_3	5-0.50-1	15457009	17508622	650806
LP80_4	1-0.50-0	17353796	20734511	1010593
LP100_1	5-0.25-1	20645166	23834001	783239
LP100_2	5-0.50-1	18694220	20810168	676606
LP100_3	5-0.25-1	19418940	20709942	597942
LP100_4	5-0.50-0	18014838	20092936	615742

For collection LP40 the best parameter settings are the same for LP40_1, LP40_2 and LP40_4 instances. The best parameter setting for LP40_3 are not equal to the other sets. The best results were obtained giving more relevance to the heuristic information and updating the trails in a not elitist way. Most of the smaller areas were obtained using configurations with β=5, ρ=0.50 and *elit*=0.

For collection LP80 the best parameter settings are the same for LP80_1 and LP80_3 instances. The LP80_2 and LP80_4 instances share the same parameter setting with each other except for the elitism parameter. The best results were obtained updating the trails in an elitist way. The smaller areas were obtained using configurations with β between 1 and 5, ρ=0.50 and $elit$=1.
For collection LP100 the best parameter settings are the same for LP100_1 and LP100_3 instances. Between LP100_2 and LP100_4 instances the best parameter settings are different only for the elitism parameter. The best results were obtained updating the trails with an elitist way and giving more relevance to the heuristic information. The smaller areas were obtained using configurations with β=5, ρ between 0.25 and 0.50 and $elit$=1.
To the best knowledge of the authors, there do not exist benchmarks publicly available for MAP problem that allow us to compare our results. Table 3 shows the best areas obtained from the proposed algorithms. The results obtained from Greedy-MAP algorithm are presented in the first column of the table. The second and third column shows the best areas obtained from RS-MAP and ACO-MAP algorithms. As we can observe in table 3, for collection LP40 in only two instances the smaller areas were obtained from ACO-MAP. For all instances of collections LP80 and LP100 the smaller areas were obtained from RS-MAP. Although, the results obtained from RS-MAP seem to be the most promising, it is required to contrast them with more experimental results obtained from other techniques. These results obtained from RS-MAP could be used as a benchmark. Metaheuristic techniques have proved to be very well behaved in solving NP-hard problems. So, a further study about the behavior of the ACO algorithm is necessary regarding the performance observed for the larger instances. This initial experimental study provides preliminary results that will guide future experimentation.

Table 3. Comparison of best results obtained from Greedy-MAP, RS-MAP and ACO-MAP.

Instance	***Greedy-MAP***	***RS-MAP***	***ACO-MAP***
LP40_1	24153933	14247604	**13407676**
LP40_2	24016120	**16682123**	16878114
LP40_3	21197963	**12397361**	14191890
LP40_4	18749234	13855953	**13782235**
LP80_1	18465063	**12656857**	15644003
LP80_2	24920432	**15400620**	18267612
LP80_3	13932867	**12149490**	15457009
LP80_4	16112604	**14129195**	17353796
LP100_1	23183435	**14913968**	20645166
LP100_2	18163151	**14524855**	18694220
LP100_3	16049978	**14841126**	19418940
LP100_4	17156672	**14228908**	18014838

5 Conclusion

In this work, we proposed algorithms to obtain approximate solutions for Minimum Area Polygonization problem for planar sets of points. We have generated collections of instances for the experimental evaluation. This is another contribution of this work because of there are not available collections of instances for benchmarking purposes for MAP problem. We performed an initial experimental study and we obtained preliminary results that will guide future experimentation. This experimental stage gave us information for future research concerning to improve the performance of the ACO algorithm. Actually, we are working in the experimental design with the whole collection of instances generated. In addition, we aim to carry out an statistical analysis, in order to determine the best approach for the proposed problem. As a future work, we intend to adapt and implement other metaheuristics to solve the proposed problem.

References

1. Abellanas M., *Conectando puntos: poligonizaciones y otros problemas relacionados.* Gaceta de la Real Sociedad Matemtica Espaola 11(3), 543558, (2008).
2. Auer T. and Held M., *Heuristics for the generation of random polygons.* Proc. 8th Canadian Conference Computational Geometry, 38-44, (1996).
3. Bäck T., Fogel D. and Michalewicz Z., *Handbook of Evolutionary Computation.* IOP Publishing Ltd and Oxford University Press, (1997).
4. Deneen L. and Shute G., *Polygonizations of point sets in the plane.* Discrete and Computational Geometry 3, 7787, (1988).
5. Dorigo M., Maniezzo V. and Colorni A. *The Ant System: An autocatalytic optimizing process.* Technical report 91-016 revised, Dipartimento di Elettronica, Politecnico di Milano, Milan, (1991).
6. Dorigo M. and Stützle T. *Ant Colony Optimization.* Massachusetts Institute of Technology, (2004).
7. Demaine,E. *Simple Polygonizations*, http://erikdemaine.org/polygonization/
8. Fekete S. *On simple polygonizations with optimal area.* Discrete and Computational Geometry 23, 73-110, (2000).
9. Garey M. and Johnson D. *Computers and Intractability: a guide of theory of NP-completeness.* Freeman, (1979).
10. Michalewicz Z. and Fogel D. *How to Solve It: Modern Heuristics.* 2nd Edition, Springer, (2004).