The Visual Computer

Multicriteria-optimized triangulations

I. Kolingerová[1],
A. Ferko[2]

[1] Department of Computer Science and Engineering, University of West Bohemia, Univerzitní 22, Box 314, 306 14 Plzeň, Czech Republic
E-mail: kolinger@kiv.zcu.cz
http://iason.zcu.cz/~kolinger
[2] Department of Computer Graphics and Image Processing, Comenius University, Mlynska dolina, Bratislava, Slovak Republic
E-mail: ferko@fmph.uniba.sk
http://www.uniba.sk/~kpgso

Published online: 2 August 2001

Triangulation of a given set of points in a plane is one of the most commonly solved problems in computer graphics and computational geometry. Because they are useful in many applications, triangulations must provide well-shaped triangles. Many criteria have been developed to provide such meshes, namely weight and angular criteria. Each criterion has its pros and cons, some of them are difficult to compute, and sometimes even the polynomial algorithm is not known. By any of the existing deterministic methods, it is not possible to compute a triangulation which satisfies more than one criterion or which contains parts developed according to several criteria. We explain how such a mixture can be generated using genetic optimization.

Key words: Computer graphics – Computational geometry – Minimum weight triangulation – Delaunay triangulation – Genetic optimization

Correpondence to: I. Kolingerová

1 Introduction

A triangulation $T(S)$ of a set of points in the Euclidean plane is a set of edges E such that no two edges in E intersect in a point not in S and the edges in E divide the convex hull of S into triangles.
Triangulations are necessary in many areas, e.g., computer graphics, computer vision, robotics, CAGD, etc. Therefore, significant research has been devoted to this problem and many triangulation methods developed. To work well in applications, triangle meshes are usually assumed to be locally or globally optimal under some criterion.

(Globally) optimal triangulation

A triangulation $OT(S)$ is called (globally) optimal with respect to the given criterion provided that $OT(S)$ optimizes the given criterion among all possible triangulations of S. With a few exceptions, the construction of $OT(S)$ in polynomial time is not known. In order to find a global optimum, an exponential number of triangulations would need to be checked; thus, it is necessary either to cope with local extremes or to use methods leading to a suboptimum.

Locally optimal triangulation [45]

A triangulation $LOT(S)$ is called locally optimal with respect to a given criterion provided that every convex quadrilateral (consisting of two triangles sharing an interior edge) is optimal with respect to the given criterion. Local optimality is illustrated in Fig. 1. Let the local optimality criterion be defined as the shortest possible edge length. Then in Fig. 1a, the quadrilateral is locally optimal, while in Fig. 1b it is not. Locally optimal triangulations are usually constructed by the local edge-flip procedure suggested in [33] – the diagonal of a non-optimal quadrilateral is flipped, i.e., replaced by the other diagonal. It has been proved in [33] that the local edge-flip procedure needs a finite number of steps to converge to a local optimum, as there is a finite number of triangulations on the given set of points. This maximal number of flips equals the mutual distance of two triangulations $O(N^2)$, where N is the total number of points in S. Expected behaviour is, however, acceptable – about $O(N)$ steps. The local edge-flip algorithm does not guarantee convergance to a global extreme; moreover, it is not possible to detect whether the triangulation is (globally) optimal or not. (However, even a local extreme may work well for some types of criteria and some input data).

Digital Object Identifier (DOI) 10.1007/s003710100125

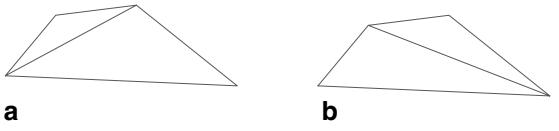


Fig. 1a,b. Local optimality; the quadrilateral in **a** is locally optimal with respect to the criterion of shorter edge, while in **b** it is not

Let us now return to global criteria and $OT(S)$. There are two main classes of criteria optimized in triangulations. The first one is connected with the weight (defined as a sum of lengths) of edges, the second one with the size of angles. Both classes can be represented by several triangulators, each having their pros and cons.
The most important criteria of the first class include the following:

- Minimization of the triangulation weight, optimized in the so-called minimum weight triangulation, *MWT* [1, 2, 5, 13, 14, 25, 27, 31, 32, 35, 37, 42, 47]
- Maximization of the minimal triangle height (minimum Euclidean distance from a vertex to the opposite edge), optimized in the so-called maximum height triangulation [6, 40]
- Minimization of the maximum edge length, optimized in the minmax length triangulation [17, 19]
- Minimization of the maximal aspect ratio (the ratio of the length of the longest side of the triangle to the height of the triangle, where the height of a triangle is the Euclidean distance of the longest edge to its opposite vertex) [6]

The second class includes the following criteria:

- Maximization of the minimum angle, optimized in the so-called Delaunay triangulation, *DT* [3, 8, 40, 41, 43]
- Minimization of the maximum angle, optimized in the so-called minmax angle triangulation [16, 18]
- Minimization of the minimal angle, optimized in the Delaunay triangulation of the farthest point [20]
- Minimization of the maximum eccentricity (the infimum over all distances between c_1 and the vertices of the triangle, where c_1 is the centre of the triangle's circumcircle), optimized in the minmax eccentricity triangulation [6, 7]
- Maximization of the sum of the angles [39]

With general data sets, the triangulations satisfying these criteria can provide different results (although there are data sets where some of them coincide). By geometrically based, deterministic methods, it is not possible to compute a triangle mesh which is a compromise between extremes, providing small weight as well as good angles. In addition, it is not possible to generate a mesh angularly optimized in some of points and weight optimized in the rest. Such triangulations can be generated only with the use of nondeterministic optimization methods, e.g., simulated annealing or genetic programming, as these techniques allow the weighted sum or multiplication of several criteria to be optimized.
In this paper it will be shown how genetic optimization (GO) can be used to construct a triangulation which fulfils more than one criterion. Moreover, we introduce a method which balances the influences of the criteria with the help of weights, according to the user's needs, and which combines different criteria in different parts of the plane.
The rest of the paper is structured as follows: Sect. 2 briefly explains GO. Section 3 shows how GO can be used in the space of triangulations. Section 4 briefly recapitulates *MWT* and *DT* as the main representatives of the two classes of triangulation criteria and presents formulations of criteria suitable for the genetic approach. Section 5 contains examples and discussion of results. Section 6 concludes the paper.

2 Principles of GO

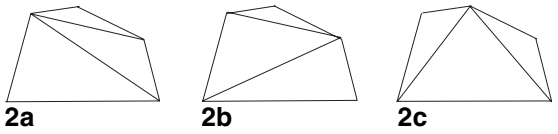
A detailed explanation of this topic has already been given, for example, in [24, 26, 38]. GO is a combination of direct and heuristic searches, usually in discrete space. A set of potential solutions is prepared. Members of this set are evaluated according to their value in an extremized function, and the better of them have a higher probability to be chosen for modification by a binary operation. Also a unary operation is used to modify the set members. From such a modified and enlarged set of potential solutions, those with higher evaluation have higher probability to be chosen for the next iteration.
The process cycles for many iterations, and the set of potential solutions converges to an extreme. Convergence is not guaranteed, as the method is probabilis-

Genetic optimization

Input: the given problem
Output: a suboptimal solution
Symbols: t – time
$P(t)$ – a population in time t (tth generation)

```
1. begin
2.   Initialize a population P(0);
3.   Evaluate P(0); t := 0;
5.   while not termination_condition do
6.     begin
7.       t := t + 1;
8.       Select a new population P(t) from P(t − 1);
9.       Alter P(t);
10.      Evaluate P(t)
11.    end;
12.  Return the best fitting individual as a solution
13. end
```

Algorithm 1



Algorithm 1. Genetic optimization
Fig. 2a–c. Dataset from [40]: {(0.0, 0.0), (1.4, 4.5), (3.8, 4.9), (7.15, 3.1), (8.0, 0.0)}. Sequence of local flips does not guarantee convergence to global extreme: **a** An initial triangulation. **b** Local extreme achieved after one flip; no more local flips are possible. **c** Global extreme (not achieved by previous flip sequence). Any initial triangulation of this data set which contains the shortest inner edge is incapable of converging to a global optimum

tic. The lower bound for the quality of the extreme solution can be ensured if some sub-optimal potential solution is included into the initial set and if the best solution found during all the iterations is kept as a candidate for the 'final answer'.
The method uses biological terminology: the set is called 'a population'; one potential solution is 'an individual'; the binary operation is 'a crossover'; and unary is 'a mutation'.
The general scheme of genetic optimization is outlined in Algorithm 1.
GO provides an efficient tool for finding the global extreme in a huge state space where an exhaustive search would not be possible. Like simulated annealing, it can, however, converge to a local extreme. In comparison with simulated annealing its greatest disadvantage is that many potential solutions are computed in each iteration – it substantially increases time and memory requirements. As already mentioned, GO is mostly used for discrete optimization. Typical problems are the knapsack problem or the travelling salesman problem.

3 GO of triangulations

The probabilistic approach has been successfully tried in [45], where simulated annealing was used to optimize lexicographical versions of various criteria. (The lexicographical version of some criterion is a lexicographically sorted vector of values of the criterion for all edges, vertices or triangles.) The *MWT* approximation has been solved by simulated annealing in [4] and by genetic programming in [11, 29, 30, 44, 46].
The reasons for trying GO on globally optimal triangulations are obvious:

- Globally optimal triangulations, such as minimum weight triangulation, are suspected of being NP-complete (but it has not been proved yet).
- The state space of triangulations is large, and direct search is infeasible.
- The quick and simple edge-flip algorithm generally does not lead to a global optimum. See the example given in Fig. 2 [40]. The choice of a more complex flip operator does not change this property.

Key points leading to the success or failure of the genetic approach are proper design and parameters of GO, such as the representation of individuals, simple and efficient operators, an adequate number and size of generations and an efficient fitness function. In the following text, we will present details of a newly proposed genetic solution and a comparison with existing ones [11, 44, 46].

Data representation

One individual is one triangulation represented as a list of edges and a list of triangles for the given data set.

An initial population

Generally, any triangulation can serve as an initial triangulation. However, the set of initial triangulations should have some diversity in order to properly represent the space searched. For this purpose, the algorithm of incremental insertion seems a good choice, because with different random orders of inserted points, different triangulations may be generated. This algorithm works as follows (see Fig. 3). First, a triangulation of the convex hull of points is constructed (Fig. 3a,b). Then inner points are inserted one at a time, and the triangulation is updated with each insertion (Fig. 3c,d). The update is usually combined with a local edge flip to avoid long, thin triangles. For the sake of GO, edge flips are not absolutely necessary, but are recommended for three reasons:

- When an inner point is inserted, the triangle in which the point lies has to be located. Long, thin triangles increase the probability of a wrong answer due to floating-point inaccuracy.
- Although the GO approach is very general, it is highly improbable that its practical application would require a criterion leading to long, thin triangles. Why not start with triangulations which are closer to the probable result, then? There is a danger of premature convergence to local optimum; however, according to our experience, better results are obtained with a better-fitting initial population.
- In genetic theory, the diversity of the initial generation is pointed out. However, according to our experiments, it pays to limit the space searched if possible, e.g., to incorporate some necessary condition for edges or triangles of the searched $OT(S)$, if such a condition is available; this decreases the number of possible triangulations and so decreases the size of the space searched, which speeds up the convergence of the GO. One example can be found in [30], where better results in the approximation of *MWT* were achieved when the search space was limited to locally minimal edges and triangulations.

Size of a generation

There is a trade-off between the size of a generation, *pop_size*, and the computation time. The number of generations times *pop_size* gives the total number of triangulations that has to be computed; the higher this number, the longer the computation time to be expected. As shown later, size 100 looks to be a compromise between the two factors.

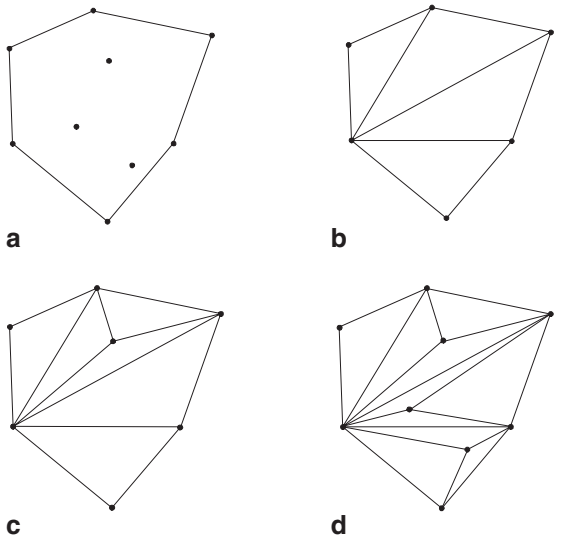


Fig. 3a–d. An example of an initial triangulation constructed incrementally (no edge flips considered): **a** The given set of points. **b** An initial triangulation of the convex hull of S. **c** After the first inner point insertion. **d** All inner points inserted

Termination condition

The best termination condition would be to stop when the required global optimum is achieved. As this value (or, at least, its good estimate) is only rarely available, such a termination condition is useless. The simplest termination condition is to stop after some given number of generations. As no general rule for the correct number of generations exists, usually some initial experiments are necessary to choose the value. We mostly used 100 or 200 generations in our implementation.

As well as this simple termination condition, we used the condition based on the rate of improvement of the best solution found during all the iterations. If the best solution has not been improved substantially during some given number of generations, the iteration is stopped (e.g., if within $0.75 \times$ total_number_of_generations there is a less than 1% improvement in evaluation of the best individual).

Fitness function

Each triangulation has to be evaluated to judge its fitness according to the given criterion. A better-fitting triangulation has a higher probability to be selected for the next generation. Therefore, a well-designed fitness function is a very important factor in GO convergence.

In the triangulation problem, the design of the fitness function is not so demanding, because triangulation criteria are mathematically clearly defined. Two versions of the fitness function can be used:

$$eval_1(T(S)) = \sum_{i=1}^{m} w_i f_i,$$

$$eval_2(T(S)) = \prod_{i=1}^{m} f_i,$$

where $T(S)$ is a triangulation; m is the total number of triangulation criteria; w_i, $i = 1, 2 \ldots, m$, are weight coefficients, $w_i \in \langle 0; 1\rangle$, $\sum w_i = 1$; and f_i, $i = 1, 2 \ldots, m$ are functions for particular criteria.

$eval_1$ is in fact a linear interpolation in the convex polyhedron, where each vertex represents some criterion. $eval_2$ does not enable particular criteria to be weight influenced; however, it can be very useful if the function values of f_i are mutually very different (several orders of magnitude). For example, let f_1 have angular values $f_1 \in \langle 0; 2\pi\rangle$, while f_2 has length values $f_2 \in \langle 0; \infty)$. Thus f_1 is upper bounded, while f_2 not. This may bring problems: If such a pair is combined in $eval_1$, it may be insensitive to f_1, because its values are bounded and probably much lower than the f_2 values. Proper mathematical formulations of geometric properties for f_i are presented in Sect. 4.

Selection of a new generation

Let one generation have *pop_size* members (triangulations). The interval $\langle 0; 1\rangle$ of probabilities for triangulations to be chosen then corresponds to *pop_size* members. Each triangulation T_i, $i = 1, 2, \ldots,$ *pop_size* can be evaluated within the interval

$$\frac{\sum_{j=1}^{i-1} eval_j}{\sum_{j=1}^{pop\_size} eval_j} \quad \text{to} \quad \frac{\sum_{j=1}^{i} eval_j}{\sum_{j=1}^{pop\_size} eval_j}$$

Mutation procedure

Input: $P(t)$ – the population of triangulations,
p_m – probability of mutation
Output: $P(t)$ – the mutated population
Symbols: nE – number of inner edges in one triangulation
pop_size – number of triangulations in one generation

```
1. begin
2.   for e := 1 to nE*pop_size do {for each edge e in P(t)}
3.     begin
4.       Generate a random value r ∈ ⟨0;1⟩;
5.       if r < p_m then Flip the edge e if possible
7.     end
8.   Return P(t)
9. end
```

Algorithm 2. Mutation

If a random number $r \in \langle 0; 1\rangle$ is generated, it lies in one of these intervals. The corresponding triangulation is then selected into a new generation. As we keep the size of the generations constant, in each iteration *pop_size* random numbers are generated. This technique picks *pop_size* triangulations; some of them are selected more than once, others not at all; and the higher the evaluation, the higher the probability to be selected.

Mutation operator

Mutation was traditionally understood as a small, atomic change performed by bit inversion if individuals were represented by binary numbers. With regard to our representation by lists of edges and triangles, mutation is done by an edge flip. The flip is also a kind of atomic operation in triangulation. The probability of mutation is given by a fixed value, p_m. (In genetic literature, $p_m = 0.01$ is most often; however, we have better experience with a lower value, such as 0.001. More details are given in Sect. 5.) See Algorithm 2.

Crossover operator

Crossover combines two individuals into two new ones. The main idea is to enable exchange of genetic information. The natural analogy for triangulation is to combine two triangle meshes into two new ones. To develop an efficient operator for this is difficult because the result might not be a valid triangulation. The invalid triangulation is, e.g., a nonplanar graph, and therefore corrections are necessary.

Parents are selected on a probabilistic basis as follows: for a particular triangulation, a random number is generated. If the number is lower than a fixed value, p_c (the probability of crossover), the triangulation is selected for crossover with another individual. According to the genetic publications and our experience, $p_c = 0.25$ is proper; more details are given in Sect. 5.

In classical binary representation, crossover is performed by a mutual exchange of parts of two binary numbers. Several alternatives to triangulation crossover have been tried in our implementation. The best one seems to be the so-called DeWall operator (named after the DeWall algorithm for Delaunay triangulation by the divide and conquer approach in [12] which inspired the operator) as it preserves all of the already formed groups of geometrically proximate edges and thus improves convergence.

Let us describe the workings of this operation in a more simplified form – as if the triangulation were created only by a set of edges. Let there be two planar triangulations, T_F (the 'father') and T_M (the 'mother'), and a randomly generated line L intersecting the convex hull of the triangulation (see Fig. 4a,b). The line divides the edges in a triangulation into three groups: those intersected by the line (T^0), those on the left side of the line (T^-) and those on the right side of the line (T^+). Shuffling of genetic information is ensured by combining information from the parents – their edges. The probabilistic nature of the operator is created by the random generation of the line. There are other possible ways to generate such a line, e.g., two points inside the minmax box of the given point set can be generated.

As the line L has the same position in both triangulations, the sets T_F^- of T_F and T_M^+ of T_M are separated by the line and cannot intersect. (The same is true for T_F^+ and T_M^-, but we will describe only one of the two possible combinations.) Therefore, T_F^- and T_M^+ can be combined into one set of edges without any intersection tests. Together, they are not a complete triangulation and have to be combined with edges from T_F^0, T_M^0. The T_F^0 edges have to be tested for intersection with the T_M^+ edges already present in a 'child' triangulation. The same occurs for the T_M^0 and T_F^- edges.

After insertion of several T_M^0 and T_F^0 edges, the resulting subgraphs still may not form a complete triangulation. This problem can be solved in two ways: (a) other possible edges are generated ("full crossover") and (b) the operation is halted ("reduced crossover"). We tried both approaches; therefore, we can compare their pros and cons.

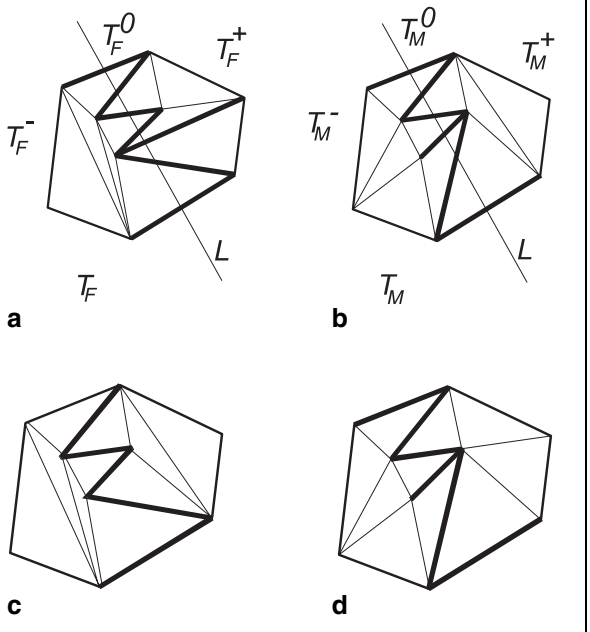


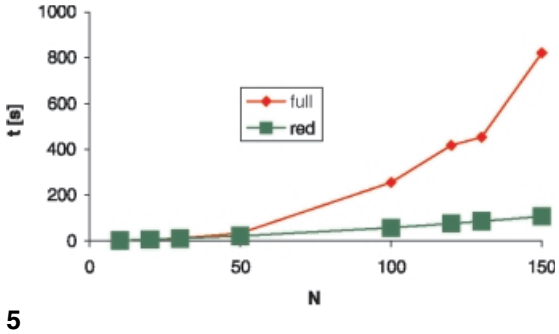
Fig. 4a–d. a,b "Father" and "mother" triangulations intersected by the random line L; **c,d** "children" of this crossover

For full crossover it is necessary either to use some sophisticated algorithm to "sew up" the gaps or to have a set of candidate edges where proper missing edges are searched. We used the latter approach: a set of all possible edges was generated in pre-processing and was used when necessary.

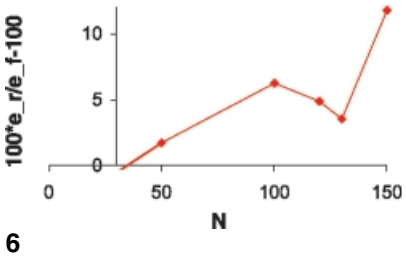
Reduced crossover has an inherent inefficiency, as part of the crossover is never finished; on the other hand, expensive intersection tests of candidate edges against edges already accepted in triangulation are avoided.

Results were slightly better for full crossover than for reduced crossover for the same number of iterations and the same population size. However, computation time for full crossover was much higher (see Fig. 5). When the time saved by reduced crossover was used for a higher number of generations, the results spoke for reduced crossover (see Fig. 6). It can be seen that reduced crossover utilizes its time better.

In our experiments, reduced crossover failed to finish in at most 10%–15% of the total number of crossovers. This wasted time is more than returned



5



6

Fig. 5. Computation time for reduced and full crossover (100 generations per 50 triangulations, $p_c = 0.25$, $p_m = 0.001$)

Fig. 6. Rate of the best evaluation obtained by reduced and full crossover within the same computational time (the time which full crossover needed to compute 100 generations per 50 triangulations): $f = 100 \times e_r/e_f - 100$ (%), where e_r is the evaluation obtained by reduced crossover and e_f the evaluation obtained by full crossover ($p_c = 0.25$, $p_m = 0.001$)

Crossover procedure

Input: two triangulations, T_F and T_M (parents), the set of all possible edges cE

Output: two triangulations, T_{C1} and T_{C2} (children)

1. **begin**
2. Generate a random line L;
3. Divide edges of T_F into T_F^-, T_F^+, T_F^0; (see Fig. 4)
4. Divide edges of T_M into T_M^-, T_M^+, T_M^0;
5. Accept T_F^- and T_M^+ into the child T_{C1};
6. **if** T_{C1} is not complete **then** Insert such edges from T_F^0 which do not intersect T_M^+ edges;
7. **if** T_{C1} is not complete **then** Insert the edges from T_M^0 which do not intersect T_F^- edges;
8. **if** T_{C1} is not complete **then** Insert the edges from cE which do not intersect edges in T_{C1};
9. Swap T_F and T_M and repeat steps $5-8$ once more for T_{C2};
10. Return T_{C1}, T_{C2}
11. **end**

Algorithm 3

Algorithm 3. Crossover

by the substantial speed-up (recall Fig. 5). Therefore, we decided to choose reduced crossover.

The full version of the crossover operator is given in more detail in Algorithm 3. For reduced crossover, the input parameter cE and step 8 of the algorithm are left out. If the triangulation is also represented by a list of triangles, which is usually the case, the triangles have to be operated together with edges to keep data structures consistent. This is not shown in Algorithm 3 for better readability.

This concludes the explanation of the GO for triangulation purposes. In regard to the complexity of the proposed solution, we see that the initial triangulations can be computed in an expected time of $O(N \log N)$ and a worst time of $O(N^2)$ (this is implied by the incremental construction). Mutation needs a time equal to $O(N)$ to visit all the edges of the triangulation. Crossover has a time equal to $O(N^2)$ in the worst case in our implementation; however, this is a very pessimistic estimate. This complexity dominates and, therefore, the overall complexity is $O(N^2)$. However, with some more sophisticated data structure this time can be reduced to $O(N \log N)$ in the worst case and $O(N)$ in the expected case. We did not proceed in this direction as our concern was to test the usability of the genetic approach and not to achieve the best possible implementation in particular problems.

As is well known, in the asymptotic complexity, the constant terms are neglected. In the genetic approach, these omissions should be pointed out, because the time demands of the genetic approach are due to a large number of computed triangulations not due to the computation of one triangulation. If 50 generations per 50 members are computed, the neglected multiplication constant is 2500; in such

a case, genetic computation could be nearly 2500 times slower than 'deterministic' triangulation.

Let us briefly present the genetic approach as it is used in the literature available to the authors and to show pros and cons of already existing solutions. In [46] a triangulation was represented as a list of edges. Both crossover and mutation were performed as edge flips. The probability of mutation was 0.05. The population size was $35-75$. The initial population was prepared as randomly generated triangulations. There were two types of testing data: randomly generated points and points sitting on an arc of a circle ($N = 10-200$), which are known to be bad for greedy triangulation (*GT*) and are used for comparison [36]. Parallel implementation on a four-processor hypercube architecture has also been attempted. In general, the sequential genetically optimized triangulation outperformed the *GT* on point sets of size 130 or less. Parallel GO performed well also for larger datasets (110 points and more) and was the best for 70% of tested data sets. For the second type of data, *GO* was better than *GT* in all cases but one. However, these results were influenced by the fact that the data type is unfavourable for comparison with *GT*; it is not a completely fair way of testing. No time measurements are provided.

In [44] triangulation is represented by a lower triangular matrix in which the element m_{ij} is equal to 1 if the edge ij is present in the triangulation and 0 otherwise. If two matrices $\boldsymbol{M}_1$, $\boldsymbol{M}_2$ are exclusive-ored, the resulting matrix $\boldsymbol{M}$ has 1s on positions corresponding to the edges which are only in one triangulation, but not in the other triangulation. Crossover is performed by choosing a random $m_{ij} = 1$ in $\boldsymbol{M}$ corresponding to an edge in one triangulation (and not present in the other) and then finding the minimum polygon, such that it contains the vertices of the random edge and resides in both triangulations. Then edges inside this polygon are exchanged between the two triangulations. Mutation is done by edge flip. The terminal condition is the given number of generations or when all the members of the current generation have nearly the same value.

Results are presented on sets of points in [34] and in two other examples (maximal $N = 40$). Population size is recommended to be $30-60$, but no relation to N is presented. Values for p_c and p_m are taken to be $0.5-0.6$ and $0.001-0.1$, respectively. The disadvantage of this solution is memory requirements. Crossover is quite complicated but based on an interesting concept. However, results are not too persuasive.

In [11] an interesting, so-called weighted coding of triangulations is offered. It associates an integer weight with each point in the given data set. Point weight is added to the length of all the edges in which that point participates. However, fitness of the triangulation has to be computed from non-modified edge lengths. This coding is used in the greedy heuristic – edges are sorted in increasing order according to their modified lengths, and in each step, the shortest edge is inserted into the triangulation if it does not intersect any other already accepted edge. Mutation is perfomed by assignment a new random value to a point. One hundred generations per $10N$ individuals were used.

The algorithm was tested on seven problems of up to 50 points. Each data set was run 10 times. It was always better than greedy triangulation in the three smaller data sets, and it gave the same answer in the larger data sets in $4-7$ cases out of 10. It needed more than 5 h for the 50-point data set, compared with the fraction of a second needed for greedy triangulation.

This concludes the explanation of the genetic approach for triangulation. In the next section, attention will be given to the proper mathematical formulation of triangulation criteria.

4 Criteria for the triangulation optimization

Of the triangulation criteria presented earlier, two are used most often: minimum weight and maxmin angle. These two triangulations form in some sense the two poles to the problem: *DT* is the leading representative of angle criteria triangulations, and *MWT* is the most difficult case of the weight criteria.

The criterion of minimum weight leading to the minimum weight triangulation is, due to its globality, very difficult. The problem of *MWT* construction is neither known to be solvable in polynomial time nor proved to be *NP*-hard. It is one of the few problems in [23] whose complexity is still unknown. Existing methods either work only for some special cases, e.g., [2, 37], or find only a subgraph of the *MWT* [1, 5, 13, 27, 28]. Substantial research has also been devoted to the development of heuristics [4, 25, 32, 42].

Table 1. Weight criteria

Name	Description	Formula	Type of extreme
Min weight	f_{MW}	$f_{\mathrm{MW}} = \sum s_i * l_i, i = 1, 2, \ldots, nE$	Min
Min edge length	f_{MINE}	$f_{\mathrm{MINE}} = \min(l_i), i = 1, 2, \ldots, nE$	Max
Max edge length	f_{MAXE}	$f_{\mathrm{MAXE}} = \max(l_i), i = 1, 2, \ldots, nE$	Min

Table 2. Symbols

Symbol	Description
nE	Number of edges in $T(S)$
$nTris$	Number of triangles in $T(S)$
l_i	Euclidean length of an edge i
s_i	Weight coefficient for the edge i
α_i^j	jth angle of a triangle i
rcc_i	Radius of a circumcircle for a triangle i

The global and local maxmin angle criteria have been proved to be satisfied by the Delaunay triangulation (DT). This is the only a local extreme which guarantees that a global extreme will be found as well. Unlike an MWT, many algorithms exist to compute DT; some of them can be found, for example, in [3, 8, 9, 12, 21, 22, 40, 41, 43]. The optimal worst-case time complexity is $O(N \log N)$; using some accelerating data structures, the expected time can be reduced to $O(N)$. These two triangulations provide the fundamentals for the formulation of criteria which can be expected to bring good and interesting results. Several variants of weight criteria are given in Table 1. Symbols are explained in Table 2. Weight coefficients s_i in f_{MW} are used if it is more important for some edges to be shorter than others and they should be stressed in the sum. Table 3 shows the angle criteria. Apart from these formulae, we also used some clearly unsuitable criteria to demonstrate the versatility of the GO. They are not included in the tables in order not to be mixed with the 'reasonable' formulae. Therefore, they will be denoted only by abbreviation, e.g., max weight, etc. Experiences and results are given in the next section.

5 Results and discussion

Genetic optimization, interpolation and Delaunay triangulation were implemented in Delphi 3 under Windows NT and ran on PC PII 200 MHz/128 MB and 233 MHz/96 MB.

Of all the criteria in Table 1, we picked only f_{MW} for further research as, usually, there is not enough sensitivity for f_{MINE} and f_{MAXE} to measure differences between triangulations, as many meshes have the same value. The function f_{MW} diversifies much better. Experience with angle criteria was very much alike to Table 1: f_{MAMIA}, f_{MIMAA}, f_{MAMIR} and f_{MIMAR} are not sensitive enough to distinguish small differences between triangulations. Summation criteria f_{MAMIS}, f_{MIMAS} and f_{MSR} were more successful. Of these three, the least persuasive results were obtained with f_{MSR}, as can be seen in the figures below.

Figures 7–20 show typical results from use of the described method. The input point configurations are

Table 3. Angle criteria

Name	Description	Formula	Type of extreme
Maxmin angle	f_{MAMIA}	$f_{\mathrm{MAMIA}} = \min(\alpha_i^j), j = 1, 2, 3, i = 1, 2, \ldots, nTris$	Max
Minmax angle	f_{MIMAA}	$f_{\mathrm{MIMAA}} = \max(\alpha_i^j), j = 1, 2, 3, i = 1, 2, \ldots, nTris$	Min
Maxmin angle sum	f_{MAMIS}	$f_{\mathrm{MAMIS}} = \sum \min_i(\alpha_i^j), j = 1, 2, 3, i = 1, 2, \ldots, nTris$	Max
Minmax angle sum	f_{MIMAS}	$f_{\mathrm{MIMAS}} = \sum \max_i(\alpha_i^j), j = 1, 2, 3, i = 1, 2, \ldots, nTris$	Min
Maxmin rad	f_{MAMIR}	$f_{\mathrm{MAMIR}} = \min(rcc_i), i = 1, 2, \ldots, nTris$	Max
Minmax rad	f_{MIMAR}	$f_{\mathrm{MIMAR}} = \max(rcc_i), i = 1, 2, \ldots, nTris$	Min
Min sum of rad	f_{MSR}	$f_{\mathrm{MSR}} = \sum rcc_i, i = 1, 2, \ldots, nTris$	Min

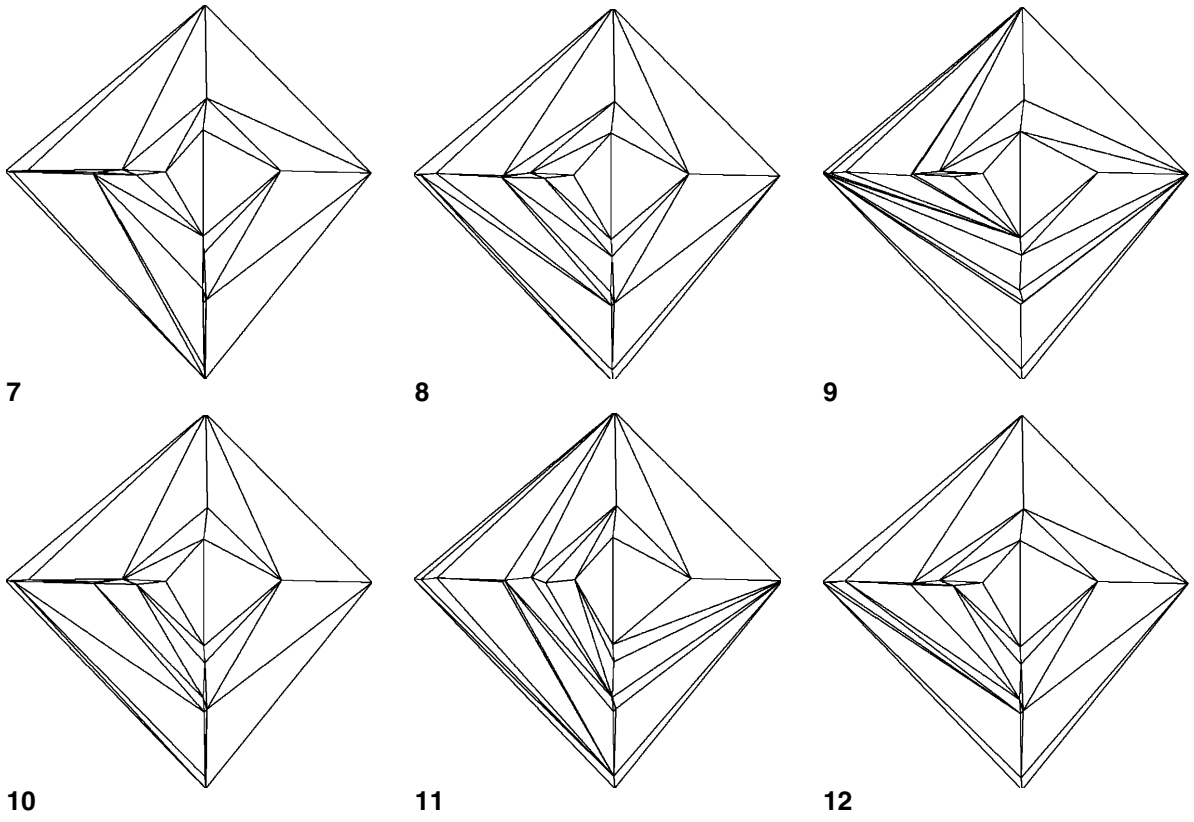


Fig. 7. *eval* = f_{MW}, $f_{MW} = 10.01$, $f_{MAMIS} = 402.64°$, $f_{MIMAS} = 4647.79°$, $f_{MAMIA} = 0.64°$, $f_{MIMAA} = 175.71°$, $f_{MAMIR} = 0.015$, $f_{MIMAR} = 1.012$, $f_{MINE} = 0.005$, $f_{MAXE} = 0.540$

Fig. 8. Delaunay triangulation, $f_{MW} = 11.06$, $f_{MAMIS} = 403.23$, $f_{MIMAS} = 4450.13°$, $f_{MAMIA} = 0.99°$, $f_{MIMAA} = 166.94°$, $f_{MAMIR} = 0.015$, $f_{MIMAR} = 0.420$, $f_{MINE} = 0.005$, $f_{MAXE} = 0.540$

Fig. 9. *eval* = maxweight, $f_{MW} = 13.56$, $f_{MAMIS} = 337.19°$, $f_{MIMAS} = 4425.97°$, $f_{MAMIA} = 0.14°$, $f_{MIMAA} = 172.35°$, $f_{MAMIR} = 0.095$, $f_{MIMAR} = 1.273$, $f_{MINE} = 0.005$, $f_{MAXE} = 0.534$

Fig. 10. *eval* = f_{MAMIS}, $f_{MW} = 10.35$, $f_{MAMIS} = 407.05°$, $f_{MIMAS} = 4553.98°$, $f_{MAMIA} = 0.64°$, $f_{MIMAA} = 175.71°$, $f_{MAMIR} = 0.015$, $f_{MIMAR} = 1.012$, $f_{MINE} = 0.005$, $f_{MAXE} = 0.540$

Fig. 11. *eval* = f_{MIMAS}, $f_{MW} = 12.87$, $f_{MAMIS} = 352.24°$, $f_{MIMAS} = 4274.24°$, $f_{MAMIA} = 0.69°$, $f_{MIMAA} = 153.48°$, $f_{MAMIR} = 0.015$, $f_{MIMAR} = 0.421$, $f_{MINE} = 0.005$, $f_{MAXE} = 0.534$

Fig. 12. *eval* = f_{MSR}, $f_{MW} = 11.08$, $f_{MAMIS} = 400.81°$, $f_{MIMAS} = 4450.31°$, $f_{MAMIA} = 0.86°$, $f_{MIMAA} = 166.94°$, $f_{MAMIR} = 0.015$, $f_{MIMAR} = 0.482$, $f_{MINE} = 0.00524$, $f_{MAXE} = 0.534$

slightly artificial but demonstrate the strength and versatility of the genetic approach.

Triangulations for Figs. 7–17 are computed for a data set S with 20 points, optimized with respect to f_{MW}, f_{MAMIS}, f_{MIMAS} and f_{MSR}. Both sum and multiplication types of the evaluation function were used. In the presented example, values of angles and weights are of similar order; therefore, no problems with sensitivity to too low values was detected. The data set was kept small for easy comparison of the resulting meshes.

In Fig. 7, the min weight criterion was used. Most of the triangles have a reasonable shape, with the exception of the third quadrant. However, due to the distribution of points on the negative coordinate axes, better results may not be obtained. Compare this with Fig. 8, which presents a Delaunay triangulation. *DT* has more "equalized" triangles – less very small, nar-

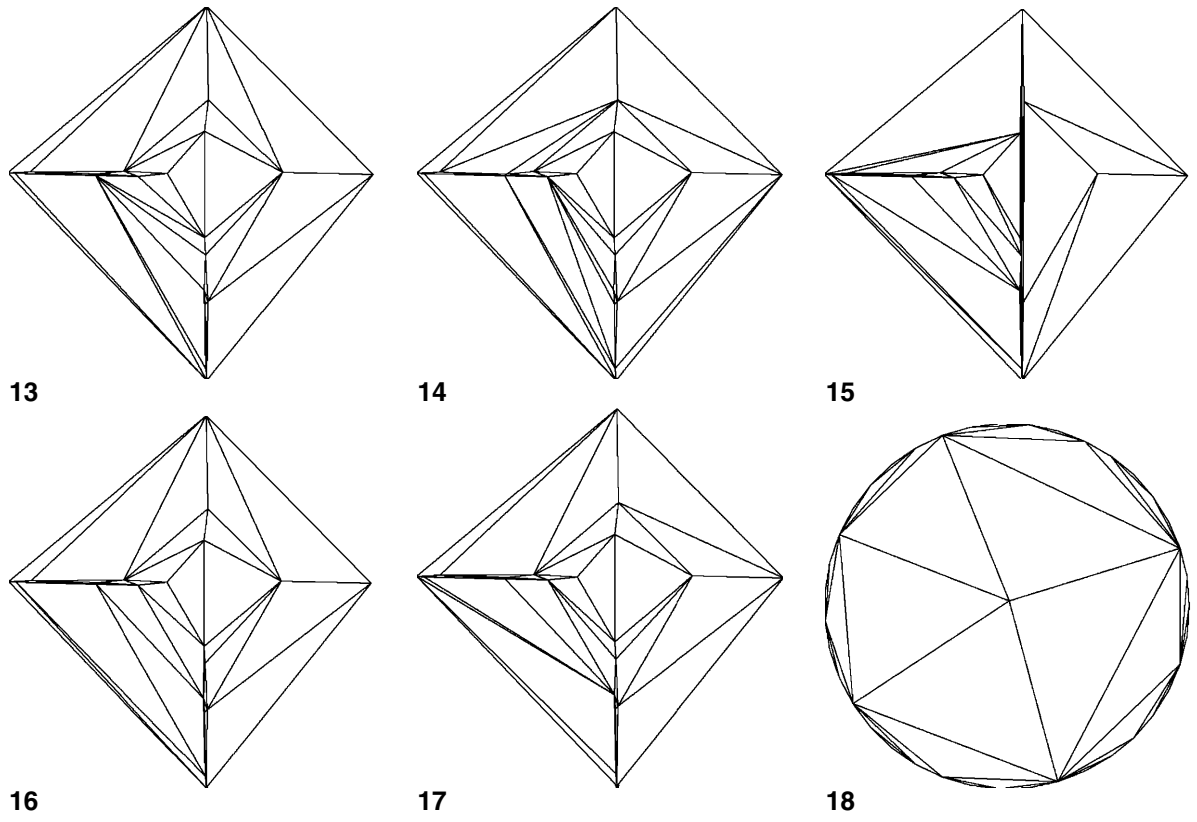


Fig. 13. $eval = 0.6 \times f_{MW} + 0.4 \times f_{MAMIS}$, $f_{MW} = 10.17$, $f_{MAMIS} = 404.11°$, $f_{MIMAS} = 4588.45°$, $f_{MAMIA} = 0.36°$, $f_{MIMAA} = 177.69°$, $f_{MAMIR} = 0.015$, $f_{MIMAR} = 1.817$, $f_{MINE} = 0.005$, $f_{MAXE} = 0.540$

Fig. 14. $eval = 0.5 \times f_{MW} + 0.5 \times f_{MSR}$, $f_{MW} = 10.74$, $f_{MAMIS} = 391.94°$, $f_{MIMAS} = 4524.80°$, $f_{MAMIA} = 0.99°$, $f_{MIMAA} = 169.74°$, $f_{MAMIR} = 0.015$, $f_{MIMAR} = 0.493$, $f_{MINE} = 0.005$, $f_{MAXE} = 0.540$

Fig. 15. $eval = 0.33 \times \text{maxweight} + 0.33 \times \text{maxmaxanglesum} + 0.33 \times \text{minminanglesum}$, $f_{MW} = 12.35$, $f_{MAMIS} = 298.17°$, $f_{MIMAS} = 5118.40°$, $f_{MAMIA} = 0.13°$, $f_{MIMAA} = 179.63°$, $f_{MAMIR} = 0.094$, $f_{MIMAR} = 20.093$, $f_{MINE} = 0.014$, $f_{MAXE} = 0.538$

Fig. 16. $eval = 0.33 \times f_{MW} + 0.33 \times f_{MIMAS} + 0.33 \times f_{MAMIS}$, $f_{MW} = 10.10$, $f_{MAMIS} = 407.40°$, $f_{MIMAS} = 4617.19°$, $f_{MAMIA} = 0.36°$, $f_{MIMAA} = 177.69°$, $f_{MAMIR} = 0.015$, $f_{MIMAR} = 1.817$, $f_{MINE} = 0.005$, $f_{MAXE} = 0.540$

Fig. 17. $eval = f_{MW} \times f_{MIMAS} \times f_{MAMIS}$, $f_{MW} = 10.08$, $f_{MAMIS} = 405.79°$, $f_{MIMAS} = 4596.47°$, $f_{MAMIA} = 0.64°$, $f_{MIMAA} = 175.71°$, $f_{MAMIR} = 0.015$, $f_{MIMAR} = 1.012$, $f_{MINE} = 0.005$, $f_{MAXE} = 0.534$

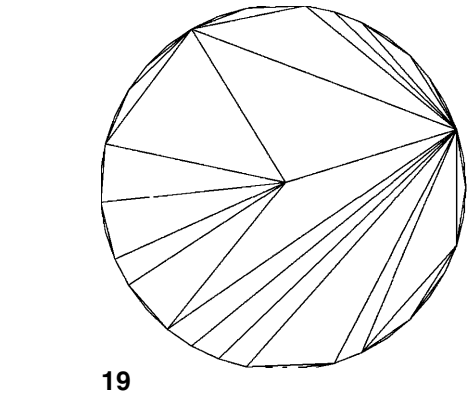
Fig. 18. $eval = f_{MW}$, $f_{MW} = 30.59$, $f_{MAMIS} = 480.73°$, $f_{MIMAS} = 5938.50°$, $f_{MAMIA} = 4.614°$, $f_{MIMAA} = 170.771°$, $f_{MAMIR} = 0.539$, $f_{MIMAR} = 1.000$, $f_{MINE} = 0.321$, $f_{MAXE} = 1.442$

row triangles is balanced by a deterioration in some other triangle shapes (mainly in the first and second quadrant) and by a visible increase in the triangulation weight. This observation is verified by the numerical values of criteria. For comparison, Fig. 9 shows a mesh obtained by maximizing triangulation weight. It is slightly ridiculous, as we optimized by using a criterion leading to "bad" shapes of triangles; however, it shows the strength of GO and the broad possibilities for its application. Figure 10 is a result of maxmin angle sum, Fig. 11 a result of minmax angle sum and Fig. 12 a result of min sum of rad. The second of these is the worst – it has the worst weight and a low min angle sum, while max angle sum is not bad. Of the other two, Fig. 10 looks to be the better.

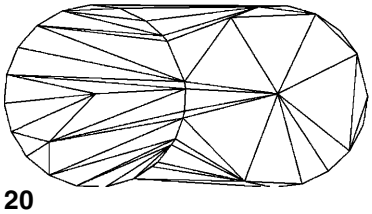
Let us continue showing the effects of mixed criteria (Figs. 13–17). Figure 13 is $0.6 \times f_{\mathrm{MW}} + 0.4 \times f_{\mathrm{MAMIS}}$. Figure 14 is $0.5 \times f_{\mathrm{MW}} + 0.5 \times f_{\mathrm{MSR}}$. Figure 15 is $0.33 \times \mathrm{maxweight} + 0.33 \times \mathrm{maxmax\,anglesum} + 0.33 \times \mathrm{minminanglesum}$. Figure 16 is $0.33 \times f_{\mathrm{MW}} + 0.33 \times f_{\mathrm{MIMAS}} + 0.33 \times f_{\mathrm{MAMIS}}$. Figure 17 was obtained using $f_{\mathrm{MW}} \times f_{\mathrm{MIMAS}} \times f_{\mathrm{MAMIS}}$. Compare Figs. 15 and 16 – Fig. 15 was again optimized "to be as bad as possible". Notice the long, narrow triangles around the y-axis. Both Figs. 16 and 17 show triangles with a relatively good shape. Optimization in Fig. 16 would be problematic if values of angle and weight parts of the function were of very different order; however, this is not the case here.

Figure 18 shows 39 points regularly distributed on a circle; one point is near the centre, triangulated with the min weight criterion. Figure 19 shows the same data set, but the min weight criterion was slightly modified: edges with angles to the $+x$-axis within the range $\langle 30°, 50° \rangle$ were favoured. (This was performed using an extra condition: if the slope of an edge is greater than or equal to 30° and equal to or less than 50°, then multiply the length of the edge by 0.1 – recall s_i in the definition of f_{MW} in Table 1.) Figure 20 demonstrates that a criterion can be different in different parts of the data. Here, the right part is optimized on f_{MW} and the left part on max weight. From these figures it can be seen that a difficulty in predicting the exact result and the success of a particular criterion are disadvantages of GO. However, this unpredictability is weaker if more criteria are combined. On the other hand, the generality of GO is attractive – the influences of different criteria may be combined and can be used either on the whole data set or on a subset; anisotropy can be included by utilizing some preferred direction. Therefore, GO seems to have high potential and flexibility in triangulation.

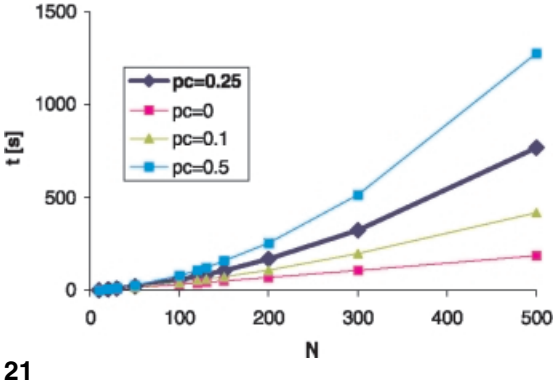
From this exposition of possibilities, let us proceed to more detailed results showing the behaviour of GO. The following group of results shows properties of the genetic operators described in Sect. 3. The maximized criterion is as follows: $eval = 0.33 \times f_{\mathrm{MW}} + 0.33 \times f_{\mathrm{MIMAS}} + 0.33 \times f_{\mathrm{MAMIS}}$. All results were obtained as an average for five data sets: two with a uniform point distribution, two with a Gaussian point distribution and one with an "eccentric" distribution of the type shown in Fig. 18. There were no substantial differences among results for various types of data; therefore, only averages are presented.



19



20



21

Fig. 19. $eval = f_{\mathrm{MW}}$, edges with angles to the $+x$-axis within $\langle 30°, 50° \rangle$ favoured. $f_{\mathrm{MW}} = 38.92$, $f_{\mathrm{MAMIS}} = 372.12°$, $f_{\mathrm{MIMAS}} = 5654.61°$, $f_{\mathrm{MAMIA}} = 4.614°$, $f_{\mathrm{MIMAA}} = 170.772°$, $f_{\mathrm{MAMIR}} = 0.511$, $f_{\mathrm{MIMAR}} = 1.710$, $f_{\mathrm{MINE}} = 0.321$, $f_{\mathrm{MAXE}} = 1.921$

Fig. 20. The left part optimized on max weight, the right part on min weight

Fig. 21. Computation time for 100 generations, each with 50 triangulations, $p_{\mathrm{c}} = 0, 0.1, 0.25, 0.5$; $p_{\mathrm{m}} = 0.001$

Tables 4, 5, 6 and 7 and Figs. 21 and 22 show results for various values of the probability of crossover (p_{c}) and the probability of mutation (p_{m}). They were computed for 100 generations per 50 triangulations, using the reduced crossover operator. The best eval-

Table 4. Computational times (s) for 100 generations, each with 50 triangulations, and various p_c ($p_m = 0.001$)

N	$p_c = 0$	$p_c = 0.1$	$p_c = 0.25$	$p_c = 0.5$
10	3.18	3.22	3.36	3.60
20	6.50	6.89	7.53	8.51
30	9.77	10.76	12.13	14.26
50	16.82	19.74	23.70	29.84
100	33.84	44.12	59.01	81.60
120	41.43	56.62	77.48	110.69
130	44.95	62.93	87.84	124.87
150	52.87	75.77	108.64	161.13
200	71.36	110.39	170.42	256.72
300	109.02	200.01	325.43	515.52
500	188.63	419.14	768.60	1276.96

Table 5. Percentage of cases in which iteration with the given value of p_c provided results better than, worse than or the same as iteration with $p_c = 0.25$; (100 generations, each with 50 triangulations, $p_m = 0.001$)

	$p_c = 0$	$p_c = 0.1$	$p_c = 0.5$
Better	12.00	33.20	54.60
Worse	69.80	51.80	24.20
The same	18.20	15.00	21.20

Table 6. Computational times (s) for 100 generations, each with 50 triangulations, and various p_m ($p_c = 0.25$)

N	$p_m = 0.001$	$p_m = 0$	$p_m = 0.01$	$p_m = 0.1$	$p_m = 1$
10	3.36	3.36	3.68	3.46	3.93
20	7.53	7.57	7.61	7.89	9.64
30	12.13	12.14	12.27	12.90	16.48
50	23.70	23.56	24.12	25.49	35.38
100	59.01	58.96	60.45	65.06	100.10
120	77.48	77.21	79.28	85.89	136.15
130	87.84	87.32	89.73	97.80	155.97
150	108.64	108.01	111.42	122.51	200.38
200	170.42	169.33	174.18	191.23	326.59
300	325.43	322.12	331.36	367.60	660.15
500	768.60	763.72	785.18	881.51	1697.56

Table 7. Percentage of cases in which iteration with the given value of p_m provided results better than, worse than or the same as iteration with $p_m = 0.001$ (100 generations, each with 50 triangulations, $p_c = 0.25$)

	$p_m = 0$	$p_m = 0.01$	$p_m = 0.1$	$p_m = 1$
Better	15.20	15.20	6.10	6.10
Worse	66.60	60.55	75.75	75.75
The same	18.20	24.25	18.15	18.15

uations and computation times are compared. Figure 21 and Table 4 are not too surprising; they show that the higher the probability of crossover (and thus the number of crossovers made), the higher the computational time. Table 5 shows that the higher the probability of crossover, the greater the number of successful cases. However, in order to keep the computational time reasonable, we decided that $p_c = 0.25$ was a reasonable compromise. Figure 22 and Tables 6 and 7 present reasons of our choice of $p_m = 0.001$ as the correct value. The results for different probability of mutation in Table 7 show very clearly that for a probability of mutation that is too high "good" solutions are broken by random changes and results deteriorate. On the other hand, results for $p_m = 0$ document that "no mutation" is not a good choice either.

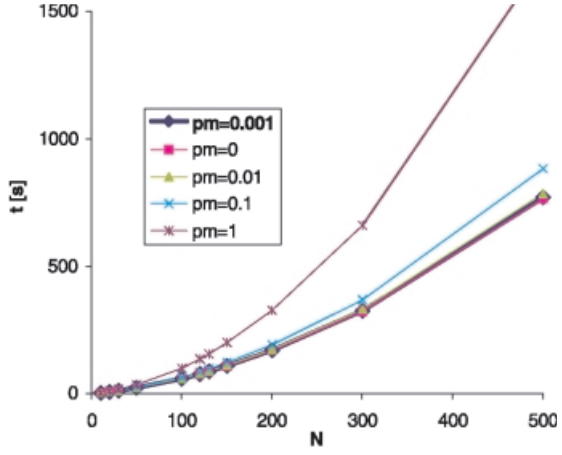
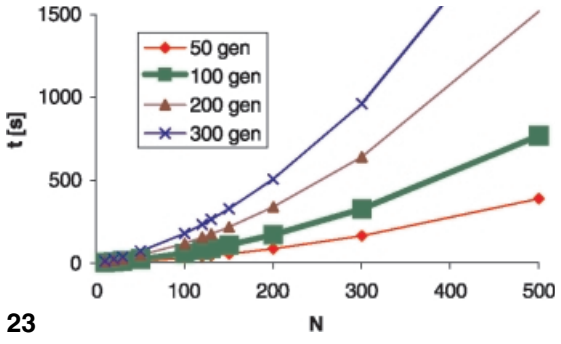


Fig. 22. Computation time for 100 generations, each with 50 triangulations, $p_c = 0.25$; $p_m = 0, 0.001, 0.01, 0.1, 1$

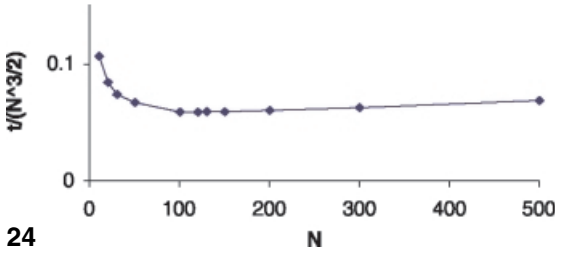
Figure 23 and Table 8 present different computation times for various numbers of generations (50 triangulations per generation, $p_c = 0.25$, $p_m = 0.001$). For most experiments, we picked 100 generations as a basis; higher numbers of generations are generally too slow.

Table 9 documents that, generally, it is more efficient to have more generations and less triangulations per generation than vice versa. Times of computation are not documented, as they are nearly the same, the lower number of generations being slightly quicker.

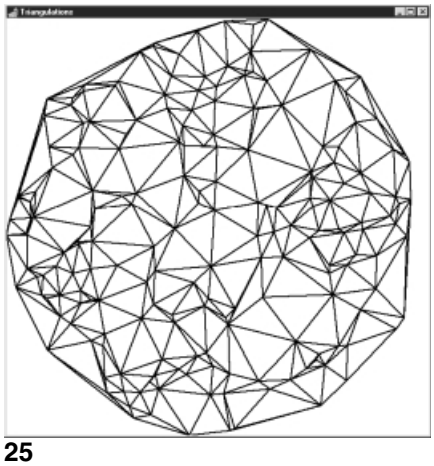
Table 10 shows that the higher the number of generations, the better the solutions found. However,



23



24



25

Fig. 23. Computation time for different number of generations; 50 triangulations per generation, $p_m = 0.001$, $p_c = 0.25$

Fig. 24. Time complexity measured for 100 generations, each with 50 triangulations, $p_m = 0.001$, $p_c = 0.25$

Fig. 25. Example of the triangulation on 200 points, distributed uniformly in a unit circle, optimized on $0.25 \times f_{MW} + 0.25 \times f_{MIMAS} + 0.25 \times f_{MAMIS} + 0.25 \times f_{MSR}$

time demand has to be taken into account; it prevents a very large number of generations. Figure 24 documents that measured time complexity is better than $O(N^2)$ – approximately $O(N^{3/2})$.

Table 8. Computation times (s) for different number of generations, each with 50 triangulations ($p_c = 0.25$, $p_m = 0.001$)

N	50 gen	100 gen	200 gen	300 gen
10	1.659	3.358	6.74	10.141
20	3.762	7.530667	15.136	22.676
30	6.059	12.13433	24.258	36.449
50	11.844	23.69733	47.388	70.952
100	29.626	59.011	118.053	177.655
120	38.949	77.48167	154.936	231.65
130	44.411	87.83933	174.888	263.299
150	54.308	108.643	217.075	326.737
200	85.739	170.4217	338.66	507.069
300	163.842	325.431	639.843	961.586
500	387.594	768.602	1517.598	2270.05

Table 9. Percentage of cases in which 100 generations, each with 50 triangulations, is better than 50 generations, each with 100 triangulations ($p_m = 0.001, p_c = 0.25$)

100 gen. × 50 tr. is better	48.50
100 gen. × 50 tr. is worse	33.33
Both the same	18.17

Table 10. Percentage of cases in which more than 100 generations found a better solution than 100 generations (50 triangulations per generation, $p_c = 0.25$, $p_m = 0.001$)

	50 gen.	200 gen.	300 gen.
Better than 100 gen.	0	75.00	78.85
Worse than 100 gen.	66.75	0	0
The same	33.25	25.00	21.15

Figure 25 shows an example of generated triangulation on uniform random data optimized with $0.25 \times f_{MW} + 0.25 \times f_{MIMAS} + 0.25 \times f_{MAMIS} + 0.25 \times f_{MSR}$.

Let us sum up the results. The most important argument against GO is time requirement. Because instead of one triangulation, several thousands are generated, computation time inevitably must be much worse than if a deterministic triangulation algorithm, computing only one mesh, is used. In our opinion, GO is especially suitable in a situation where quality is preferred to speed. However, if processing time is substantial, GO cannot compare with the 'classical' deterministic algorithm, e.g., for 500 data points, a Delaunay triangulation can be computed in less than 1 s, while GO needs about 12 min. Some increase in speed might be achieved by parallelization, as members of one generation could be distributed

among processing elements, a natural synchronization point being the end of the computation of a new generation. The most important argument for GO is its versatility – its application is independent of the type of criterion. In other words, GO contains an infinite number of particular triangulators.

6 Conclusion

This paper introduces a new category of multicriteria-optimized triangulations and presents how the genetic approach can be used to obtain them. Combination of weight and angle criteria in one triangulation may bring new possibilities for geometrical modeling and data visualization. It is also possible to compute triangle meshes, which are optimized differently in different subareas. The method also enables a particular direction to be preferred, which may be useful in terrain modeling (the so-called data-dependent triangulations [10, 15]). The main weakness of the proposed method is time demand. However, if quality is preferred to speed, the method may provide an interesting tool for unusual triangulations.

Acknowledgements. Authors would like to thank to Prof. Dr. V. Skala from the University of West Bohemia in Pilsen, Czech Republic for providing the environment in which this work has been possible and to anonymous referee for substantial encouragement and set of particular ideas how to improve the quality of the paper. This work was supported by the Ministry of Education of The Czech Republic – project VS 97 155 and project GA AV A2030801.

References

1. Aichholzer O, Aurenhammer F, Hainz R (1998) New results on MWT subgraphs. TR No. 140, 1998, Institute for Theoretical Computer Science, Graz University of Technology
2. Anagnostou E, Corneil D (1993) Polynomial-time instances of the MWT problem. Comput Geom Theor Appl 3:247–259
3. Aurenhammer F (1991) Voronoi diagrams – a survey of a fundamental geometric data structure. ACM Comput Surv 23(3):345–405
4. Bartánus M, Ferko A, Mag R, Niepel L, Plachetka T, Šikudová E (1996) New heuristics for Minimum Weight Triangulation. In: WSCG 96, Conference Proceedings, University of West Bohemia, Pilsen, pp 31–40
5. Beirouti R, Snoeyink J (1998) Implementations of the LMT heuristic for minimum weight triangulation. Proc 14th Annual Symposium Comput Geom, Minneapolis, ACM, pp 96–105
6. Bern M, Edelsbrunner H, Eppstein D, Mitchell S, Tan TS (1993) Edge insertion for optimal triangulations. Discrete Comput Geom 10:47–65
7. Bern M, Eppstein D (1995) Mesh generation and optimal triangulation, 2nd edn. In: Lecture Notes Series on Computing, vol 4, World Scientific, pp 47–123
8. de Berg M, van Kreveld M, Overmars M, Schwarzkopf O (1997) Computational geometry. Algorithms and applications. Springer, Berlin Heidelberg
9. Brown KQ (1979) Voronoi diagrams from convex hulls. Inform Proc Lett 9(5):223–228
10. Brown JL (1991) Vertex based data dependent triangulations. Comput Aided Geom Des 8:239–251
11. Capp K, Julstrom BA (1998) A weight-coded genetic algorithm for the minimum weight triangulation problem. In: Carroll J, Lamont GB, Oppenheim D, George KM, Bryant B (eds) Applied computing 1998: Proceedings of the 1998 ACM Symposium on Applied Computing, ACM, New York, pp 327–331
12. Cignoni P, Montani C, Scopigno R (1992) A Merge-first divide & conquer algorithm for Ed Delaunay triangulations. Intern Rep C92/16, CNUCE/CNR, Pisa
13. Dickerson MT, Montague MH (1996) A (usually ?) connected subgraph of the minimum weight triangulation. Proc ACM 12th Symp Comput Geom, Philadelphia, pp 204–213
14. Duppe RD, Gottschalk HH (1970) Automatische Interpolation von Isolinien bei willkürlichen Stützpunkten. Allg Vermessungsber 77:423–426
15. Dyn N, Levin D, Rippa S (1990) Data dependent triangulations for piecewise linear interpolation. IMA J Numer Anal 10:137–154
16. Edelsbrunner H, Tan TS, Waupotisch R (1990) An $O(n2 \log n)$ time algorithm for the MinMax angle triangulation. Proc. 6th ACM Symp Comput Geom, Berkley, pp 44–52
17. Edelsbrunner H, Tan TS (1991) A quadratic time algorithm for the minmax length triangulation. SIAM J Comput 22:527–551
18. Edelsbrunner H, Tan TS, Waupotisch R (1992) An $O(N^2 \log N)$ time algorithm for the minmax angle triangulation. SIAM J Stat Sci Comput 13, 1992:994–1008
19. Edelsbrunner H, Tan TS (1993) A quadratic time algorithm for the minmax length triangulation. SIAM J Comput 22(3):527–551
20. Eppstein D (1992) The farthest point Delaunay triangulation minimizes angles. Comput Geom Theory Appl 1:143–148
21. Fang T-P, Piegl LA (1992) Algorithm for Delaunay triangulation and convex-hull computation using a sparse matrix. CAD 24(8):425–436
22. Fang T-P, Piegl LA (1993) Delaunay triangulation using a uniform grid. IEEE Comput Graph Appl 5:36–47
23. Garey MR, Johnson DS (1979) Computers and intractability. Freeman, San Francisco
24. Goldberg DE (1989) Genetic algorithms in search, optimization and machine learning, Addison-Wesley, New York
25. Heath LS, Pemmaraju SV (1994) New results for the MWT problem. Algorithmica 12:533–552
26. Holland JH (1975) Adaptation in Natural and Artifical Systems. University of Michigan Press, Ann Arbor
27. Keil JM (1994) Computing a subgraph of the minimum weight triangulation. Comput Geom 4:13–26
28. Kolingerová I, Magová I, Ferko A, Niepel L (1997) Better subgraph of minimum weight triangulation. In: Spring Con-

ference on Computer Graphics SCCG 1997, Conference Proceedings, Comenius University, Bratislava, pp 49–56
29. Kolingerová I (1998) Genetic approach to the minimum weight triangulation. In: WSCG'98 Conference Proceedings, Vol. II, University of West Bohemia, Pilsen, pp 184–191
30. Kolingerová I (1998) Genetic optimization of the triangulation weight. In: 3IA '98 International Conference Proceedings, Technical University of Limoges, Limoges, pp 23–34
31. Krznaric D (1997) Progress in hierarchical clustering & minimum weight triangulation. PhD Thesis, University of Lund
32. Kyoda Y, Imai K, Takuchi F, Tajima A (1997) A branch-and-cut approach for minimum weight triangulation. In: Proceedings of Algorithms & Computations, 8th International Symposium ISAAC '97, Lecture Notes on Computer Science 1350. Springer, Berlin, Heidelberg, pp 384–393
33. Lawson CL (1977) Software for C^1 interpolation. In: Rice JR (ed) Mathematical Software III, Academic, New York, pp 161–194
34. Levcopoulos C (1986) An $\Omega(\sqrt{N})$ lower bound for the nonoptimality of the greedy triangulation. Inform Proc Lett 22:25–31
35. Magová I, Ferko A, Niepel L (1997) On edges elimination for the shortest mesh. In: WSCG'97 Conference Proceedings, University of West Bohemia, Pilsen, pp 396–403
36. Manacher GK, Zobrist AL (1979) Neither the greedy nor the Delaunay triangulation of a planar point set approximates the optimal triangulation. Inform Proc Lett 9(1):31–34
37. Meijer H, Rappaport D (1992) Computing the MWT of a set of linearly ordered points. Inform Proc Lett 42:5–38
38. Michalewitz Z (1996) Genetic algorithms + data structures = evolution programs. Springer, Berlin, Heidelberg
39. Midtbo T (1993) Spatial modelling by Delaunay networks of two and three dimensions.
Available at http://www.iko.unit.no/tmp/term.html
40. Okabe A, Boots B, Sugihara K (1992) Spatial tesselations: concepts and applications of Voronoi diagrams. Wiley, Chichester
41. O' Rourke J (1994) Computational geometry in C. Cambridge University Press, New York
42. Plaisted DA, Hong J (1987) A heuristic triangulation algorithm. J Algorithms 8:405–437
43. Preparata FP, Shamos MI (1985) Computational geometry: an introduction. Springer, Berlin Heidelberg New York
44. Qin K, Wang W, Gong M (1997) A genetic algorithm for the minimum weight triangulation. In: Proc. 1997 IEEE Int. Conf. Evol. Comput., Piscataway, IEEE, pp 541–546
45. Schumaker LL (1993) Computing optimal triangulations using simulated annealing. Comput Aided Geom Des 10:329–345
46. Wu Y, Wainwright RL (1993) Near-optimal triangulation of a point set using genetic algorithms. In: Proc. 7th Oklahoma Symp. Artificial Intelligence, USA, pp 121–131
47. Yang B-T, Xu Y-F, You Z-Y (1994) A chain decomposition algorithm for the proof of a property on minimum weight triangulations. In: Du D-Z, Zhang X-S (eds) Algorithms and computation. Springer, Berlin, Heidelberg, pp 423–427

IVANA KOLINGEROVÁ graduated in Computer Science in 1987, received her PhD in Informatics and Computer Science in 1994 and habilitated in 2000. She works as an associate professor in the Department of Computer Science and Engineering at the University of West Bohemia in Pilsen, Czech Republic. Her research interests include computer graphics and computational geometry, especially application of non-traditional mathematical methods. Her international experience includes Denmark, U.S.A. and Slovenia.

ANDREJ FERKO studied numerical analysis and obtained his CSc. (PhD equivalent) at Comenius University in integrated circuit design. Currently, he is lecturing professor with the Department of Computer Graphics and Image Processing at Comenius University in Bratislava, Slovakia. His current research interests include computer graphics, standardization and multimedia, computational geometry (meshing). He is a Slovak representative for ISO/IEC JTC1/SC24 Information Processing – Computer Graphics and Image Processing (O-membership). Annually he co-organizes the Spring Conference on Computer Graphics and Central European Student Seminar on Computer Graphics.