Mesh Simplification for 3D Modeling using Evolutionary Multi-Objective Optimization

B. Rosario Campomanes-Álvarez[1], Sergio Damas[1], Óscar Cordón[1, 2, 3]

[1]Fuzzy and Evolutionary Algorithms Research Unit
European Centre for Soft Computing
Mieres, Asturias, Spain

[2]Dept. of Computer Science and Artificial Intelligence
University of Granada
Granada, Spain

[3]Research Center on Information and Communication Technologies
University of Granada
Granada, Spain

Abstract— **Polygonal surface models are typically used in three-dimensional (3D) visualizations and simulations. They are obtained by laser scanners, computer vision systems or medical imaging devices to model highly detailed object surfaces. Surface mesh simplification aims to reduce the number of faces used in a 3D model while keeping the overall shape, boundaries, and volume. In this work, we propose to deal with the mesh simplification problem from an evolutionary multi-objective viewpoint. The quality of a solution is defined by two conflicting objectives: the accuracy and the simplicity of the model. The Non-dominated Sorting Genetic Algorithm II (NSGA-II) is adapted to tackle the problem. We compare the NSGA-II performance with a classical approach and a single-objective implementation. The comparison has been carried out using different datasets.**

Keywords-component; Delaunay triangulation; 3D modeling; evolutionary multi-objective optimization; mesh simplification

I. INTRODUCTION

Polygonal surface models are the representation of three-dimensional visualizations and simulations. They are obtained by laser scanners, computer vision systems or medical imaging devices to model highly detailed object surfaces. These surface models are used in many different areas such as computer vision, computer-aided design, medicine, topography, etc.

Typically, a model surface is composed of thousands of polygons. The size of the files causes long processing times. Obtaining a reduced model with a smaller polygonal surface and a similar accuracy is a challenge in the area [9].

Surface mesh simplification is the process that aims to reduce the number of polygons used in a surface while preserving the overall shape, volume, and boundaries as much as possible.

There are several techniques in order to simplify a mesh. The decimation [11], [13], [22] and the energy function optimization [18] are the most popular and classical methods. There are some other recent proposals considering other ways to optimize 3D surfaces with evolutionary algorithms [9], [19], [25].

There is no work on mesh simplification using multi-objective evolutionary algorithms (MOEAs) that can lead to different kind of problems. In particular, file size. Thus, the formulation of mesh simplification as a multi-objective problem, the combination of classical techniques with a genetic algorithm, and the comparison with a classical method is an interesting research line which has been studied in this work. In particular, we decided to use a computationally fast and elitist multi-objective evolutionary algorithm based on a non-dominated sorting genetic approach (NSGA-II) proposed in [7]. Our methodology is based on *the simplification of a 3D open model* by an evolutionary multi-objective algorithm. An open model refers to a surface with open ends. The problem is based on the location of a certain number of points in order to approximate a mesh as accurately as possible to the initial surface. It will consider two conflicting objectives: the accuracy and the simplicity of a mesh, which are conflicting in nature. The orthogonal array crossover (OAX) function [19], [20] will be tested for the problem.

We will perform a comparison between the proposed multi-objective method, the developed one by Huang et al. in [19] which uses a single-objective algorithm to simplify 3D facial meshes, and the classical approach edge decimation [22].

This work is structured as follows. Section 2 introduces a short survey regarding classical techniques for mesh simplification and some others that consider evolutionary algorithms to simplify a surface. Section 3 describes all the components of the proposed approach. Section 4 presents the performed experiments and the results obtained. Finally, Section 5 concludes the whole work and provides some guidelines for future works.

II. STATE OF THE ART IN MESH SIMPLIFICATION

There are many mesh simplification approaches [4]. They can be either local or global. The former simplifies a mesh by the repeated use of some local operator iteratively. The latter are applied to the input mesh as a whole. The following two subsections review local and global families of methods respectively. Finally, subsection II.C presents the evolutionary approaches to the problem.

A. Incremental Methods Based on Local Updates

These methods run the simplification process as a sequence of local updates. Each update reduces the mesh size and decreases the surface approximation accuracy.

The decimation method can be classified into three different approaches [11], [13], [22] according to the difference of the selected objects: removal of vertex [22], removal of edge [17] and removal of triangle [14]. This method makes several iterations over all the vertices, edges or faces depending on the type of the decimation approach. If a vertex/edge/face satisfies a decimation criterion then it is removed. The obtained hole is patched by local triangulation. The algorithm finishes when it reaches a certain termination condition. The method has a simple implementation, achieves a good approximation and preserves the topology of the original mesh. However, it does not use an evaluation of error.

Other iterative method is the energy function optimization approach [18]. The mesh reduction is iteratively obtained by performing legal moves on mesh edges: collapsing, swapping or splitting. The approximation quality of the reduced mesh M is evaluated with a function which is composed by the sum of the squared distances of the original points from M, a factor proportional to the number of vertex in M, and the sum of the edge lengths. At each step, the element whose elimination causes the lowest increase in the energy function is deleted.

The progressive meshes method [17] is an enhanced version of the previously mentioned technique. This method obtains high quality results. However, it is not easy to implement and use.

On the other hand, the quadric error metric algorithm [11] is based on the iterative contraction of vertex pairs. It uses a value called remove cost in order to determine the points to be removed.

B. *Non Incremental Methods*

The coplanar method [16] uses a named detecting plane to determine whether a vertex is near enough. If the distance between a vertex and the detecting plane is small and it lies within the specified bounded threshold, the vertex will join the plane. The method preserves the geometry but the evaluation of the error approximation is highly inaccurate and it is not bounded.

The re-tiling method [23] starts with a polygonal surface and creates a triangulation of it with a user-specified number of vertices. The number of polygons shared by any given edge is the main restriction. The method is suitable for polygonal models in which each edge is shared by either one or two polygons. If a model satisfies this restriction, the algorithm will produce a new model with the same topology. This method removes part of the original vertices and it finally makes a local re-triangulation.

The clustering technique [21] is based on the *octree* concept. An *octree* is a tree structure where each branch or node has eight offspring who are the vertices of the mesh. This method is based on replacing the stored vertices in a node with only one.

Finally, the algorithms based on wavelets [12], [15] and the simplification using envelopes [6] provide tight error bounds on arbitrary triangulated meshes while allowing topological changes during the simplification.

C. *Mesh Simplification Approaches Based on Evolutionary Algorithms*

There are a few studies based on applying evolutionary computation to deal with the mesh simplification problem.

In [9], Fujiwara and Sawai tackled the problem of approximating a human facial surface by constructing a triangular mesh with a limited number of sample points. They developed a single-objective genetic algorithm that selects a given number of points from the whole dataset. Points are located in such a way that the resulting polygonal mesh approximates the original surface as closely as possible. The authors used 3D facial surfaces models.

Huang and Ho [19] proposed an evolutionary algorithm as an extension of Fujiwara and Sawai. The improvement is based on using the orthogonal array crossover. This algorithm also works with open surfaces models.

Finally, Xiandong et al. proposed a method of triangular mesh reduction based on a new concept called super-face and using a genetic algorithm in [25]. They used the STL format (which is a triangular representation of a 3D surface geometry) to manage the data structure.

III. EVOLUTIONARY 3D MULTI-OBJECTIVE MESH SIMPLIFICATION

In this Section, we formulate the mesh simplification problem. We detail a first proposal based on the Huang and Ho method and, finally, we describe our multi-objective technique.

A. *Problem Formulation*

Image data are generally represented using the standard cylindrical coordinate system (ρ, θ, ζ) (CCS) where the $\rho = 0$ axis runs vertically through the middle of the head, θ is the yaw angle, and ζ axis is the vertical axis. A point on the surface can be represented by a triplet (ρ, θ, ζ). It is easy to spread out the surface over $\mathbb{R}^2$ as $x = \theta, y = \zeta$ and $z = \rho$ [19], as it shows in Fig. 1 [29].

Let M be a scanned 3D model, it is possible to reduce this polygonal surface to a two dimensional problem. A three dimensional surface of this kind can be represented by the function:

$$f: (x, y) \in \mathbb{R}^2 \rightarrow z \in \mathbb{R} . \quad (1)$$

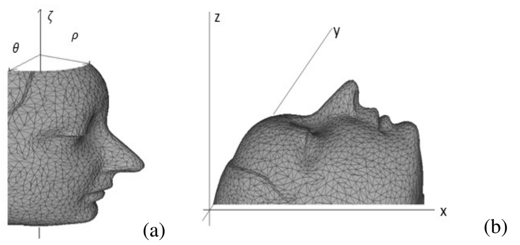

Figure 1. Surface representations. (a) Cylindrical coordinate system representation. (b) Grid plane representation.

Therefore, we need to locate *n* points which are less than *N* (the number of points of the original mesh). The number of points to be located can be either fixed during the task or chosen by the algorithm. Our experiments are based on this second option, i.e., the algorithm attempts to solve the problem with few data.

The previous methods in Section II use the Delaunay triangulation [1]. Given a set of points P in the plane, a Delaunay triangulation is a triangulation DT(P) such that no point in P is on the circumscribed circle of any triangle of DT(P). Delaunay triangulations maximize the minimum angle of all angles of the triangle; they tend to avoid skinny triangles.

So, let P_n, be an initial set of n points in $\mathbb{R}^2$, its corresponding Delaunay triangulation is denoted by $DT(P_n)$.

In our problem, a chromosome encodes a configuration P_n of n points in the 3D space. The genotype space consists of those 3D configurations:

Let a population of N individuals, each individual represents a certain mesh configuration. An individual is a simplified mesh, i.e., a mesh with fewer points than the reference model one. It will use a global array with the original mesh coordinates, and a binary chromosome array, with length n *(the number of points which will be located on the new simplified mesh)* for each individual.

The members of the population share a global array with the original mesh coordinates. Each position of the array stores the x, y and z coordinates. Every individual is defined by a binary chromosome with n genes. A non-zero value of the i_{th} position of the chromosome array means that the i_{th} vertex of the original model remains in the simplified mesh represented by such chromosome. By contrary, a zero means that there is not a point on the grid plane for this position.

The four points of the mesh corners are included in all the chromosomes. Hence, it avoids an abnormal boundary shape because of maintaining the rectangular shape as a whole [9], [19].

The algorithm performs the following simplification process: it converts the original 3D model of N points (P_N) into a 2D model with the same number of points. It carries out simplification and obtains a new 2D mesh with n points (P_n), with $P_n < P_N$.

It applies Delaunay triangulation to the new 2D mesh getting $D(P_n)$. This 2D triangulated final mesh $D(P_n)$ is converted into 3D and it obtains a final 3D model with n points. The algorithm selects the model that it is the best approximation of the original, i.e. the lowest error mesh. The following subsection explains how to calculate the error between two meshes, the original and the approximation one.

B. Objectives to Be Optimized

We have considered two objectives to be optimized, accuracy and simplicity. The former is guided by the minimization of an error metric. The latter is given by the number of triangles in the mesh. Therefore, we aim to minimize both objectives.

We have followed the same procedure presented by Huang et al. in [19] to calculate the approximation error. Each Delaunay triangle $T_i \in D(P_n)$ contains a certain number of grid points (x, y). The distance d_p (2) at each sample point p is defined as follows:

$$d_p = \left| z_p - \tilde{z}_p \right| , \qquad (2)$$

where z_p is the height value of the surface at the point p and $\tilde{z}_p$ is the linearly interpolated value of height at p determined by the triplet of heights for the three grid points of triangle T_i. The error e_i (3) for T_i is the sum of these distances over all the sample points p inside T_i where:

$$e_i = \sum_{p \epsilon T_i} d_p . \qquad (3)$$

The total approximation error e is defined by

$$e = \sum_{T_i \epsilon D(P_n)} e_i . \qquad (4)$$

Therefore, given a triangle of the Delaunay triangulation T_i the equation of its plane is calculated using the three triangle vertices. We selected the points of the original mesh which are into the triangle. Then, the z coordinate is estimated in the plane for each point inside the triangle. The error is the Euclidean distance between the original z coordinate of p, z_p, and the z coordinate of p inside the triangle Ti, $\tilde{z}_p$. Finally, the overall error is the sum of these distances over all triangles of the Delaunay triangulation.

C. Recombination

We have used the orthogonal array crossover recombination based on the Taguchi matrices [20] and developed in [19] in order to obtain more diversity than using others recombination operators.

D. Extension of a Single-Objective Genetic Algorithm

As a first approximation to the multi-objective problem, we will extend the Huang and Ho proposal in order to compare it with our method.

This algorithm consists of several functions such as, population initialization, selection scheme, genetic operations and termination criterion. It is based on the Huang and Ho proposal in [19]. Each gene has a phenotype given by its Delaunay triangulation $D(P_n)$. The fitness function has been adapted in order to consider the explained two objectives:

$$F(m) = wE(m) + (1 - w)T(m), \qquad (5)$$

where m is the mesh, $w \in [0,1]$ is a weight, E(m) is the error determined by (4) to be minimized and T(m) is the number of triangles of the final mesh.

The extension of the original method to tackle the fitness function in (5) is based on an evaluation function combining several normalized objectives into a weighted sum function [2], [5], [8]. It generates a set of Pareto optimal solutions giving different weights to the function and running repeatedly the algorithm. The method pseudo-code of the extended single-objective algorithm is as follows:

Population initialization: The population is initialized by randomly locating n points to each individual. The four points of the mesh corners are fixed to maintain the boundaries.

Selection scheme: It uses an elitist selection model in such a way that the individual with better fitness is selected to reproduce.

Mutation: It makes mutation in each gen with a probability p_m = 1/chromosome length.

Crossover: From two randomly chosen parents, we generate two offspring using an orthogonal array crossover operator [19], [20].

Termination criterion: Given by a maximum number of fitness evaluations.

E. The Evolutionary Multi-Objective Proposal

Our multi-objective algorithm proposal is based on NSGA-II [7]. As said, two conflicting objectives are considered: accuracy and simplicity. The former is guided by the minimization of an error metric. The latter is given by the number of triangles in the mesh. Therefore, we aim to minimize both objectives.

Given a mesh m for the multi-objective method, the fitness function is the following:

$$\min FM^1(m) = E(m)$$

$$\min FM^2(m) = T(m). \qquad (6)$$

The method scheme is detailed below:

Population initialization: This task is the same than the single-objective algorithm population initialization.

Selection scheme: The algorithm combines the current population with the obtained offspring using recombination in order to generate the next population. The best individuals according to non-dominance and diversity are selected to be reproduced.

Mutation: A gen is mutated with a probability p_m = 1/chromosome length.

Crossover: From two randomly chosen parents, we generate two offspring using an orthogonal array crossover operator [19], [20].

Termination criterion: Given by a maximum number of fitness evaluations.

IV. Experiments

We have used four datasets to accomplish all the simplification experiments. They are synthetic meshes, Laurana.ply, Cheff.ply, Ramses.ply and Duck.ply. These models are provided courtesy of the AIM@SHAPE Shape Repository [29]. All of them are open 3D models (Fig. 2). Table I presents the name of the mesh, the code in the experiments and the number of vertices and triangles which constitute each mesh.

TABLE I. DIMENSION OF THE PROBLEM DATASETS

Datasets Information		
Dataset Code and Name	***Vertices***	***Faces or triangles***
M1- Laurana.ply	922	1667
M2- Cheff.ply	2622	4864
M3- Ramses.ply	1420	2734
M4- Duck.ply	1047	2044

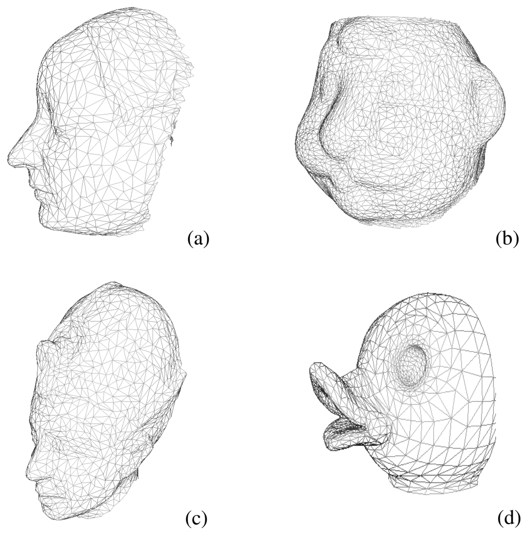

Figure 2. Original meshes. (a) Laurana.ply-M1. (b) Cheff.ply-M2. (c) Ramses.ply-M3. (d) Duck.ply-M4.

A. Experimental Setup

The algorithms have been implemented in C/C++ and all experiments have been performed on an Intel Core 2 Quad CPU Q8400 2.66 GHz, with 4 GB RAM, running Windows 7 Professional.

The parameters used are as follows: size population= 100, number of generation= 50, crossover probability= 0.8 and mutation probability= (1/chromosome length).

The multi-objective method has been run ten times with different seeds. Different weight vectors have been considered for the single-objective problem. The weight of the first objective function, the Delaunay triangle error ranges from 1 to 0 (step 0.1), and the number of triangles weight from 0 to 1 in the same step. It has resulted in 11 weighted vectors per objective. The algorithm has been run for each of the eleven weighted vectors and the obtained solutions have been grouped together to compose a Pareto-front approximation.

In the case of the classic method, 10 different simplifications have been performed changing the reduction percentage, i.e., from 5% to 50% of reduction with a 5% step. A reduction percentage means that the algorithm reduces the mesh into a certain number of faces or triangles. Therefore, the classic method will provide ten solutions which compose the Pareto-front.

In order to compare the behavior of different algorithms, the optimal Pareto-front is usually considered. However, in most real-world problems, the optimal Pareto-front is unknown and it cannot be calculated in reasonable time. It is the case of this study.

Hence, we have considered (in the single and multi-objective algorithms) the Pareto-front approximation which is obtained from the aggregation of the set of solutions P produced by each method in all the performed runs.

The Pareto-front approximation of the single-objective algorithm is calculated merging the eleven solutions obtained by each weighted vector. Repeated solutions are removed, and a later domination checks using the Pareto dominance definition is performed to produce the Pareto-front approximation.

In order to calculate the Pareto-front approximation for the multi-objective algorithm, the solutions obtained by ten runs are merged. Then, the repeated ones are removed, and finally the Pareto dominance is applied in order to achieve the aggregated Pareto-front approximation.

In the case of the classic method, the Pareto-front approximation is calculated joining the ten calculated solutions. At first, the ten solutions are merged. Then it removes the repeated ones, and finally, it applies the Pareto dominance to obtain the final Pareto-front approximation.

We have used three of the usual multi-objective performance indicators to measure the quality differences between two sets of solutions. One of them is the *Hypervolume Ratio* or HVR [26]. HVR is commonly used for comparing multi-objective optimizers [27], [28]. The other two indicators considered are the Epsilon [28] and the Coverage (C) metric [26]. Both are binary indicators.

B. *Analysis of the Results*

We have performed experiments using M1, M2, M3 and M4 datasets. Regarding to M1, M2 and M3, Figs. 3, 4 and 5 show the aggregated Pareto-front approximations in the three datasets. The behavior in the fourth dataset (M4) is rather similar. We do not present the Pareto-front approximation because of lack of space. The Pareto-front approximation of the decimation approach is not shown because this classical technique obtains very large errors, between 13500 and 19100 for the M1 dataset, around 22000 for the M2 dataset and 15000 for the third dataset; in addition to a high number of triangles, so its graphical representation is far from the other two algorithms. The multi-objective method converges better than the single-objective one. The multi-objective proposal obtains solutions covering the whole search space. The single-objective just finds solutions in the centre (Figs. 3 and 5) or near the right side of the search space (Fig. 4).

Table II shows the mean and standard deviation (in parenthesis) values for the HVR metric of the four methods in the studied datasets. A higher value represents a better behavior. Hence, the multi-objective technique outperforms the rest of the methods.

Table III shows the mean and standard deviation for the Epsilon and C indicators for the M1, M2, M3 and M4 datasets. In the case of the Epsilon indicator, a lower value indicates a better performance. On the contrary, in the C metric, a higher value corresponds to a better performance.

Note that our multi-objective algorithm achieves better results than the single-objective and the classic proposals in both indicators. The difference between algorithms is higher in the M2 dataset. Furthermore, the multi-objective algorithm provides higher diversity as it is shown in Figs. 3-5. The cardinality of the decimation approach is 9 for the M1 dataset, 2 for M2 dataset and 4 in the third and fourth datasets.

Regarding run time, all methods are similar. They converge in around 10 minutes per run.

TABLE II. MEAN AND STANDARD DEVIATION OF THE UNARY HVR INDICATOR, DATASETS M1, M2, M3 AND M4

Datasets	$\bar{x}(\sigma)$		
	Single-objective	***Multi-objective***	***Decimation***
M1	0.633(0)	0.833(0.047)	0(0)
M2	0.467(0)	0.886(0.009)	0(0)
M3	0.169(0)	0.982(0.004)	0(0)
M4	0.328(0)	0.826(0.001)	0(0)

TABLE III. MEAN AND STANDARD DEVIATION OF THE BINARY EPSILON AND C INDICATORS, DATASET M1, M2, M3 AND M4

M1 Dataset				
Methods	**Metrics**	***Single-objective***	***Multi-objective***	***Decimation***
Single-objective	***Epsilon***	•	1.027(0.024)	0.393(0)
	C	•	0.300(0.165)	1(0)
Multi-objective	***Epsilon***	1.010(0.009)	•	0.600(0.001)
	C	0.667(0.150)	•	1(0)
Decimation	***Epsilon***	3.543(0)	3.559(0.014)	•
	C	0(0)	0(0)	•
M2 Dataset				
Methods	**Metrics**	***Single-objective***	***Multi-objective***	***Decimation***
Single-objective	***Epsilon***	•	1.295(0.120)	0.695(0)
	C	•	0(0)	1(0)
Multi-objective	***Epsilon***	0.993(0.400)	•	0.706(0)
	C	0.600(0.516)	•	1(0)
Decimation	***Epsilon***	5.705(0)	6.902(0.011)	•
	C	0(0)	0(0)	•
M3 Dataset				
Methods	**Metrics**	***Single-objective***	***Multi-objective***	***Decimation***
Single-objective	***Epsilon***	•	1.011(0.007)	0.803(0)
	C	•	0(0)	1(0)
Multi-objective	***Epsilon***	1.005(0.004)	•	0.815(0.002)
	C	0.566(0.041)	•	1(0)
Decimation	***Epsilon***	2.056(0)	2.235(0.134)	•
	C	0(0)	(0)	•
M4 Dataset				
Methods	**Metrics**	***Single-objective***	***Multi-objective***	***Decimation***
Single-objective	***Epsilon***	•	1.011(0.007)	0.804(0)
	C	•	0(0)	1(0)
Multi-objective	***Epsilon***	1.004(0.002)	•	0.814(0.003)
	C	0.576(0.009)	•	1(0)
Decimation	***Epsilon***	2.057(0)	2.241(0.012)	•
	C	0(0)	0(0)	•

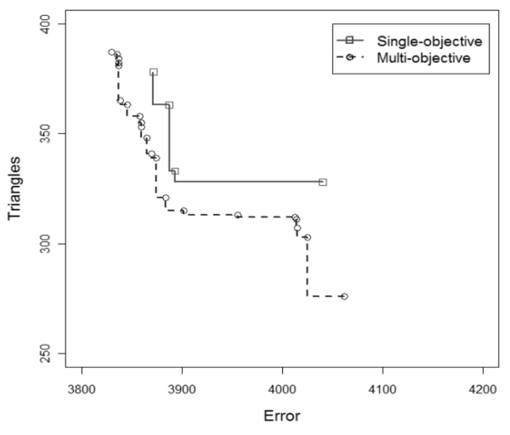

Figure 3. Pareto-front approximations of the single and multi-objective algorithms for the M1 dataset

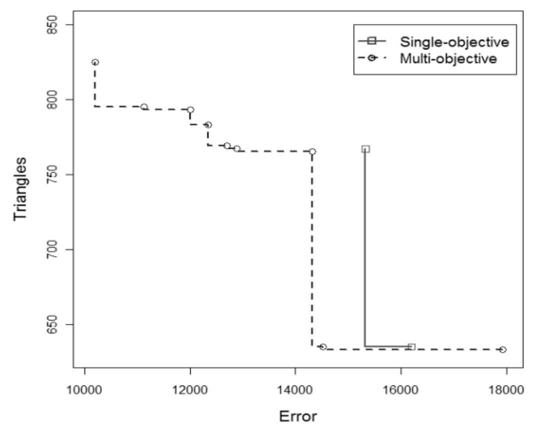

Figure 4. Pareto-front approximations of the single and multi-objective algorithms for the M2 dataset

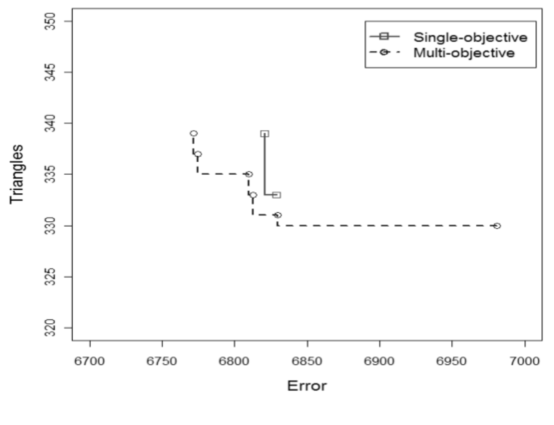

Figure 5. Pareto-front approximations of the single and multi-objective algorithms for the M3 dataset

C. Analysis and Comparison of Solutions

From each Pareto-front approximation we have selected three different solutions, the one having the best value in the first objective, the other with the best value in the second objective, and a compromise solution with the best tradeoff value. The tradeoff solution is selected as follows: we compute 1000 random weights w ∈[0,1], take the average value of the aggregation function of both objectives *O_1* (error) and *O_2* (number of triangles):

$$\bar{F}(s_i) = \frac{\sum_{i=j}^{1000} w_j O\_1(s_i)+(1-w_j)O\_2(s_i)}{1000}. \quad (7)$$

The solution with the lowest aggregated value is selected. For each solution, we present the values of two global learning objectives, error and number of triangles.

Due to the fact that the objectives *O_1* and *O_2* are not normalized we need to apply a factor in order to scale them:

$$\alpha = \frac{\sum_i^{|PA_a|} \frac{O\_2(s_i)}{O\_1(s_i)}}{|PA_a|}, \quad (8)$$

where $|PA_a|$ is the cardinality of the approximation Pareto-front and s_i is a solution of this Pareto-front.

The final aggregation formula to compute the average value is the following:

$$\bar{F}(s_i) = \frac{\sum_{j=1}^{1000} \alpha w_j O\_1(s_i)+(1-w_j)O\_2(s_i)}{1000}. \quad (9)$$

Table IV contains the three types of best solutions of the studied methods for the M1, M2, M3 and M4 models. The multi-objective algorithm achieves better solutions than the rest in all the datasets. The worst performance corresponds to the decimation approach.

Table IV shows that the multi-objective method is better than the single-objective one in all the solutions; in the case of the minimum error (3829.92 against 3871.72), minimum number of triangles (276 versus 328) and best tradeoff, where it gets a low error using a similar number of triangles.

Regarding to the results of the M2 dataset, the multi-objective algorithm outperforms the other two proposals. Compared to the single-objective technique, the multi-objective achieves the minimum error (10198.99 against 15324.94) and the minimum number of triangles (633 versus 635).

The multi-objective method performs better than the other proposals in the M3 and M4 results. Regarding to the M3 dataset, in spite of the fact that both evolutionary algorithms obtain the same number of triangles (331), in the minimum solution, the multi-objective error is lower than the single-objective one (6829.47 against 6980.84). In relation to the M4 model, the multi-objective algorithm achieves the minimum error solution (13592.33 against 13917.14) and the minimum number of triangles (260 versus 322) as compared to the single-objective method. It is worth noting that the differences of error among algorithms are larger for the M2 dataset than for the others. In the case of the number of triangles, evolutionary algorithms obtain similar results being better the multi-objective proposal.

Fig. 6 presents the solutions with minimum number of triangles and minimum error achieved by the evolutionary approaches tackling the M1 dataset.

TABLE IV. BEST SINGLE SOLUTIONS OBTAINED FROM THE PARETO-FRONT APPROXIMATIONS IN M, M2, M3 AND M4 DATASETS

Solutions	M1 Dataset					
	Single-objective		*Multi-objective*		*Decimation*	
	O_1	***O_2***	***O_1***	***O_2***	***O_1***	***O_2***
Min. error	3871.72	378	3829.92	387	13527.7	1583
Min. no. triangles	4040.54	328	4061.76	276	19050.2	833
Best tradeoff	3864.69	348	3894.34	343	13559.8	1499
	M2 Dataset					
Solutions	*Single-objective*		*Multi-objective*		*Decimation*	
	O_1	***O_2***	***O_1***	***O_2***	***O_1***	***O_2***
Min. error	15324.94	767	10198.99	825	22034.4	4376
Min. no. triangles	16198.9	635	17920.45	633	22391.4	2188
Best tradeoff	16198.9	635	12886.9	767	22391.4	2188
	M3 Dataset					
Solutions	*Single-objective*		*Multi-objective*		*Decimation*	
	O_1	***O_2***	***O_1***	***O_2***	***O_1***	***O_2***
Min. error	6820.696	339	6771.51	339	15658.23	1499
Min. no. triangles	6980.848	331	6829.479	331	19897.74	833
Best tradeoff	6828.913	333	6771.51	339	15789.61	1415
	M4 Dataset					
Solutions	*Single-objective*		*Multi-objective*		*Decimation*	
	O_1	***O_2***	***O_1***	***O_2***	***O_1***	***O_2***
Min. error	13917.14	349	13592.33	349	20146.27	1940
Min. no. triangles	14080.93	322	14905.42	260	23635.83	1021
Best tradeoff	14080.93	322	14085.48	296	21645.90	1430

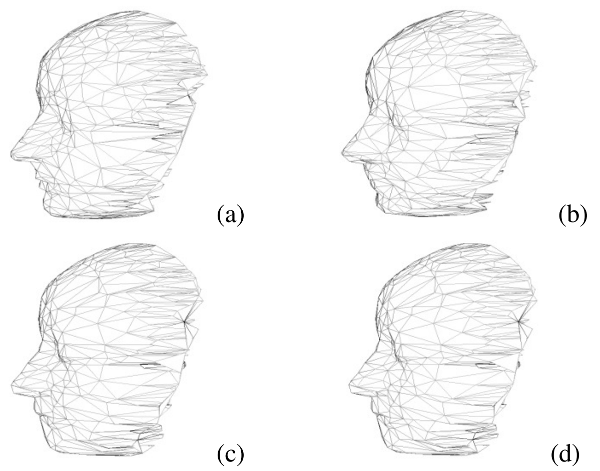

Figure 6. M1 Dataset solutions: (a) Min. error, single-obj. method. (b) Min. number of triangles,single-obj. method. (c) Min. error, multi-obj. method. (d) Min. number of triangles, multi-obj. method.

D. The Wilcoxon Test

The Wilcoxon rank sum test [24] has been used to analyze the significance of the results in the comparison of the quality of the Pareto front approximations obtained by the studied algorithms by means of the previously explained unary and binary indicators. This is done in order to avoid the fact that one exceptionally good result in any of the compared algorithms produces a wrong analysis.

Unlike the commonly used t-test, the Wilcoxon test does not assume normality of the samples and it has already demonstrated to be helpful analyzing the behavior of evolutionary algorithms [10]. Nevertheless, we should remark that there is not any reference methodology to apply a statistical test to a binary indicator in multi-objective optimization. Thus, we have decided to follow the procedure proposed in [3].

Table V shows the results of the statistical test for the dominance probabilities of the algorithms. Every cell of the table includes the averaged dominance probability explained in [3] for the two problem datasets together with a "+", "-", or "=" symbol, with a different meaning. Every symbol shows that the algorithm in that row is significantly better (+), worse (-) or equal (=) in behavior (using the said two binary indicators) than the one that appears in the column.

For example, the pair (0.23, +) must be interpreted as follows: the averaged dominance probability of the multi-objective with respect to the single-objective method (for the Epsilon indicator) is 0.23, and there is statistical significance (+) on the performance of the multi-objective and the single-objective algorithms. In view of this statistical study, we can draw the conclusion that the multi-objective algorithm is significantly better than the rest of the methods.

TABLE V. AVERAGED DOMINANCE PROBABILITY AND STATISTICAL SIGNIFICANCE FOR THE ALGORITHMS

Methods	**Metrics**	***Single-objective***	***Multi-objective***	***Decimation***
Single-objective	***Epsilon***	•	(0.03, -)	(1, +)
	C	•	(0.03, -)	(1, +)
Multi-objective	***Epsilon***	(0.23, +)	•	(0.9, +)
	C	(0.5, +)	•	(1, +)
Decimation	***Epsilon***	(0, -)	(0, -)	•
	C	(0, -)	(0, -)	•

V. CONCLUSIONS

We have proposed a multi-objective evolutionary algorithm to solve the 3D mesh simplification problem. The multi-objective solution consists of applying an NSGA-II algorithm to find the best configurations of points, i.e. the mesh with the minimum error and number of triangles. In order to compare the performance of our proposal, we have considered two different methods: single-objective evolutionary algorithm and a classical technique.

On the one hand, the single-objective evolutionary algorithm is based on the proposal by Huang and Ho [19]. It approximates a 3D model using a mesh generated from a certain number of points located in an adaptive way. This method uses a genetic algorithm to create different Delaunay triangulations and the most accurate is chosen. We have adapted its fitness function to a scalar that combines the total error of the mesh solution with the number of triangles.

On the other hand, we have chosen the classical technique named edge decimation [22] in order to compare it with the evolutionary algorithm behavior. This approach is based on making several iterations over all the edges of the mesh; if an edge satisfies a decimation criterion then it is removed. The algorithm will finish if it reaches a certain termination condition.

We have compared the performance of the previous approaches considering four public available datasets. In addition, the performance comparison includes the Pareto-front approximations of all the methods. Furthermore, a Wilcoxon rank sum test was accomplished in order to analyze the significance of the results. We have considered three performance indicators: HVR, Epsilon and C.

From the developed experiments, the analysis of the performance indicators, the significance of the results and the Pareto-front representations, we can conclude that our proposal clearly outperforms the other two.

Finally, several ideas arise for future works. On the one hand, it would be interesting to extend the method to tackle complete 3D surface approximations. On the other hand, it could be designed a post-processing task to improve the quality of the final model.

ACKNOWLEDGMENT

This work has been supported by the Spanish Ministerio de Educación y Ciencia under the project TIN2009-07727.

REFERENCES

[1] F. Aurenhammer, “Voronoi diagrams. A survey of a fundamental geometric data structure”, ACM Computing Surveys, 23(3), pp. 345-405, 1991.

[2] V. Chankong and Y. Y. Haimes. Multiobjective Decision Making Theory and Methodology, North-Holland, Amsterdam, 1983.

[3] M. Chica, O. Cordón, S. Damas, and J. Bautista, “Multi-objective, constructive heuristics for the 1/3 variant of the time and space assembly line balancing problem: ACO and random greedy search”, Information Sciences, 180, pp. 3465–3487, 2010.

[4] P. Cignoni, C. Montani, and R. Scopigno, “A comparison of mesh simplification algorithms”, Computers & Graphics, 22(1), pp. 37-54, 1998.

[5] C.A. Coello, G. B. Lamont, and D. A. Van Veldhuizen, Evolutionary Algorithms for Solving Multiobjective Problems, Springer, Berlin, 2007.

[6] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright, “Simplification envelopes”, Computer Graphics (SIGGRAPH ’96 Proceedings), ACM Press, pp 119-128, Aug. 6-8 1996.

[7] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, “A fast and elitist multiobjective genetic algorithm: NSGA-II”, IEEE Transactions on Evolutionary Computation, 6(2), pp. 182-194, April 2002.

[8] A.E Eiben and J.E Smith. Introduction to Evolutionary Computing. Springer Berlin Heidelberg, New York, 2003.

[9] Y. Fujiwara and H. Sawai, “Evolutionary computation applied to mesh optimization of a 3-D facial image”, IEEE Transactions Evolutionary Computation, 3(2), pp. 113-123.

[10] S. García, D. Molina, M. Lozano, and F. Herrera, “A study on the use of nonparametric tests for analyzing the evolutionary algorithms’ behaviour: a case of study”, CEC’2005 Special Session on Real Parameter Optimization, Journal of Heuristics, 15, pp. 617–644, 2009.

[11] M. Garland and P.S. Heckbert, “Simplifying surfaces with color and texture using quadric error metrics”, Computer Graphics (SIGGRAPH ’97 Proceedings), pp. 209–216, 1997.

[12] M.H. Gross, O.G. Staadt, and R. Gatti, “Efficient triangular surface approximations using wavelets and quadtree data structures”, IEEE Transactions on Visualization and Computer Graphics, 2(2), pp. 130-144, June 1996.

[13] M.J. Haemer and M.J. Zyda, “Simplification of objects rendered by polygonal approximation”, Computers and Graphics, 15(2), pp. 175–184, 1991.

[14] B. Hamann, “A data reduction scheme for triangulated surfaces, Computer Aided Geometric Design”, 11(20), pp. 197–214, 2004.

[15] D.J. Hebert and H.-J. Kim, “Image encoding with triangulation wavelets”, Proceedings SPIE, 2569(1), pp. 381-392, 1995.

[16] P. Hinker and C. Hansen, “Geometric optimization”, Gregory M. Nielson and Dan Bergeron, editors, Proceedings Visualization ’93, pp. 189–195, October 1993.

[17] G. Hoppe, “Progressive meshes”, SIGGRAPH’96 Conference Proceedings, pp. 99–108, 1996.

[18] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, “Mesh optimization”, ACM Computer Graphics Proceedings, pp. 19- 26, 1993.

[19] H.L. Huang, S. Y. Ho, “Mesh optimization for surface approximation using an efficient coarse-to-fine evolutionary algorithm”, Pattern Recognition, 36, pp. 1065-1081, 2003.

[20] T. Mori. Taguchi techniques for image and pattern developing technology, Prentice-Hall, New Jersey, 1995.

[21] G. Pengdong, L. Ameng, L. Yongquan, H. Jintao, L. Nan, and Y. Wenhua, “Adaptative mesh simplification using vertex clustering with topology preserving”, CSSE '08 Proceedings of the 2008 International Conference on Computer Science and Software Engineering, 2, pp. 971-974, December 2008.

[22] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen, “Decimation of triangular meshes”, Computer Graphics (SIGGRAPH ’92 Proceedings), pp. 65–70, 1992.

[23] G. Turk, “Re-tiling polygonal surfaces”, Edwin E. Catmull, editor, Computer Graphics (SIGGRAPH ’92 Proceedings), 26, pp. 55–64, July 1992.

[24] F. Wilcoxon, “Individual comparisons by ranking methods”, Biometrics Bulletin, 1(6), pp. 80-83, 1945.

[25] T. Xiaodong, W. Yuexian, Z. Xionghui, and R. Xueyu, “Mesh simplification based on super-face and genetic algorithm”, Reverse Engineering. Advanced Manufacturing Technology, 29, pp. 303-312, 2002.

[26] E. Zitzler and L. Thiele, “Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach”, IEEE Transactions on Evolutionary Computation, 3(4), pp. 257–271, 1999.

[27] E. Zitzler, L. Thiele, and K. Deb, “Comparison of multiobjective evolutionary algorithms: Empirical results, evolutionary computation”, 8(2), pp. 173–195, 2000.

[28] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V.G. da Fonseca, “Performance assessment of multiobjective optimizers: An analysis and review”, IEEE Transactions on Evolutionary Computation, 7(2), pp. 117–132, 2003.

[29] http://shapes.aim-at-shape.net/viewmodels.php