International Journal of Computer Science and Applications

Vol. 11 No. 2, pp. 1 - 18, 2014

Bézier Curve Parametrization Using a Multiobjective Evolutionary Algorithm

ALLAN YOSHIO HASEGAWA
Department of Electrical Engineering, Santa Catarina State University
Joinville, SC 89219-710, Brazil
hasegawa.aran@gmail.com

CAMILA TORMENA
Department of Computer Science, Santa Catarina State University
Joinville, SC 89219-710, Brazil
trmcami@gmail.com

RAFAEL STUBS PARPINELLI
Graduate Program in Applied Computing
Department of Computer Science, Santa Catarina State University
Joinville, SC 89219-710, Brazil
rafael.parpinelli@udesc.br

Curve fitting is a classical problem in several areas, and the choice of parameters can drastically change the results obtained. Optimizing the parameters is a complex task, and there are many works in the literature applying nature inspired algorithms to solve this problem. A common observation in these works is that they minimize only the error of the curve and this can result in anomalies on its shape. This work proposes a parameter optimization approach that uses a multiobjective evolutionary algorithm. We used the GDE3 to minimize two objective functions: the error and length of the curve. Results showed that our method is able to obtain aesthetically pleasing curves and maintains a good trade-off between error and length. Also, it can perform similarly to a single objective function algorithm if needed.

Keywords: Curve fitting; Evolutionary computation; Multiobjective optimization; Bézier curves; GDE3; Parametrization; Optimization.

1. Introduction

Obtaining a curve or surface that approximates a given cloud of data points is a classical problem in several scientific and technological domains such as computer-aided design (CAD) [Sevaux and Mineur (2007)], computer-aided manufacturing (CAM) [Pottmann (2005)], data compression [Khan (2012)], virtual reality [Leu *et al.* (2005)] and data visualization [Prasad and Fitzgibbon (2006)]. The data points can come from a physical object that is converted into a fully usable digital model, a

process called reverse engineering. The digital models are usually easier and cheaper to modify, leading to a reduction of the processing and manufacturing costs, and also become available anytime and anywhere [Gálvez and Iglesias (2013)].

Due to problems like measurement noise and irregular sampling, a good fitting of data is generally based on approximation (where the curve is expected to pass near the data points) rather than on interpolation (where the curve is constrained to pass through all the input data points). There are two key components for a good approximation of data points: a proper choice of the approximating function and a suitable parameter tuning. The usual models for curve approximation are the free-form curves, such as Bézier, B-spline, and NURBS. In this paper we focus on polynomial Bézier curves [Gálvez and Iglesias (2013)]. Although nowadays Bézier splines have been overtaken by the B-splines (a generalization of the Bézier splines), they are still widely used for different purposes, such as computer fonts (e.g., True-Type Fonts, PostScript), computer animation (for simple movements of objects in programs such as Adobe Flash), and computer design (Adobe Photoshop, Corel Draw, Adobe illustrator) [Gálvez and Iglesias (2013)].

Curve/surface fitting methods are mainly based on the least-squares approximation (LSQ) scheme, a classical optimization technique that, given a series of measured data, attempts to find a function which closely approximates the data (a best fit) [Gálvez *et al.* (2007)]. The procedure starts with an estimation of parameters, followed by applying the LSQ to find the shape of the curve.

Considering the curve fitting problem using a Bézier curve detailed in Section 2, the LSQ determines the curve control points from the information given by the data points. Because the curve is parametric, the parameter values corresponding to the data points also appear as unknowns in the formulation. This turns the error minimization problem into a nonlinear problem, having a high number of unknowns for large sets of data points [Gálvez *et al.* (2007)]. Still, by defining the curve parameters before solving the LSQ, and assuming a number of data points larger than the degree of the curve, the problem can be solved as an overdetermined linear system [Farin (2002)].

The choice of parameters for the curve can drastically change the results of the curve fitting problem. This problem has been extensively studied and the three primary parametrization methods reported in the literature are: equally spaced (or uniform), chord length and centripetal. The uniform method creates the parameter vector with equally spaced values. This method is not recommended, as it can produce erratic shapes when the data is unevenly spaced. Chord length uses the distance between data points to approximate the curve parameters, while the centripetal method can be compared to the chord length but provides better results when the data takes very sharp turns [Piegl and Tiller (1997)]. Lee [Lee (1989)] proposed a generalization of the uniform, chord length and centripetal methods, through the use of a single equation, as presented in Section 2.1.

Optimizing the parameters in order to perform least square fitting is a com-

plex problem, and nature inspired algorithms are a viable option. In this way, the Computational Intelligence field provides these biologically inspired techniques to address complex problems to which traditional approaches are ineffective or infeasible. These include Swarm Intelligence and Evolutionary Algorithms [Engelbrecht (2007)] [Parpinelli *et al.* (2011)]. These techniques can be used to optimize curve parameters and works in the area have shown that they can achieve remarkable results regarding this problem [Gálvez *et al.* (2007); Gálvez and Iglesias (2013)].

Kumar *et al.* [Kumar *et al.* (2003)] proposed the use of genetic algorithms (GA) to optimize the parameters. They showed a novel population initialization where half the population is generated using randomly chosen parametrization methods through the equation proposed by [Lee (1989)], while the other half is created through an approach based on radial basis function networks (RBFN). This population initialization method allowed for a faster rate of convergence to an acceptable answer. Gálvez and Iglesias [Gálvez and Iglesias (2013)] proposed the use of the Firefly algorithm to optimize the parameters of Bézier surfaces. The firefly algorithm is based on the social flashing behavior of fireflies in nature, considering the variation of light intensity and formulation of attractiveness. In general, the attractiveness of an individual is assumed to be proportional to their brightness and associated with the encoded objective function.

Solving the LSQ will result in an *optimal* solution aimed only at minimizing the error function. Such method, as applied by [Kumar *et al.* (2003)] and [Gálvez and Iglesias (2013)], ignore aesthetic aspects of the curve and may result in a curve with more wiggles than desirable. In applications where the shape of the curve is important, as in computer graphics and computer animation, it's important to take into account other information in addition to the approximation error. Farin [Farin (2002)] presented a way of reducing the undesirable wiggles by analysing the polygon created from the control points of the curve. The method works by extending the LSQ to also minimize the difference between control points, thus creating a smoother curve. The balance between the curve error and wiggles in this extension can be weighted by a variable $\alpha \in [0, 1)$ — the bigger the value, the smoother the curve will be.

This work extends the works from [Kumar *et al.* (2003)] and [Gálvez and Iglesias (2013)] by using a multiobjective evolutionary algorithm, which minimizes both the error and the length of the curve, resulting in a more visually pleasant shape for the curve. While Farin [Farin (2002)] extends the LSQ in order to obtain a smoother curve, our work is focused on the optimization of the curve parameters, even before the LSQ phase, aiming for an aesthetically pleasing curve with low error.

This paper is structured as follows: the problem is detailed in Section 2 and multiobjective evolutionary algorithms are presented in Section 3. Our method is described in Section 4 and the results are shown in Section 5. Section 6 presents the conclusions.

2. Bézier Curve

A Bézier curve is a parametric curve composed by a set of control points $(\mathbf{p}_0, \mathbf{p}_1, \ldots, \mathbf{p}_n)$, where n is the curve's order.

A general $(n+1)$-degree Bézier curve is defined by

$$\mathbf{c}(u) = \sum_{i=0}^{n} \mathbf{p}_i B_{i,n}(u) \tag{1}$$

where $u \in [0,1]$ is the curve parameter [Mortenson (1997)]. The basis functions $B_{i,n}(u)$ are defined by

$$B_{i,n}(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

where the binomial coefficient function is equal to

$$\binom{n}{i} = \frac{n!}{i!(n-i)!}$$

Let $\mathbf{u} = (u_0, u_1, \ldots, u_w)$, where $\mathbf{u}$ is a vector containing $w+1$ equidistant parameters in the parametric axis. The arc-length of a Bézier curve can be approximated by

$$S = \sum_{i=0}^{w-1} \|\mathbf{c}(\mathbf{u}_{i+1}) - \mathbf{c}(\mathbf{u}_i)\| \tag{2}$$

As w approaches infinity, the distance between the consecutive parameters in $\mathbf{u}$ approaches zero, thus S converges to the arc length of the curve [Farin (2002)].

2.1. *Curve Fitting*

Bézier Curve Fitting is the problem of finding the $\mathbf{c}(u)$ that approximates a sequence of $m+1$ data points $(\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_m)$ using the control points as variables. LSQ finds the set of control points $\mathbf{p}_i$ that best fits the data points $\mathbf{d}_j$. Assuming $m > n$, $\mathbf{u} = (u_0, u_1, \ldots, u_m)$ with $u_0 = 0$, $u_m = 1$ and $\mathbf{u}$ being a strictly increasing ordered vector, the LSQ will seek a curve that minimizes the function

$$f = \sum_{k=1}^{m-1} |\mathbf{d}_k - \mathbf{c}(u_k)|^2 \tag{3}$$

with respect to the $n+1$ variables $\mathbf{p}_i$ [Piegl and Tiller (1997)]. The first and last data points are ignored because the Bézier curve formulation guarantees interpolation on those points.

Expanding $\mathbf{c}(u_k)$ from Eq. (3) with Eq. (1) shows the unknowns of the problem: the set of control points $(\mathbf{p}_i)$ and the curves parameters $(\mathbf{u})$. By defining the vector $\mathbf{u}$ before solving Eq. (3), and assuming a number of data points larger than the

degree of the curve, the problem can be solved as an overdetermined linear system [Farin (2002)].

The three primary parametrization methods reported in the literature are: equally spaced (or uniform), chord length and centripetal. Lee [Lee (1989)] proposed a generalization of these methods as

$$\begin{gathered} u_0 = 0, \\ u_k = u_{k-1} + \frac{\|\mathbf{d}_i - \mathbf{d}_{i-1}\|^e}{\sum_{j=1}^{m} \|\mathbf{d}_j - \mathbf{d}_{j+1}\|^e}, 1 \leq k \leq m \end{gathered} \tag{4}$$

where $m+1$ is the number of data points and $\mathbf{d}_i$ is the ith data point. The e variable defines which method is being used — for $e = 0$ the equation reduces to the uniform method, for $e = 0.5$ to the centripetal method and for $e = 1$ to the chord length method.

3. MOEA

Many real-world optimization problems involve multiple objectives which often conflict with each other — the improvement of one may lead to deterioration of another. In these cases a single solution that optimizes all objectives simultaneously does not exist. By evolving a population of solutions, multiobjective evolutionary algorithms (MOEAs) are able to approximate the Pareto optimal set, which contains the best trade-off solutions that are later evaluated by a decision maker (DM) [Zhou *et al.* (2011)].

One multiobjective evolutionary algorithm is the Generalized Differential Evolution (GDE), which extends the Differential Evolution (DE), a floating-point encoded Evolutionary Algorithm (EA) introduced by Storn and Price in 1995 [Storn and Price (1997)], for constrained multiobjective optimization. Its third version, the GDE3, improves population diversity and stability for the selection of control parameter values when compared to earlier versions, as it implements a growing population and a non-dominated sorting with pruning to decrease population size [Kukkonen and Lampinen (2005)].

Another multiobjective evolutionary algorithm is the nondominated sorting genetic algorithm (NSGA) proposed in [Srinivas and Deb (1995)] and improved in [Deb *et al.* (2002)] as the NSGA-II. From the simulation results done in [Deb *et al.* (2002)] on test problems, the NSGA-II outperformed two other MOEAs, Pareto-archived evolution strategy (PAES) and strength-Pareto EA (SPEA), in terms of finding a diverse set of solutions and in converging near the true Pareto-optimal set.

In [Kukkonen and Lampinen (2005)], GDE3 was tested with a set of different types of test problems, and when compared to the NSGA-II method, it demonstrated being able to obtain a more diverse solution and to reduce the number of function evaluations needed. Depending on the test problem, GDE3 was also able to find a better converged solution. Therefore, the GDE3 was chosen for this work.

3.1. *GDE3*

GDE3 works in a similar way to the classic Differential Evolution algorithm. They share the initialization, mutation and crossover steps, but differ in the selection and pruning steps. In the case of a problem with a single objective and without constraints, GDE3 falls back to the original DE [Kukkonen and Lampinen (2005)].

The initialization step creates the initial population. Usually, this population is created randomly and then improved using mutation, crossover and selection on every solution at each generation G.

DE mutates the population to create N trial vectors, where N is the population size. Eq. (5) shows the creation of a trial vector $\mathbf{v}_{i,g}$, where i is its index and g its generation. The $\mathbf{x}_{ry,g}$ vectors, with $y \in (1, 2, 3)$, are distinct solutions chosen at random. The $F \in (0, 1+)$ is the scale factor and controls the rate at which the population evolves [Storn and Price (1997)].

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F \cdot (\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}) \tag{5}$$

The crossover combines each vector with a mutant vector, as in Eq. (6) where j defines the vector dimensions. The condition "$j = j_{\text{rand}}$" guarantees that at least one dimension changes from the old generation [Storn and Price (1997)].

$$\mathbf{u}_{i,g} = u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } (\text{rand}_j(0,1) \leq CR \text{ or } j = j_{\text{rand}}) \\ x_{j,i,g} & \text{otherwise.} \end{cases} \tag{6}$$

This method is what characterizes the "classic DE", or more technically "DE/rand/1/bin", but other variations can be used [Price *et al.* (2005)].

The selection step is where GDE3 differs from DE. In the traditional DE the trial vector $\mathbf{v}_{i,g}$ is compared against $\mathbf{x}_{i,g}$ using a single objective function to decide which one will be passed to the next generation. GDE3 uses a dominance criteria between the two vectors to support multiple objective functions.

Considering a multiobjective minimization problem, a vector $\mathbf{x}_1$ weakly dominates $\mathbf{x}_2$, *i.e.*, $\mathbf{x}_1 \preceq \mathbf{x}_2$ iff $\forall i : f_i(\mathbf{x}_1) \leq f_i(\mathbf{x}_2)$, where f_i is the ith objective function. A vector $\mathbf{x}_1$ dominates $\mathbf{x}_2$, *i.e.*, $\mathbf{x}_1 \prec \mathbf{x}_2$ iff $\mathbf{x}_1$ and $\mathbf{x}_2 \wedge \exists i : f_i(\mathbf{x}_1) < f_i(\mathbf{x}_2)$. A vector $\mathbf{x}_1$ constraint-dominates $\mathbf{x}_2$, *i.e.*, $\mathbf{x}_1 \prec_c \mathbf{x}_2$ iff any of the following conditions is true [Kukkonen and Lampinen (2005)]:

- $\mathbf{x}_1$ is feasible and $\mathbf{x}_2$ is not.
- $\mathbf{x}_1$ and $\mathbf{x}_2$ are infeasible and $\mathbf{x}_1$ dominates $\mathbf{x}_2$ in constraint function space.
- $\mathbf{x}_1$ and $\mathbf{x}_2$ are feasible and $\mathbf{x}_1$ dominates $\mathbf{x}_2$ in objective function space.

Using the above dominance relationships, the GDE3 selection step is then based on the following rules [Kukkonen and Lampinen (2005)]:

- In the case of infeasible vectors, the trial vector is selected if it weakly dominates the old vector in constraint violation space, otherwise the old vector is selected.

- In the case of the feasible and infeasible vectors, the feasible vector is selected.
- If both vectors are feasible, then the trial is selected if it weakly dominates the old vector in the objective function space. If the old vector dominates the trial vector, then the old vector is selected. If neither vector dominates each other in the objective function space, then both vectors are selected for the next generation.

After a generation, the size of the population may have increased, in which case the pruning step should reduce the population back to its original size. GDE3 implements an improved version of the original pruning algorithm from NSGA-II [Kukkonen and Lampinen (2005); Kukkonen and Deb (2006)]. The population is sorted by non-dominated sets and crowdedness, approximated by the Crowding Distance [Deb *et al.* (2002)], then the worst solutions are discarded. Non-dominated sorting for problems with two objectives can be done with complexity $O(N \log N)$, where N is the population size. For more than three objective functions the complexity is $O(N \log^{M-1} N)$, with M being the number of objective functions. Overall running time for GDE3 is $O(G_{\max} N \log^{M-1} N)$ [Kukkonen and Lampinen (2005)].

As in other multiobjective optimization approaches, the GDE3 approximates a set of Pareto optimal solutions. This set contains all non-dominated solutions from which a posteriori Decision Maker (DM) can choose the most appropriate ones [Zhou *et al.* (2011)].

Figure 1 presents a pseudo-code for GDE3.

4. The Proposed Method

In this work, a MOEA is used to solve the problem of choosing the parameters for the LSQ method for Bézier Curve fitting. Given a set of data points $(\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_m)$, the proposed method returns the set of parameters $(u_0, u_1, \ldots, u_m)$ that best fits the data points taking into account the error, as in Eq. (3), and the length, as in Eq. (2).

The GDE3 framework is used to approximate the Pareto optimum solutions. The selection and pruning steps do not deviate from the pseudo-code presented in Figure 1. The following subsections describe the initialization, mutation, crossover and decision maker.

4.1. *Initialization*

Two populations of size $N/2$ are created. Each member of a population is a vector $\mathbf{u}_k = (u_{0,k}, u_{1,k}, \ldots, u_{m-1,k})$, where $m + 1$ is the number of data points to be approximated. Each $\mathbf{u}_k$ is a solution containing the parameters for the Bézier Curve fitting problem. There is no need to map the parameters for the first and last data points, as they will always be 0 and 1, respectively. Each solution has the following constraints:

- $0 \leq u_i \leq 1$
- $u_0 \leq u_1 \leq \cdots \leq u_{m-1}$

Require: D, $G_{\max}$, N ≥ 4, F $\in$ (0,1+], CR $\in$ [0,1]
  Initialize population randomly.
  while $G < G_{\max}$ **do**
    for $\forall i \leq N$ **do** ▷ Mutation and crossover
      $r_1, r_2, r_3 \leftarrow \text{rand(1,N)} : r_1 \neq r_2 \neq r_3 \neq i$
      $j_{\text{rand}} \leftarrow \text{rand(1,D)}$
      for $\forall j \leq D$ **do**
        if $\text{rand}_j(0,1) \leq CR$ or $j = j_{\text{rand}}$ **then**
          $u_{j,i,G} \leftarrow x_{j,r1,G} + F \cdot (x_{j,r2,G} - x_{j,r3,G})$
        else
          $u_{j,i,G} \leftarrow x_{j,i,G}$
        end if
      end for
      if $\mathbf{u}_{i,G} \preceq_c \mathbf{x}_{i,G}$ **then** ▷ Selection
        $\mathbf{x}_{i,G+1} \leftarrow \mathbf{u}_{i,G}$
      end if
      if $\begin{pmatrix} \forall j : u_{j,i,G} \text{ is inside constraints} \\ \wedge \\ \mathbf{x}_{i,G+1} = \mathbf{x}_{i,G} \\ \wedge \\ \mathbf{x}_{i,G} \not\prec \mathbf{u}_{i,G} \end{pmatrix}$ **then**
        $m \leftarrow m + 1$
        $\mathbf{x}_{N+m,G+1} \leftarrow \mathbf{u}_{i,G}$ ▷ Population Growth
      end if
    end for
    while $m > 0$ **do** ▷ Pruning
      Select $\mathbf{x} \in (\mathbf{x}_{1,G+1}, \mathbf{x}_{2,G+1}, \ldots, \mathbf{x}_{N+m,G+1})$:
      $\begin{cases} \forall i \ \mathbf{x} \not\prec_c \mathbf{x}_{i,G+1} \\ \wedge \\ \forall (\mathbf{x}_{i,G+1} : \mathbf{x}_{i,G+1} \not\prec_c \mathbf{x}) \ \text{CD}(\mathbf{x}) \leq \text{CD}(\mathbf{x}_{i,G+1}) \end{cases}$
      Remove $\mathbf{x}$
      $m \leftarrow m - 1$
    end while
    $G \leftarrow G + 1$
  end while

Fig. 1. Pseudo-code for GDE3 [Kukkonen and Lampinen (2005)] . D is the number of dimensions. $G_{\max}$ is the number of generations to be processed. N is the population size. F and CR are DE constants. The CD function computes the Crowding Distance.

The solutions for the first population are created by choosing random values for the e variable in Eq. (4), as in [Kumar *et al.* (2003)]. The second population is generated using random values for each solution. The first population increases the convergence rate, while the second one increases the diversity. The two populations are then merged into a single one, resulting in a population of size N.

4.2. *Mutation and Crossover*

The mutation and crossover proposed are similar to the "DE/rand/1/bin" method, as discussed in Section 3.1. The only difference is in the parameter constraints, as presented in Figure 2. When a value from the trial vector does not satisfy the parameter constraints, the algorithm will approximate a valid solution. This algorithm is based on the deterministic variation of the Bounce-Back technique [Price *et al.* (2005)] for boundary constraints.

Require: i = index of population member
  $r_1, r_2, r_3 \leftarrow \text{rand(1,N)} : r_1 \neq r_2 \neq r_3 \neq i$
  $j_{\text{rand}} \leftarrow \text{rand(1,D)}$
  for $\forall j \leq D$ **do**
    if $\text{rand}_j(0,1) \leq CR$ or $j = j_{\text{rand}}$ **then**
      $u_{j,i,G} \leftarrow x_{j,r1,G} + F \cdot (x_{j,r2,G} - x_{j,r3,G})$
      while $u_{j,i,G} < 0$ **do**
        $u_{j,i,G} \leftarrow x_{j,i,G} - (u_{j,i,G} + x_{j,i,G})/2$
      end while
      while $u_{j,i,G} > 1$ **do**
        $u_{j,i,G} \leftarrow x_{j,i,G} + (u_{j,i,G} - x_{j,i,G})/2$
      end while
    else
      $u_{j,i,G} \leftarrow x_{j,i,G}$
    end if
  end for
  sort $\mathbf{u}_{i,G}$

Fig. 2. Pseudo-code for mutation and crossover of the proposed method.

4.3. *Posteriori Decision Maker*

After computing $G_{\max}$ generations, GDE3 will have a set of non-dominated Pareto points in the Pareto Front (PF). To find a single solution, a posteriori Decision Maker, presented in Figure 3, was implemented. The goal of this step is to find a curve that balances error and length. The variable β controls the weight of the error function, and refers to the percentage of error allowed in relation to the smallest

error found. For example, a $\beta = 0.2$ will allow the algorithm to choose a curve with smaller length, as long as the error of this curve is not higher than 20% of the smallest error found. The β value can be set a priori, specific to a certain application, or manipulated by the user until a suitable curve is found. The cost for changing β is cheap since there is no heavy computation involved.

Require: β, PF_0
  $f_1 \leftarrow$error objective function
  $f_2 \leftarrow$length objective function
  $l_{\min} \leftarrow$ solution with the lowest objective value in PF_0 using f_1
  sort PF_0 using f_2
  for $\forall \mathbf{t}_k \in PF_0$ **do** ▷ For loop in increasing order
    if $f_1(\mathbf{t}_k) \leq l_{\min} + l_{\min} * \beta$ **then**
      return $\mathbf{t}_k$
    end if
  end for

Fig. 3. Pseudo-code for the proposed decision maker. β controls the weight of the error function (f_1). PF_0 is the set of non-dominated solutions in the population.

4.4. *Problem Constants*

The GDE3 requires some constants to be defined before beginning its execution. Some of these values can drastically affect the output of the algorithm. For this work, most of them were chosen following guidelines from other works in the area, and the remaining ones were chosen empirically.

$G_{\mathbf{max}}$
: The $G_{\max}$ controls the number of generations to be processed. Since the convergence to a good result happens before the 100th generation, as shown in Figure 11, a $G_{\max}$ of 100 was used. $G_{\max}$ is used as stop criteria in this work, but there are other viable options, for example stopping when the curve's error is below a certain threshold.

CR **and** F
: CR and F are two "classic DE" constants. In this work, $CR = 0.1$ and $F = 0.2$ were used, based on suggestions from [Price *et al.* (2005)].

N
: This constant defines the size of the population. A population of size 100 was used in this work, allowing for a good diversity. Based on empirical tests conducted, higher values did not result in meaningful changes to the algorithm's output, but increased the computational cost considerably.

5. Experimental Results

In this Section, six test results are presented. The examples were chosen to demonstrate the adaptability of the proposed method.

Each test compares our method (multiobjective optimization using GDE3) with three other methods: Firefly Algorithm, Differential Evolution and Differential Evolution implementing Farin's smooth extension to the LSQ problem [Farin (2002)], described in Section 3. Each test case was executed 100 times for each algorithm, and each execution consisted of 10 000 function evaluations (a population consisting of 100 members executing 100 generations).

For the DE with smooth algorithm, the parameter α controls the weight of the smoothness: $\alpha = 0$ ignore the smoothness factor while values closer to 1 give more weight to the fitting equation. Thus, the value used will depend on the test case being considered [Farin (2002)]. In the same way, the values of the β parameter in our algorithm also depend on each case and were chosen empirically. While changing α requires recomputing all the generations, β only chooses a member of a non-dominated population. This allows a fine tuning of the results, since no heavy computation is required.

The curve for the first example was constructed with two straight line segments (in the shape of a '7') followed by a quadratic curve segment. Another particularity is that the number of data points in this example is very low. This test fits a single 6th-Order Bézier Curve segment and aims to demonstrate how the proposed method performs with unevenly spaced data points.

The curve for the second example is defined by the parametric equation[a]

$$x = 16\sin^3(t)$$
$$y = 13\cos(t) - 5\cos(2t) - 2\cos(3t) - \cos(4t)$$

with $t \in (0, 2\pi)$. A 12th-Order Bézier Curve was used in this example and it shows how the proposed method deals with symmetrical closed-form curves.

The curve for the third example is defined by

$$y = x * sin(x) + rand(-0.5, 0.5)$$

with the stochastic part simulating noise in the signal. A 18th-Order Bézier Curve was used in this example and it demonstrates how the proposed method deals with a set of tightly spaced data points.

The curve for the fourth example is an Epicycloid curve[b] defined by the parametric equation

$$x = (a+b)\cos(t) - b\cos(\frac{a+b}{b}t)$$
$$y = (a+b)\sin(t) - b\sin(\frac{a+b}{b}t)$$

[a]Heart Curve. `http://mathworld.wolfram.com/HeartCurve.html`.
[b]Epicycloid. `http://mathworld.wolfram.com/Epicycloid.html`.

where b is the radius of the circle rotating around a circle with radius a, and $t \in (0, 2\pi)$. A 21th-Order Bézier Curve was used.

The curves for the fifth and sixth examples are 3D curves. The fifth example is a Conical Spiral curve[c], defined by

$$\begin{aligned} x &= t\ r\ \cos(a\ t) \\ y &= t\ r\ \sin(a\ t) \\ z &= t \end{aligned} \tag{7}$$

with $t \in (0, 1)$. The sixth example is a Viviani curve[d], defined by

$$\begin{aligned} x &= 1 + \cos(t) \\ y &= \sin(t) \\ z &= 2\sin(\frac{1}{2}t) \end{aligned}$$

with $t \in (-2\pi, 2\pi)$. In both cases a 12th-Order Bézier Curve was used.

Table 1 presents the values obtained for error and length, including the average, standard deviation and lowest value, computed from a hundred executions of each test case. The method proposed usually does not obtain the smallest error when compared to the other algorithms, but instead balances it with the length of the curve, as it was designed to. In the following paragraphs, the results obtained are analysed.

The first and second examples (Figures 4(a) and 4(b)) show that the DE method considers only the error of the curve and ignores its shape, as especially visible by the anomaly at the top of the image in Figure 4(b). Even thought the DE can find the curve with the least error among all the tested methods, the curve's shape obtained may be undesirable. The DE with smooth variation, using $\alpha = 0.01$ in both the first and second examples, produced a rounder curve but there's no direct control over how much error it allows. The Firefly algorithm seems to perform similarly to the proposed method, but it is not actually considering any information besides the fitting error, and brings up problems similar to the DE algorithm, as visible by the anomaly inside the shape in the second example. The proposed method using GDE3, with $\beta = 0.1$ in the first example and $\beta = 2$ in the second example, managed to find a good balance between error and length.

In the third example (Figure 5(a)), the proposed method using GDE3, with $\beta = 0$, performed similarly to the DE algorithm. When lots of data points are available, minimizing just the error is a good option, and this example shows the flexibility of the proposed method in falling back to DE when β equals zero. The Firefly algorithm also approximated the curve well, but once again presented an anomaly (that can be seen at the top left corner of the image). The DE with smooth, even with a small $\alpha = 0.001$, failed to approximate the original curve properly. The

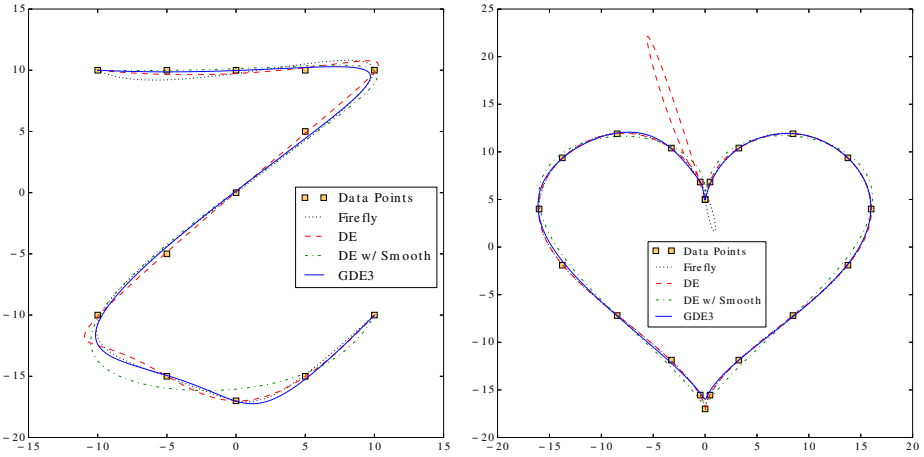
[c]Conical Spiral. `http://mathworld.wolfram.com/ConicalSpiral.html`.
[d]Viviani's Curve. `http://mathworld.wolfram.com/VivianisCurve.html`.

Table 1. Comparison of error and length obtained in the curves generated by the tested methods.

Example	Metric		Firefly	DE	DE w/ Smooth	GDE3
1st	Error	Avg.	4.75	3.56	4.99	5.86
		SD	0.41	0.94	0.52	1.57
		Lowest	3.16	1.61	4.13	3.04
	Length	Avg.	79.10	75.32	74.99	60.64
		SD	6.99	8.02	2.99	2.05
		Lowest	73.67	70.75	69.74	50.66
2nd	Error	Avg.	1.84	1.47	9.86	2.46
		SD	0.33	0.66	1.16	0.95
		Lowest	1.00	0.66	6.01	1.15
	Length	Avg.	111.45	114.98	101.27	97.61
		SD	13.25	6.07	1.02	0.55
		Lowest	101.95	104.43	99.46	95.92
3rd	Error	Avg.	24.30	12.73	37.84	16.72
		SD	1.65	2.35	2.74	2.02
		Lowest	20.38	7.99	30.71	12.33
	Length	Avg.	136.01	137.24	131.28	112.30
		SD	2.60	2.33	4.74	2.86
		Lowest	130.76	134.24	124.12	104.68
4th	Error	Avg	10.07	5.65	29.41	6.09
		SD	0.72	0.24	2.21	0.21
		Lowest	8.71	5.06	23.53	5.44
	Length	Avg	48.51	50.25	43.72	45.31
		SD	0.89	0.63	1.25	0.22
		Lowest	47.51	49.11	42.58	43.65
5th	Error	Avg.	0.37	0.55	1.09	0.85
		SD	0.04	0.31	0.23	0.40
		Lowest	0.26	0.19	0.61	0.38
	Length	Avg.	10.36	10.24	10.18	8.71
		SD	0.50	0.13	0.12	0.75
		Lowest	10.16	10.01	9.92	6.69
6th	Error	Avg	0.48	0.51	0.61	0.60
		SD	0.05	0.28	0.32	0.44
		Lowest	0.36	0.12	0.16	0.11
	Length	Avg	15.30	15.37	15.22	14.96
		SD	0.14	0.18	0.05	0.14
		Lowest	15.26	15.15	15.13	14.47

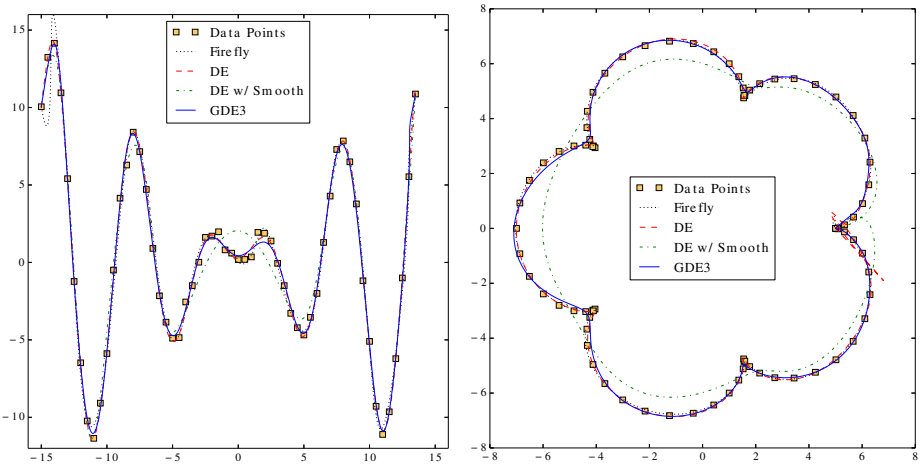
reason why the DE with smooth is not a good alternative for this curve is because α tries to minimize the difference between adjacent control points, but that difference is what characterizes this curve's shape.

In the fourth example (Figure 5(b)), the proposed method using GDE3, with $\beta = 5$, performed similarly to the Firefly algorithm. The DE algorithm presented an anomaly while the DE with smooth, even with $\alpha = 0.001$, failed to approximate the original curve properly. This example just emphasizes the problems presented by the previous test cases.



(a) 1st example (b) 2nd example

Fig. 4. Comparison between the four tested methods with a a) small set of unevenly spaced data points, and b) closed form curve.



(a) 3rd example (b) 4th example

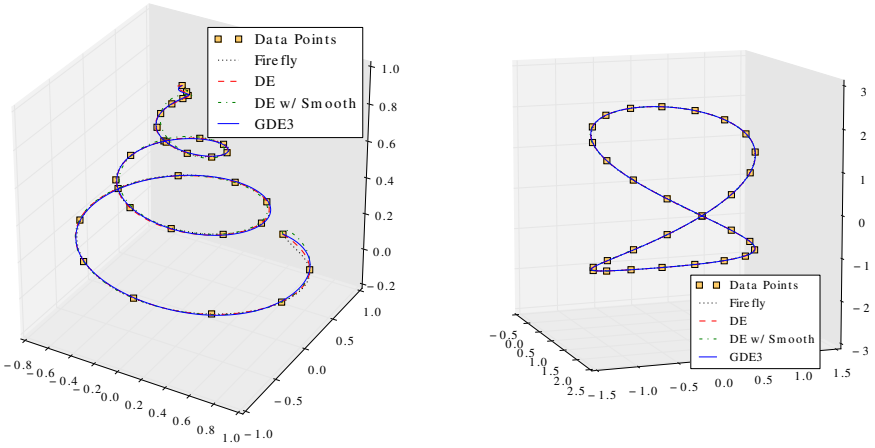
Fig. 5. Comparison between the four tested methods with a a) set of tightly-packed data points, and b) epicycloid curve.

In the fifth and sixth examples (Figures 6(a) and 6(b)), because of the curve's simplicity, the algorithms performed similarly.

Overall, from Figures 4(a) to 6(b) it is possible to notice that the results obtained by the method are more visually pleasing than those obtained by the other

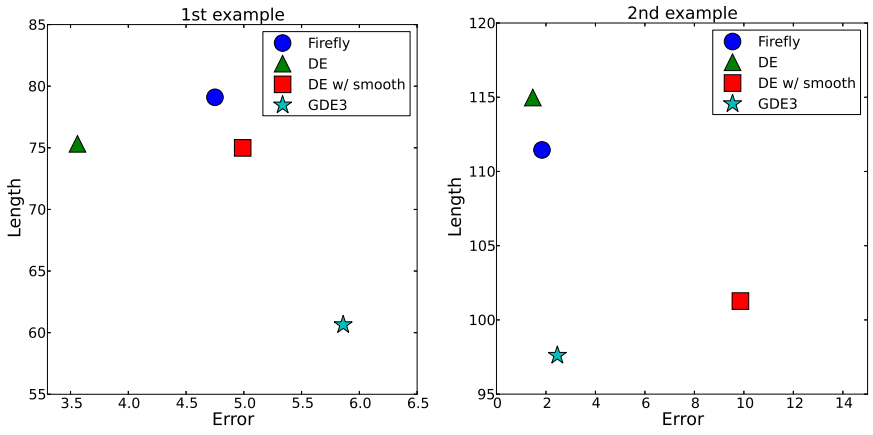
algorithms.

In order to compare the algorithms concerning both objectives, Figures 7(a) through 9(b) present the Pareto plot for the average error and length values obtained by each algorithm in each example, as seen in Table 1. The graphs show that, in all cases, the proposed method is part of the non-dominated set of algorithms. For the first example, the non-dominated algorithms are DE, DE w/ smooth and GDE3.



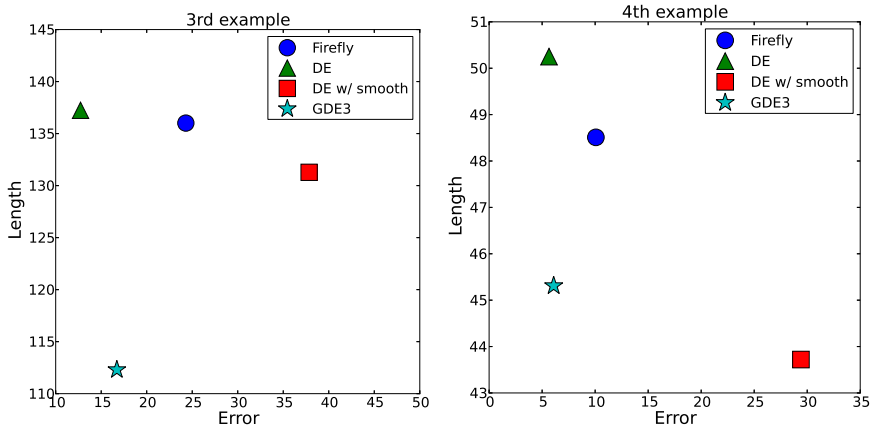
(a) 5th example (b) 6th example

Fig. 6. Comparison between the four tested methods with a a) conical spiral curve, and b) viviani curve.



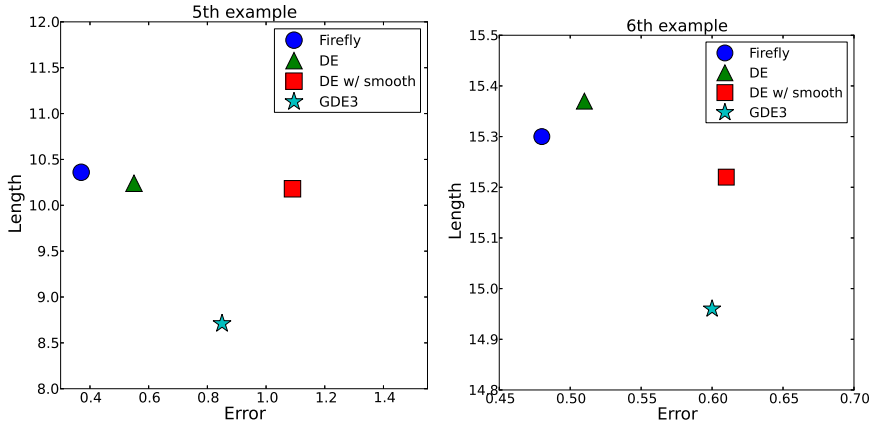
(a) 1st example (b) 2nd example

Fig. 7. Comparison of the results obtained in the a) first example and b) second example.



(a) 3rd example (b) 4th example

Fig. 8. Comparison of the results obtained in the a) third example and b) fourth example.



(a) 5th example (b) 6th example

Fig. 9. Comparison of the results obtained in the a) fifth example and b) sixth example.

For the second example, they are DE, Firefly and GDE3. For the third example, DE and GDE3, and for the fourth example, DE, DE w/ smooth and GDE3. For the fifth example, Firefly, DE and GDE3, and for the sixth example, Firefly and GDE3. It is also possible to notice that the proposed method provides solutions with good trade-off between error and length, leading to aesthetic curves in all cases.

Figure 10 presents the Pareto Front obtained by our method when executing the first example (Figure 4(a)). The other examples follow the same pattern. The results obtained were well distributed between length and error, indicating that the

method provides a diversified set of points.

Figure 11 presents the convergence graph of our method when executing the first example (Figure 4(a)), showing that the error and length converge to their minimum at similar rates. For the other examples, the same pattern is observed.

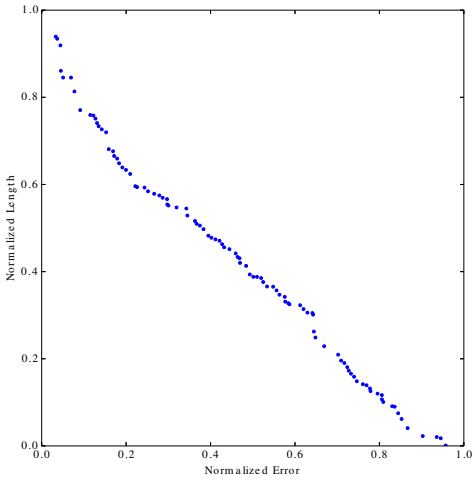


Fig. 10. Pareto Front for the first example.

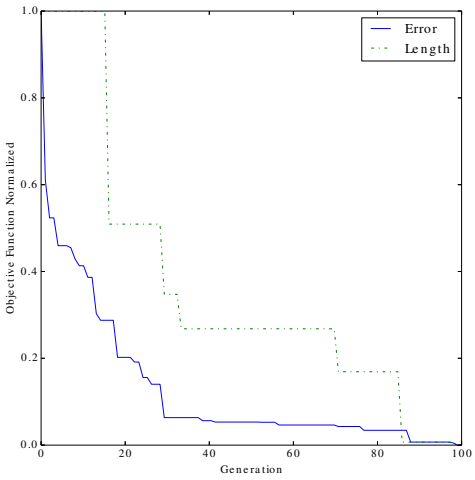


Fig. 11. Convergence plot for the first example.

6. Conclusions

This work proposed a parameter optimization method for the Bézier Curve Fitting problem. This approach uses a multiobjective Evolutionary Algorithm with two objective functions, minimizing error and length to find the best fit for a set of data points. Test results showed that the proposed method can produce better looking solutions than single objective methods when working with small number of data points. For sets of tightly-packed data points, the proposed method can perform similarly to others methods.

When compared to other methods, our method is in the non-dominated set of algorithms. It is also flexible, since the variable β can control the weight of the error function. A $\beta = 0$ makes the method fall back into a single objective problem, while higher values trade curve's error for a shorter curve. Since the choice of β happens after the execution of the algorithm, no heavy computation is required. This allows for a good trade-off between error and length leading to aesthetically pleasing curves in all cases.

Possible future works include adding more objective functions or using different ones, in order to better fit the curve for different types of applications. In addition, the application of the proposed approach in real data is planned.

References

Deb, K. *et al.* (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *Evolutionary Computation, IEEE Transactions on*, vol. 6, no. 2, pp. 182–197.

Engelbrecht, A.P. (2007). *Computational Intelligence: An Introduction*. Wiley Publishing.

Farin, G. (2002). *Curves and Surfaces for CAGD: A Practical Guide*, 5th edition. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Gálvez *et al.* (2007). Bézier curve and surface fitting of 3d point clouds through genetic algorithms, functional networks and least-squares approximation. *Computational Science and Its Applications – ICCSA*, ser. Lecture Notes in Computer Science, O. Gervasi and M. Gavrilova, Eds. Springer Berlin Heidelberg, vol. 4706, pp. 680–693.

Gálvez, A., Iglesias, A. (2013). Firefly algorithm for polynomial bézier surface parameterization. *Journal of Applied Mathematics*, vol. 2013.

Khan, M. (2012). A new method for video data compression by quadratic bézier curve fitting. *Signal, Image and Video Processing*, vol. 6, no. 1, pp. 19–24.

Kukkonen, S., Lampinen, J. (2005). GDE3: the third evolution step of generalized differential evolution. *Evolutionary Computation. The 2005 IEEE Congress on*, vol. 1, pp. 443–450 Vol.1.

Kukkonen, S., Deb, K. (2006). Improved pruning of non-dominated solutions based on crowding distance for bi-objective optimization problems. *Evolutionary Computation. CEC 2006. IEEE Congress on*, pp. 1179–1186.

Kumar, G., Kalra, P., Dhande, S. (2003). Parameter optimization for b-spline curve fitting using genetic algorithms. *Evolutionary Computation. CEC '03. The 2003 Congress on*, vol. 3, pp. 1871–1878.

Lee, E. (1989). Choosing nodes in parametric curve interpolation. *Computer-Aided Design*, vol. 21, no. 6, pp. 363 – 370.

Leu, M., Peng, X., Zhang, W. (2005). Surface reconstruction for interactive modeling of freeform solids by virtual sculpting. *CIRP Annals - Manufacturing Technology*, vol. 54, no. 1, p. 131–134.

Mortenson, M.E. (1997). *Geometric modeling*, 2nd edition. New York, NY, USA: John Wiley & Sons, Inc.

Parpinelli, R.S., Lopes, H.S. (2011). New Inspirations in Swarm Intelligence: a Survey. *International Journal of Bio-Inspired Computing*, vol. 3, no. 1, pp. 1 – 16. Inderscience Publishers.

Piegl, L., Tiller, W. (1997). *The NURBS Book*. ser. Monographs in Visual Communication Series. Springer-Verlag GmbH.

Pottmann, H., *et al.* (2005). Industrial geometry: recent advances and applications in cad. *Computer-Aided Design*, vol. 37, no. 7, pp. 751–766.

Prasad, M., Fitzgibbon, A.W. (2006). Single view reconstruction of curved surfaces. *CVPR*, vol. 2. IEEE Computer Society, pp. 1345–1354.

Price, K., Storn, R., Lampinen, J. (2005). *Differential Evolution: A Practical Approach to Global Optimization*. ser. Natural Computing Series. Springer.

Sevaux, M., Mineur, Y. (2007). A curve-fitting genetic algorithm for a styling application. *European Journal of Operational Research*, vol. 179, no. 3, pp. 895 – 905.

Srinivas, N., Deb, K. (1995). Multiobjective function optimization using nondominated sorting genetic algorithms. *Evol. Comput.*, vol. 2, no. 3, pp. 221 – 248.

Storn, R., Price, K. (1997). Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359.

Zhou, A. *et al.* (2011). Multiobjective evolutionary algorithms: A survey of the state of the art. *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32 – 49.