Curve Fitting Using Coevolutionary Genetic Algorithms

Nejat A. Afshar, Mohsen Soryani, and Adel T. Rahmani

Department of Computer Engineering,
Iran University of Science & Technology, Tehran, Iran
n_afshar@comp.iust.ac.ir, {soryani,rahmani}@iust.ac.ir

Abstract. Curve fitting has many applications in lots of domains. The literature is full of fitting methods which are suitable for specific kinds of problems. In this paper we introduce a more general method to cover more range of problems. Our goal is to fit some cubic Bezier curves to data points of any distribution and order. The curves should be good representatives of the points and be connected and smooth. Theses constraints and the big search space make the fitting process difficult. We use the good capabilities of the coevolutionary algorithms in large problem spaces to fit the curves to the clusters of the data. The data are clustered using hierarchical techniques before the fitting process.

Keywords: Curve fitting, Bezier curves, coevolutionary genetic algorithms, hierarchical clustering.

1 Introduction

Fitting of continuous and smooth curves to discrete data points is an essential task in many engineering problems. Geometric modeling, data analysis and image processing are some applications in which curve fitting is an important tool.

Many well established fitting methods are known, most of which are variants of least-squares techniques. These techniques perform well in problems with several defined parameters. For example in spline fitting, the process can be successful if the degree, knot positions, and the distribution of the data points are given and fixed. However in practice, these are not known, meanwhile these parameters influence the quality of the fitting result greatly. If they are not tuned properly, the fitting algorithm may have poor accuracy and the quality of the shapes will not be satisfactory. The complicated interdependence of the parameters and their influence on the fitting process is hard to control, but can be managed using evolutionary algorithms [1].

Evolutionary algorithms and their special case, genetic algorithms (GA) are a kind of stochastic search process inspired from Darwinian natural evolution, in which a population of candidate solutions are evolved. GAs are used in curve fitting [2-6] to avoid the complicated and unreliable process of finding the fitting parameters. [4] interpolates a cubic spline curve to data points by finding the knots using a genetic algorithm. Its goal is to minimize the curvature integral in order to reach an optimal shape. A real-coded genetic algorithm is developed in [2] to find good knots of a fitting spline. Data fitting with polygons to approximate an object curve is fundamental in pattern recognition, image processing and computer graphics. [5] reduces the integral square error between the curve and the polygon using a GA. Fitting of univariate cubic splines to noisy data points is sought in [3] by applying a genetic search to find the number of the knots and balancing the interpolating and smoothness capabilities of the fitting splines. A GA fits some curves to functional data in [6].

Most of the related works in this context are suitable for specific kinds of problems and perform well in their appropriate domain. For example some of the developed methods in curve fitting are intended for functional data points and cannot be used for nonfunctional data. Some other fitting techniques can be applied for nonfunctional data points but the structure and the distribution of data should be of some specific type, e.g. closed shapes, connected regions and data without noise and outliers. Moreover the data points in most of the works in the literature should be in an ordered list and cannot be distributed randomly on a plane.

In this paper we propose a method for fitting some cubic spline curves to data points on a plane, independent of the distribution of the data. This method can be used for functional and nonfunctional data and the structure of data does not matter, i.e. data points can be closed or open shapes, connected or not connected and noise and outliers do not affect the fitting quality. First the data points are clustered in a way that every group of data is suitable for fitting a cubic spline. Next for each cluster a genetic algorithm is run to fit a Bezier spline to the cluster. These genetic algorithms are run simultaneously and cooperatively, consisting of a coevolutionary genetic algorithm. The final solution of the fitting algorithm is derived by combining the partial solutions of the GAs.

2 The Proposed Fitting Method

It is difficult to fit splines to data points of any distribution, since there are many unknown parameters. The number of the curves and the position of the knots are two important parameters to be detected, and many of the works on the literature focus on determination of these parameters [2, 7].

Our goal is to fit some smooth Bezier splines to data points on 2-dimentional space in a way that the distances between the curves and the points are minimum and the curves are good representatives of the data. We assume that the input data to be fitted are given on a plane and can be of arbitrary distribution. The set of data points consists of N points, having real values and can be written as

$$D=\{(x_i,y_i)|x_i,y_i\in R\}, \qquad i=1,\ldots,N . \tag{1}$$

The output of our algorithm is a set of M cubic Bezier splines, joined together smoothly as necessary. Each Bezier spline curve is in parametric form and can be defined by two knots and two control points and is written as (2)

$$B_j(t)=\sum_{i=0}^{3}\binom{3}{i}t^i(1-t)^{3-i}P_{j,i} \ , t\in[0,1], \qquad j=1,\ldots,M . \tag{2}$$

By setting proper values for knots and control points, appropriate fitting curves can be obtained.

The proposed method consists of two major phases. First the data points are grouped together using hierarchical clustering techniques, in order to prepare points for the fitting phase. Next a coevolutionary genetic algorithm is run; in which M population of individuals evolve together simultaneously and cooperatively to fit smooth Bezier curves to the clusters. The number of the clusters from the first phase is M, equal to the number of the output curves. Since the appropriate number of the curves is not known, by using hierarchical clustering and choosing a specific level of the resulting dendrogram, a good value for the number of the curves can be obtained. The location of the knots and control points is set by the coevolutionary process. By moving the knots between the adjacent clusters, an optimum position can be determined. The clustering phase and the coevolutionary process are described in the upcoming sections.

2.1 The Clustering Algorithm

Clustering of data points is done in two steps. A single linkage clustering algorithm defines connected regions at the first step. At the second step, an average linkage clustering algorithm is run to partition each connected region to some related groups of data. These groups are later used in the genetic algorithm. The algorithm for the two step clustering is as follows.

```
(1) Compute $d_{avg}$, the average distance between the points in $D$.
(2) Create clusters $C_{SL}$ of points in $D$, using single linkage algorithm and $k*d_{avg}$
    as minimum distance between clusters.
(3) For each cluster $C_i$ in $C_{SL}$:
        a. Create clusters $C_{AL}$ of points in $C_i$, using average linkage algorithm
           and a measure as minimum distance between clusters.
        b. Add clusters in $C_{AL}$ to $C_{final}$.
(4) Output $C_{final}$.
```

The single linkage algorithm is responsible for putting different connected regions in different clusters. For this purpose a stop criterion is needed. If the distances between all the clusters are more than $k*d_{avg}$, the merging of the clusters stops.$d_{avg}$is a measure that shows the looseness of data points and can be computed as (3).

$$d_{avg}=\frac{1}{N}\sum_{i=1}^{N} \min d(D_i,D_j)\,, \qquad j\neq i\,. \tag{3}$$

Each of the connected regions is clustered by an average linkage algorithm with a distance measure for stopping the algorithm like the previous step. The larger the measure, the smaller the number of the clusters and respectively the number of the curves. By setting these measures properly, the appropriate number of curves is achieved.

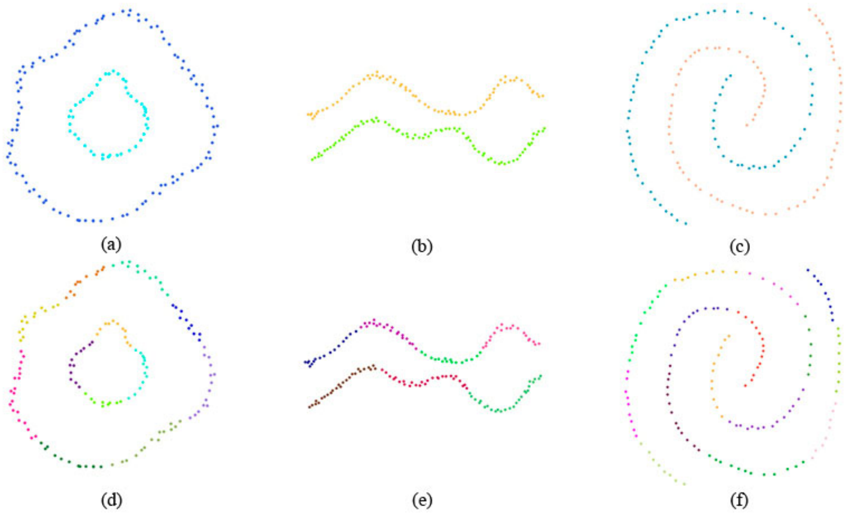

Fig. 1. Results of the clustering algorithm. Different clusters are shown in different colors. The results of the first step of the algorithm for the corresponding data points are shown in (a), (b) and (c). Final clusters are indicated in (d), (e) and (f).

In order to improve the quality of fitting, noise and outliers can be omitted in this phase. This is done by removing the clusters smaller than a specific amount. Fig. 1 shows some results of the proposed clustering algorithm.

As indicated in Fig. 1 (a), (b) and (c), the single linkage algorithm detects the connected regions. These regions are defined with respect to the distance between the neighbor points. By this method data points are prepared for curve fitting easily. Fig. 1 (d), (e) and (f) show the final clusters that are ready for the coevolutionary process.

2.2 The Coevolutionary Process

The groups of data points created by the clustering algorithm are used in a coevolutionary process to yield a set of well-fitted splines. The coevolutionary genetic algorithm consists of M population of individuals, each one representing solutions for a specific cluster of points. Recall that M is the number of clusters created in the previous phase. These GAs work together to overcome the constraints of the overall problem. Each GA cooperates with the GAs in its neighborhood. The clusters with adjacent knots (start or end points) have corresponding neighbor GAs. Fig. 2 shows four clusters and their corresponding knots. As indicated, the clusters C1, C2 and C3 are in a connected region and their corresponding curves should be merged together. The cluster C2 has two neighbors, C1 and C3 have one neighbor and there is no neighbor for the cluster C4. The GA for C2 acts in direct cooperation with C1 and C3, independent of C4.

The farthest points in a cluster are considered as the two knots of it. Adjacent clusters are identified using the distance between their knots and a measure like the one

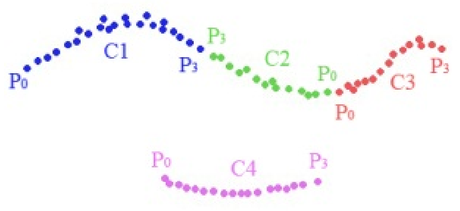

Fig. 2. Four clusters and their knots. Note that the start or end points of a cluster can be adjacent with even the start or end point of another cluster.

used in the previous section. If the distance between two knots of a cluster is less than k*d_{avg}, then the clusters are considered as neighbors. The knots between adjacent clusters should be merged, so the mean point between them is assigned as the new knot.

After defining the neighbor clusters and setting their common knots, the data points are ready for the coevolutionary fitting algorithm. The proposed coevolutionary genetic algorithm is as follows.

```
(1)  Randomly generate $M$ subpopulations of size $N_{pop}$ and evaluate each Bezier curve in each subpopulation.
(2)  Keep the best individuals in each subpopulation.
(3)  For gen = 1 to $G$ do
(4)      For i = 1 to $M$ do
(5)          Select $N_{pop}$ parents from subpopulation $P_i$, using tournament selection to breed offsprings.
(6)          Perform crossover on parents with the probability of crossover rate and keep $N_{pop}$ produced offsprings.
(7)          Perform mutation on the bred offsprings with the probability of mutation rate.
(8)          Evaluate each individual of the offsprings.
(9)          Perform a selection on $P_i$ and offsprings by tournament to get a new subpopulation $\acute{P}_i$ of the same size as $P_i$.
(10)         Replace $P_i$ with $\acute{P}_i$.
(11)         Keep the best curve in $P_i$ and obtain appropriate curves for the neighbor GAs, so that the splines remain smooth.
(12)         Correct the best curves in neighbor GAs and replace a part of their subpopulations with the revised curves.
(13)         Select the best curves from each subpopulation to form the final Bezier splines and output it.
```

Some important parameters are the population size N_{pop}, the number of generations, and the crossover and mutation rates. These parameters can be set through trial and test.

Smoothness is a constraint that the GAs cope with it cooperatively. After a solution is found in a subpopulation, the solutions of the neighbor GAs should be updated in a way that the sequence of the joined curves remains smooth. These revised solutions are found in step (11) of the coevolutionary genetic algorithm. After that, a part of the subpopulations of the neighbor GAs are replaced with corrected solutions.

A spline is smooth if the tangent vectors of the piecewise curves are equal in common knots. This is indicated in Fig. 3. As indicated, the control points before and after a knot are on a direct line with the knot. This constraint can be used to obtain smooth curves. (4) shows the mathematical notation for the smoothness constraint. In this equation k, c_1 and c_2 refer to the knot, the preceding and the subsequent control points respectively.

$$\frac{y_{c_1}-y_k}{x_{c_1}-x_k}=\frac{y_k-y_{c_2}}{x_k-x_{c_2}}. \tag{4}$$

When a solution is found in a subpopulation, the solution of a neighbor subpopulation is updated by correcting the position of its control point near the common knot. The new position is a point on the line passing from the knot and the new control point of the found solution, having the same distance from the knot. This ensures that the final curves obtained by combining the partial solutions, remain smooth.

The Bezier curves are encoded in the chromosomes simply using their knots and control points. Since cubic splines are used here, the genotype of a chromosome consists of four genes. Each gene encodes a point, including the x and y characteristics of the point. The coding strategy is shown in Fig. 4. The start and end points are P_0 and P_3 respectively, and the two control points are P_1 and P_2. The x and y in each gene have real values.

The distance of the points to the curves can be used as a good fitness measure. The fitness of a curve in subpopulation P_i can be computed as (5) and (6).

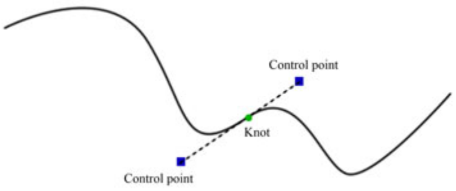

Fig. 3. An example of a smooth spline. Tangents of the two joined curves are equal at the knot

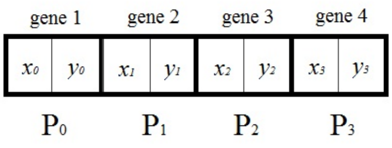

Fig. 4. The coding of Bezier curves in chromosomes

$$distance = \sum_{j=1}^{N_{C_i}} mind(c_{ij}, B_k) + \sum_{j=1}^{N_{C_L}} mind(c_{Lj}, B_k) + \sum_{j=1}^{N_{C_R}} mind(c_{Rj}, B_k)\,, \quad k = i, L, R, \tag{5}$$

$$\text{fitness} = -(\text{distance})\,. \tag{6}$$

In (5), C_i is the i^{th} cluster, N_{C_i} is the number of the points in the i^{th} cluster, c_{ij} is the j^{th} point in the cluster C_i, L and R are the index of the two neighbors of i, B_k is the Bezier curve in k^{th} subpopulation and d is the distance between a point and a curve. The curves of neighbor subpopulations are first smoothed in according to the new curve. The distance between a point and a parametric Bezier curve is computed by combining Newton's method and quadratic minimization. This method is fast and robust. For a detailed description see [8].

As indicated in (6), the total distance of the points to the curves is negated and returned as the fitness, so that a less distance corresponds to a more fitness. Notice that the fitnesses of neighbor solutions influence the fitness of the current solution.

The GA operations are applied in the order of crossover, mutation and selection. First a set of individuals are selected for the recombination. Then a one-point crossover is applied to tuples of the parents with the probability p_c. This is done by selecting a random position in the genotype of the parents and then splitting both parents at this point and creating the two children by exchanging the tails. The mutation is done by adding to the current gene values of x and y an amount drawn randomly from a Gaussian distribution with mean zero and standard deviation σ. This takes place within probability of p_m for every gene. Tournament selection is used for parent and survivor selection. This selection mechanism is simple and fast to apply and the selection pressure can be easily controlled by varying the tournament size k [9].

3 Experimental Results

In this section, some test examples are provided to show the effectiveness of the proposed method. All of the data points here are randomly distributed on a plane, having real values in x and y dimensions and lie within a unit square. Some of the parameters of the CGA used through the experiments are shown in Table 1.

Table 1. Parameters of the CGA used throughout the experiments

Parameter	Value
Subpopulation size	10
Number of generations	50
Crossover rate	0.9
Mutation rate	0.2
Tournament size	2
Number of runs	30

The parameter *k* for the clustering algorithm is 3 and the standard deviation for the mutation is 2 in the experiments. 30 runs are performed for each of the examples.

Fig. 5 shows the fitting results for the data points in Fig 1. As you can see, the points in Fig. 1 (c) are mingled together. Using the two step clustering algorithm, the data are prepared for the fitting algorithm. In the coevolutionary process, the positions of the knots are changed by genetic operators. As shown in Fig. 5, the final results of the CGA are a set of smooth curves that are good representatives of the data points.

Fig. 5 (a), (b) and (c) show the best curves in each subpopulation in the first generation of the best run of the coevolutionary process. As you can see the initial curves are approximately close to the final result. This is because of the good initialization of the individuals. The final results are shown in Fig. 5 (d), (e) and (f). These curves are the best solutions in each subpopulation in the 50th generation of the best run.

Fig. 6 (a), (b) and (c) show the average errors in each generation of the coevolutionary process. The average error is the average distance of the points to the curves. The dashed line in each diagram shows the average of the distances in 30 runs and the solid lines show the average distance in the best trial.

The experiments show that the coevolutionary process converges at about the 40th generation and only slight changes are made after that.

We have also compared our algorithm with some other curve fitting methods to show its effectiveness. The final results of our curve fitting method on two well-known data points are shown in Fig. 7. Table 2 presents a comparison with the proposed methods in [10-14]. n_c is the number of the curves and ISE stands for integral square error and is the sum of the squared distance of all the points to the curves. As shown in Table 2 most of the methods do not consider smoothness. The advantage of our method is the smoothness of the curves with a relatively low error.

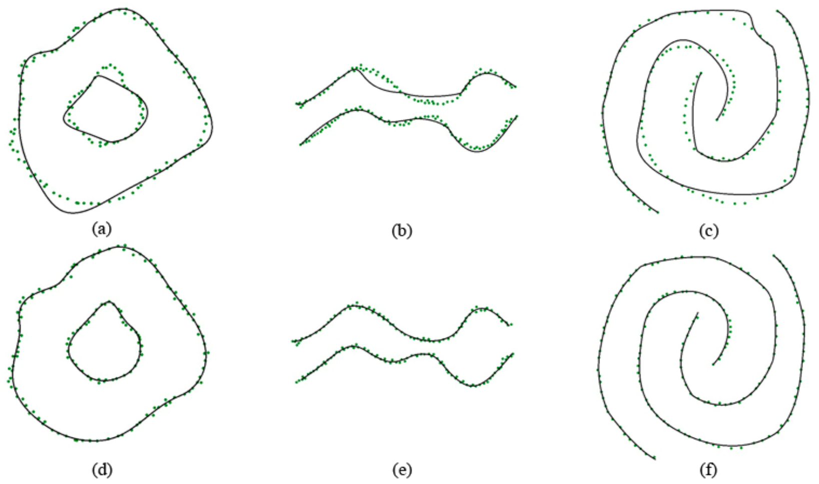

Fig. 5. Fitting results in the first and last generations. The final Bezier splines for the data points in Fig. 1 (a), (b) and (c) are presented in (d), (e) and (f) respectively.

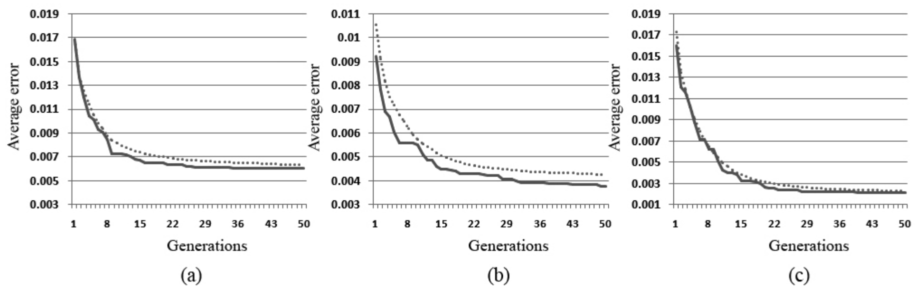

Fig. 6. The average error for the generations of the CGA in the experiments. (a), (b) and (c) show the diagrams for the data points in Fig. 5 (a), (b) and (c). The dashed lines show the average results of 30 runs and the solid lines show the best trial in the runs.

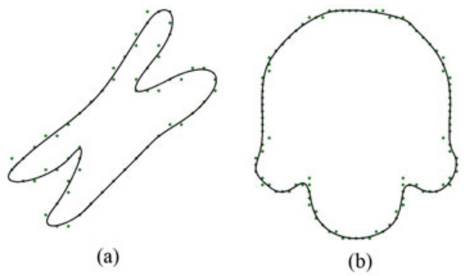

Fig. 7. Results of the proposed method on two well-known data points. A chromosome-shaped curve and four semicircles are shown in (a) and (b) respectively.

Table 2. Results of the proposed algorithm with five other methods of curve approximation

Method	Chromosome			Semicircles		
	Smooth	n_c	ISE	Smooth	n_c	ISE
Pei and Horng [10]	Yes	15	0.0171	Yes	12	0.0094
Pei and Horng [11]	No	10	0.0083	No	4	0.0060
Horng and Li [12]	No	10	0.0074	No	4	0.0060
Sarkar et al. [13]	No	10	0.0074	No	4	0.0060
	No	11	0.0072	No	6	0.0056
	No	15	0.0061	No	12	0.0037
Pal et al. (TDLSA) [14]	No	5	0.0102	No	4	0.0060
Pal et al. (ESA) [14]	No	5	0.0062	No	4	0.0056
Proposed Method	Yes	10	0.0071	Yes	10	0.0076

4 Conclusions

In this paper, a new method for fitting a series of Bezier curves to data points is proposed. The points are placed randomly on a plane and there is no knowledge about the distribution and the order of them. This method can be used for functional or nonfunctional data, data with closed or open shapes and can be scattered in different regions on the plane.

This method consists of two phases. First the data points are clustered using hierarchical clustering techniques. Next the prepared clusters are given to a coevolutionary process to fit a set of connected and smooth Bezier curves to them. The coevolutionary GAs can find good positions for the knots and the control points.

The experimental results show that the algorithm converges at about 40th generation. The final result is a set of continuous and smooth Bezier curves with a relatively low error. The major advantage of our method is that it can be used for data points with any distributions.

Our future work will be on more efficient clustering techniques to cope with more complex data and also using more capabilities of coevolutionary algorithms.

References

1. Renner, G., Ekart, A.: Genetic algorithms in computer aided design. Computer-Aided Design 35, 709–726 (2003)
2. Yoshimoto, F., Harada, T., Yoshimoto, Y.: Data fitting with a spline using a real-coded genetic algorithm. Computer-Aided Design 35, 751–760 (2003)
3. Manela, M., Thornhill, N., Campbell, J.: Fitting spline functions to noisy data using a genetic algorithm. In: Proceedings of the Fifth International Conference on Genetic Algorithms, pp. 549–556. Morgan Kaufmann Publishers Inc. (1993)
4. Markus, A., Renner, G., Vancza, J.: Spline interpolation with genetic algorithms. In: Proceedings of the International Conference on Shape Modeling and Applications, pp. 47–54. IEEE (2002)
5. Yin, P.: Polygonal approximation using genetic algorithms. Pattern Recognition, 838–838 (1999)
6. Gulsen, M., Smith, A., Tate, D.: A genetic algorithm approach to curve fitting. International Journal of Production Research 33, 1911–1924 (1995)
7. Yang, H., Wang, W., Sun, J.: Control point adjustment for B-spline curve approximation. Computer-Aided Design 36, 639–652 (2004)
8. Wang, H., Kearney, J., Atkinson, K.: Robust and efficient computation of the closest point on a spline curve, pp. 397–406 (2002)
9. Eiben, A., Smith, J.: Introduction to evolutionary computing. Springer, Heidelberg (2003)
10. Pei, S.C., Horng, J.H.: Fitting digital curve using circular arcs. Pattern Recognition 28, 107–116 (1995)
11. Pei, S.C., Horng, J.H.: Optimum approximation of digital planar curves using circular arcs. Pattern Recognition 29, 383–388 (1996)
12. Horng, J.H., Li, J.T.: A dynamic programming approach for fitting digital planar curves with line segments and circular arcs. Pattern Recognition Letters 22, 183–197 (2001)
13. Sarkar, B., Singh, L.K., Sarkar, D.: Approximation of digital curves with line segments and circular arcs using genetic algorithms. Pattern Recognition Letters 24, 2585–2595 (2003)
14. Pal, S., Ganguly, P., Biswas, P.: Cubic Bézier approximation of a digitized curve. Pattern Recognition 40, 2730–2741 (2007)