Applied Soft Computing 13 (2013) 1491–1504

Contents lists available at SciVerse ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

A new iterative mutually coupled hybrid GA–PSO approach for curve fitting in manufacturing

Akemi Gálvez, Andrés Iglesias*

Dept. of Applied Mathematics and Computational Sciences, University of Cantabria, Avda. de los Castros s/n, E-39005 Santander, Spain

ARTICLE INFO

Article history:
Received 30 December 2011
Received in revised form 12 May 2012
Accepted 24 May 2012
Available online 4 July 2012

Keywords:
Hybrid evolutionary systems
Genetic algorithms
Particle swarm optimization
Curve fitting
Reverse engineering
CAD/CAM
3D laser scanning
B-spline curves

ABSTRACT

Fitting data points to curves (usually referred to as *curve reconstruction*) is a major issue in computer-aided design/manufacturing (CAD/CAM). This problem appears recurrently in *reverse engineering*, where a set of (possibly massive and noisy) data points obtained by 3D laser scanning have to be fitted to a free-form parametric curve (typically a B-spline). Despite the large number of methods available to tackle this issue, the problem is still challenging and elusive. In fact, no satisfactory solution to the general problem has been achieved so far. In this paper we present a novel hybrid evolutionary approach (called IMCH-GAPSO) for B-spline curve reconstruction comprised of two classical bio-inspired techniques: genetic algorithms (GA) and particle swarm optimization (PSO), accounting for data parameterization and knot placement, respectively. In our setting, GA and PSO are mutually coupled in the sense that the output of one system is used as the input of the other and vice versa. This coupling is then repeated iteratively until a termination criterion (such as a prescribed error threshold or a fixed number of iterations) is attained. To evaluate the performance of our approach, it has been applied to several illustrative examples of data points from real-world applications in manufacturing. Our experimental results show that our approach performs very well, being able to reconstruct with very high accuracy extremely complicated shapes, unfeasible for reconstruction with current methods.

1. Introduction

Due to globalization, manufacturers are constantly challenged to optimize processes in order to deliver more competitive products with better quality and lower prices. Under these new rules of competition in global market, design is taking a central role in the product development lifecycle. The prior emphasis on designing whole product lines for customers has experienced a radical shift as companies increasingly enable customers to mass customize their own products. Under this new paradigm, products are manufactured in lesser amounts but with greater product diversity. As a result, the geometric model of the product has to be changed frequently during the design process, making CAD/CAM systems inevitable tools for the design and manufacturing processes.

As part of the initial conceptual design process, it is common in many industries (automotive, aerospace, ship building) to build prototypes in a real workshop with materials such as foam, clay, wood, metal or plastics, in order to explore ideas for shape, size, proportions and the human sense of interacting with the model. The model thus obtained is digitally stored through data sampling by means of technologies such as 3D laser scanning. Data points are then fitted to mathematical curves and surfaces, a process called *reverse engineering* [47,64,65]. Advantages of this process are obvious: digital models are easier and cheaper to modify and analyze than the physical counterparts. In addition, digital data are very well suited for efficient storage and transmission among providers and manufacturers and can still be used even when the real objects are lost or become unavailable.

Mathematical models in reverse engineering for design and manufacturing are usually represented in parametric form, particularly B-splines. In fact, owing to their advantages in terms of flexibility and control, B-splines have become the industry standard for CAD/CAM data representation [16,25]. However, to obtain a reliable curve reconstruction method with B-splines is not an easy task. After decades of intensive research, B-spline curve reconstruction is still a challenging and elusive problem as many data sets cannot be properly reconstructed yet. Theoretically, it is possible to reconstruct an optimal B-spline curve from the given data points by solving a nonlinear optimization problem with the number of control points, the control points, the parameter values of data points, and the knots as unknowns. To do so, three major steps must be carried out: data parameterization, knot placement and least-squares minimization. The last step is transformed into that of solving a linear system, provided that the first two steps

* Corresponding author.
E-mail address: iglesias@unican.es (A. Iglesias).

1568-4946/$ – see front matter © 2012 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.asoc.2012.05.030

have been properly attained. It is important to remark, however, that such steps cannot be solved separately, because the choice of knots influences the parameterization of data points and conversely. Unfortunately, performing both steps simultaneously leads to a (very difficult) high-dimensional, continuous, multimodal and multivariate nonlinear optimization problem. This explains why all methods reported in the literature focus exclusively on an individual step and skip the other one, either data parameterization or knot placement. In such methods, the unknown parameters of the missing step are either assumed a priori or obtained approximately according to some given formula or heuristics (see Section 2 for details).

Due to their good behavior for complex optimization problems involving ambiguous and noisy data, there has recently been a growing interest in applying bio-inspired optimization techniques (such as meta-heuristics and evolutionary methods) to the curve reconstruction problem [17,55,62,68,69]. But, once again, they are still applied to one particular step and, hence, only partial solutions have been achieved. Those methods can be sufficient for simple (almost trivial) examples, but (as expected) they are not able to reconstruct complicated shapes [55,68]. In other cases, they are more powerful but only applicable to specific cases [17,69]. In our experience, the three steps mentioned above must be fully addressed in order to solve the general curve reconstruction problem. Unfortunately, no satisfactory solution to this problem has been reported so far, a clear indication of the extreme difficulty of this task.

It is well-known that the success of bio-inspired techniques relies largely on an adequate trade-off between two opposite terms: *exploration* and *exploitation*. The former accounts for *diversification*, i.e., to generate diverse solutions so as to explore the search space on a global scale of the solution domain, while the latter accounts for *intensification*, i.e., to concentrate the effort in a local region around a current good solution searching for better solutions in its neighborhood. On the other hand, as evidenced by the "no free-lunch" theorem, bio-inspired techniques are problem-dependent and none of them can claim superiority over the others for all optimization problems. These two facts open the door to the integration of evolutionary algorithms and other meta-heuristics into a single approach, the so-called *hybrid evolutionary methods*. This new paradigm, already suggested by some authors for over a decade [2,11], is gaining popularity in recent years because of its ability to combine the strengths of different methods under a common scheme. In particular, the hybridization of genetic algorithms (GA) with particle swarm optimization (PSO) has shown to improve the individual performance of each method for several problems (see Section 3.3 for further details about this issue).

1.1. *Aim and structure of the paper*

Despite the promising results of these hybrid evolutionary approaches, no serious attempt has been done so far to apply them in the context of curve reconstruction. The present contribution is intended to fill this gap. On the other hand, we remark that existing hybrid GA–PSO methods are not well suited to be directly applied to this problem, due to the strong interrelationship among a large number of (qualitatively different) unknowns. Therefore, in this paper we pursue a twofold objective:

- firstly, we report the first case of application of hybrid evolutionary methods to B-spline curve reconstruction for reverse engineering, a very important issue in industrial design and manufacturing;
- secondly, we introduce a new hybrid evolutionary method, called *IMCH-GAPSO* (that stands for *i*terative *m*utually *c*oupled *h*ybrid *g*enetic *a*lgorithms–*p*article *s*warm *o*ptimization), specially designed to tackle this problem. Our hybrid approach is very general, overcomes the limitations of existing approaches and does not require any knowledge about the structure of the underlying function of data; our only input are the data points. Furthermore, it can be successfully applied to reconstruct any arbitrary curve of complicated shape, including those unfeasible for reconstruction with current methods. All these claims will be adequately justified throughout the paper.

The structure of this paper is as follows: firstly, previous work on B-spline curve reconstruction is discussed in Section 2. Then, a gentle overview on genetic algorithms, particle swarm optimization and their hybrid approaches is given in Section 3. Some basic concepts about B-spline curves are briefly presented in Section 4. The core of the paper is in Section 5, where the fitting problem and our proposed method are reported in detail. Section 6 describes our experimental results for several illustrative examples from real-world applications in manufacturing. A comparison of our method with other alternative approaches is also discussed in this section. The paper closes with the main conclusions and our plans for future work.

2. Previous work

Intensive research on curve reconstruction has been carried out during the last decades. Most of these methods address the problem by using fixed knots [28,41,42,45,66,67]. In general, these methods apply either a given formula or some kind of heuristics to determine a suitable choice of knots. In the former case, the averaging method described in [45] is commonly applied. The later case includes a myriad of methods using error bounds [41], vector-support machines [28], dominant points [42] or curvature-based squared distance minimization [66,67]. However, it has been shown that curve fitting by splines improves dramatically if the knots are treated as free variables [4,52]. As a consequence, considerable effort has been placed upon the determination of suitable knot vectors and several methods for variable knots have been proposed. The general approach consists of starting with a certain number of knots and iteratively modify such amount to satisfy the error bound [9,30,36,37,48]. This strategy has usually been applied in two different (opposite) ways: (1) start with a small number of knots and increase that number by inserting new knots (*knot insertion*) or (2) start with a large amount of knots and reduce it by removing some knots (*knot removal*). Unfortunately, these methods usually require human intervention to specify either a tolerance error or a smoothing factor, whose determination is often based on subjective factors [9,36,37]. Other methods, such as [48], produce unnecessary redundant knots. In general, these methods can perform reasonably well under human intervention but fail to *automatically* generate a good knot vector. A different approach is based on the idea of using curvature information extracted from input data [8,51]. Unfortunately, such algorithms are restricted to smooth data points. A more recent adaptive iterative knot placement method is proposed in [34]. The method performs well but it is restricted to differentiable functions. Furthermore, the method relies heavily on the determination of feature points (such as curvature extrema and inflection points), which are hard to be obtained. Finally, since noise in the curvature is more severe than noise in data themselves, all these methods are strongly affected by noise intensity.

The multimodal nature of the B-spline curve reconstruction problem demands powerful global optimization methods for solving it. As a consequence, there is an increasing interest about the application of bio-inspired techniques to this problem. However, there are still very few references on the subject in the literature [17,55,62,68–70]. The method in [68] converts the original continuous nonlinear and multivariate optimization problem into a discrete combinatorial optimization problem, which is subsequently solved through genetic algorithms. A similar idea is applied in [55] to capture a pleasant looking spline fitting for shapes such as fonts. A more recent version of this discrete approach where genetic algorithms are replaced by artificial immune systems is given in [62]. Those methods are improved in [69] by using a real-code genetic algorithm in which knots themselves are used as genes. By focusing on the original continuous optimization problem, discretization errors are avoided. None of these methods can fully cope with underlying functions of data that exhibit discontinuities and cusps. This issue is solved in [17] by using particle swarm optimization. However, all these previous methods are designed for explicit functions exclusively, so the interesting case of parametric curves (by far, the most important one in industrial settings) is not addressed. A very recent stochastic optimization approach has been proposed in [70] where knots are computed through estimation of distribution algorithms using Gaussian mixture distributions and clustering techniques. The method performs well but it is limited to closed curves and no multiple knots are allowed.

Amazingly enough, all those previous methods focus on a specific step of the curve reconstruction problem, either data parameterization (fixed-knots methods) or knot placement (variable knots methods). So far, no evolutionary approach has been successfully applied to the general curve reconstruction problem. In this regard, the present paper represents a significant milestone in the field and paves the way for future developments in this area.

3. Genetic algorithms, particle swarm optimization and hybrid techniques

There are three main reasons to apply bio-inspired techniques to the curve reconstruction problem:

- *They can be used even when we have very little information about the problem*. Most papers discuss only academic examples whose data have some kind of topological or geometric structure. This does not happen, however, in real-world reverse engineering applications, where typically little or no information about the problem is known beyond the data points.
- *They can be successfully applied to multimodal and multivariate nonlinear optimization problems*. Bio-inspired techniques are able to find optimal solutions to nonlinear optimization problems in high-dimensional search spaces. Curve reconstruction is one of such problems. Moreover, due to the (potential) existence of many local optima of the least-squares objective function, it is also a multimodal problem. Bio-inspired techniques are well suited for multimodal problems.
- *They can deal with noisy data*. Laser scanner systems yield an enormous amount of irregular data points. They are also known to be less accurate than their contact-based counterparts [26]. As a consequence, reconstruction methods must be robust against this measurement noise.

As we mentioned above, in this paper we introduce a new hybrid GA–PSO approach. In next paragraphs we describe briefly its two components, genetic algorithms and particle swarm optimization. Then, a brief survey on hybrid GA–PSO techniques is also given.

3.1. Genetic algorithms

Genetic algorithms (GA) have been recognized as one of the most powerful computational paradigms for optimization and global search problems. Originated from the seminal work of John Holland in the 1970s [24], they combine bio-inspired processes emulating genetic evolution (namely, natural selection, mutation and crossover) in order to describe the growth and development of populations associated with the target problem.

Algorithm 1 *(Pseudocode version of the genetic algorithm).*

```
{Initialization}
  $g \leftarrow 0$                                     /* g: generation counter */
  for $i = 1$ to $M$ do                               /* M: population size */
    Initialize individuals $x_i$ to random values
    $F_i \leftarrow f(x_i)$                           /* f: fitness function */
  end for
  Pop $\leftarrow \{x_1, x_2, \ldots, x_M\}$
  F $\leftarrow \{F_1, F_2, \ldots, F_M\}$
{Main Loop}
  while (not termination condition) do
    {Genetic Operators}
      Pop $\leftarrow$ Selection(Pop, F)
      Pop $\leftarrow$ Crossover(Pop, $p_c$)          /* $p_c$: probability of crossover */
      Pop $\leftarrow$ Mutation(Pop, $p_m$)           /* $p_m$: probability of mutation */
    {Evaluation Loop}
      for $i = 1$ to $M$ do
        $F_i \leftarrow f(x_i)$
      end for
    F $\leftarrow \{F_1, F_2, \ldots, F_M\}$
    $g \leftarrow g + 1$
  end while
```

In GA populations are formulated as abstract representations (called *chromosomes*) of candidate solutions (called *individuals* or *phenotypes*) to an optimization problem. Typically, the algorithm maintains a population of M individuals $\mathbf{Pop}(g) = \{x_1(g), \ldots, x_M(g)\}$ for each iteration g (also called generation), where each individual represents a potential solution of the problem. The algorithm is an iterative process in which new populations are obtained using a selection process based on individual adaptation and some "genetic" operators (crossover and mutation). In each generation, the fitness (a measure of the quality of the represented solution) of every individual in the population is evaluated. The individuals with the best adaptation measure have more chance of reproducing and generating new individuals by crossing and muting. The selection process is repeated several times and the selected individuals form a tentative new population for further genetic operator actions. After selection some of the members of the new tentative population undergo transformations. A *crossover* operator creates two new individuals (*offsprings*) by combining parts from two randomly selected individuals of the population. In GA the crossover operator is randomly applied with a specific probability, p_c. A good GA performance requires the choice of a high crossover probability. *Mutation* is a unitary transformation which creates, with certain probability, p_m, a new individual by a small change in a single individual. In this case, a good algorithm performance requires the choice of a low mutation probability (inversely proportional to the population size). The mutation operator guarantees that all the search space has a nonzero probability of being explored. Using these genetic operators, the general structure of the algorithm is sketched in Algorithm 1.

This procedure is repeated several times (thus yielding successive generations) until a termination condition has been reached. Common terminating criteria are that a solution is found that satisfies a lower threshold value, or that a fixed number of generations has been reached, or that successive iterations no longer produce better results. Ideally, the algorithm is expected to evolve over time toward better solutions, although convergence to global optima cannot be generally assured. The interested reader is referred to the nice books in [3,22,39] for an in-depth overview about genetic algorithms, their main features, variants and applications.

3.2. Particle swarm optimization

Particle swarm optimization (PSO) is a very popular bio-inspired technique for real-valued optimization problems where potential solutions (called *particles*) are distributed over the search space and provided with an initial velocity and the capacity to communicate good positions to each other and adjust their own position and velocity based on these good positions. The dynamics of the particle swarm is considered along successive iterations, like time instances. Evolution of particle *i* is regulated by two memory factors: memory of their own best position and knowledge of the global or their neighborhood's best. For the former, we store the coordinates P_i^b associated with the best solution (fitness) it has achieved so far along with the corresponding fitness value, $f(P_i^b)$. For the latter, we also collect the position, P_g^b, and the best fitness value, $f(P_g^b)$, among all the particles in the population from the initial iteration. This is a global information for modifying the position and velocity of each particle *i* according to the evolution equations:

$$V_i(s+1) = w\,V_i(s) + \gamma_1 R_1 \otimes [P_g^b(s) - P_i(s)] + \gamma_2 R_2 \otimes [P_i^b(s) - P_i(s)] \tag{1}$$

$$P_i(s+1) = P_i(s) + V_i(s) \tag{2}$$

where $P_i(s)$ and $V_i(s)$ are respectively the position and the velocity of particle *i* at time *s*, *w* is called *inertia weight* and decide how much the old velocity will affect the new one and coefficients γ_1 and γ_2 are constant values called *learning factors*, which decide the degree of affection of P_g^b and P_i^b. In particular, γ_1 is a weight that accounts for the "social" component, while γ_2 represents the "cognitive" component, accounting for the memory of an individual particle along the time. Two random numbers, R_1 and R_2, with uniform distribution on [0, 1], are included to enrich the searching space. The symbol ⊗ denotes point-wise vector multiplication. Finally, a fitness function must be given to evaluate the quality of a position.

In the original PSO algorithm (called *global PSO*) each particle has two attractors: its own previous best position, and the swarm's global best position. Alternatively, we can use the best position of a sub-swarm (neighborhood) of the particle (*local PSO*). This allows parallel exploration of the search space while reducing the probability of the PSO to fall into local minima, at the price of slow convergence speed. But, because smaller neighborhoods lead to slower convergence while larger neighborhoods yield faster convergence, most PSO methods consider the global approach instead of a local approach. This is also the approach taken in this work.

Algorithm 2 *(Pseudocode version of the PSO algorithm).*

```
{Initialization}
  $s \leftarrow 0$                                          /* s: time variable */
  for $i = 1$ to $N$ do                                     /* N: size of the swarm */
    Initialize vectors $V_i$ and $P_i$ to random values
    $P_i^b \leftarrow P_i$
  end for
  $P_g^b \leftarrow best\{P_i^b\,;\, i = 1, \ldots, N\}$    /* initial global best */
{Main Loop}
  while (not termination condition) do
    {Evaluation Loop}
      for $i = 1$ to $N$ do
        if $f(P_i)$ is better than $f(P_i^b)$ then          /* f: fitness function */
          $P_i^b \leftarrow P_i$                            /* particle's best position */
        end if
        if $f(P_i^b)$ is better than $f(P_g^b)$ then
          $P_g^b \leftarrow P_i^b$                          /* swarm's best position */
        end if
      end for
    {Update Loop}
      for $i = 1$ to $N$ do
        $V_i \leftarrow w \cdot V_i + \gamma_1 \cdot R_1(0,1) \otimes (P_g^b - P_i) + \gamma_2 \cdot R_2(0,1) \otimes (P_i^b - P_i)$
        $P_i \leftarrow P_i + V_i$
      end for
      $s \leftarrow s + 1$
  end while
```

This procedure is repeated several times (thus yielding successive generations) until a termination condition is reached. Common terminating criteria are that a solution is found that satisfies a lower threshold value, or that a fixed number of generations has been reached, or that successive iterations no longer produce better results. This PSO procedure is sketched in Algorithm 2.

As the swarm iterates, the fitness of the global best solution improves so the swarm eventually reaches the best solution. Original PSO algorithm was first reported in 1995 by Kennedy and Eberhart in [32] (see also [13]). The reader is referred to the excellent book in [33]. See also [14] for a gentle analysis of PSO from a computational point of view.

3.3. Hybridization of GA and PSO

The key reason for hybridization is to take advantage of the strengths of each individual technique while simultaneously overcoming its main limitations. In this regard, GA and PSO have often been hybridized in order to achieve integration of their complementary features [59]. The basic idea is to overcome critical problems of these methods such as premature convergence (particles in PSO can get stuck in a poor region of the search space), memory loss (if an individual in GA is not selected, its information is lost) and parameter tuning. In fact, hybrid evolutionary systems based on GA and PSO have shown to outperform its individual components for a number of (constrained and unconstrained) optimization problems. In addition, both GA and PSO have been individually applied in the context of surface reconstruction with excellent results [18,19].

Roughly speaking, hybrid methods of GA and PSO can be grouped into three types. The first one is based on the idea to apply GA (PSO) to generate the initial population for PSO (GA); then, the complementary method is used in a rather standard way. Examples of this type are the GA–PSO and PSO–GA methods in [40,53]. In the second group, the population is divided into two parts, which are evolved separately by GA and PSO, respectively. The resulting populations are then recombined in the update population and again divided into two parts in the next iteration for further application of GA and PSO, and so on. Examples of this approach are the GSO technique in [23], which uses a hybridization parameter *HC* to account for the percentage of population evolved with GA in every iteration (with $HC = 0$ for pure PSO, $HC = 1$ for pure GA and $0 < HC < 1$ for mixed cases), and the GA–PSO method in [31], where a *D*-dimensional problem is solved by considering an initial population of 4*D* individuals that is sorted according to the fitness value and then divided into two parts: the top 2*D* individuals undergo a real-code GA to generate 2*D* offsprings, subsequently used to update the remaining 2*D* individuals for PSO. Other examples are the HGAPSO method in [54] and the GA/PSO method in [27]. Finally, the third group uses more exotic approaches. Examples include the HPSOM in [15] where PSO is coupled with the GA mutation operator only, the HGAPSO in [29] where offsprings are created not only by GA operators of crossover and mutation but also by PSO, the Breeding Swarm method in [57] where an additional parameter called the breeding ratio is used to determine the proportion of the population which undergoes breeding in the current generation and a new crossover operator (called VPAC) incorporating the PSO velocity vector is used to prevent premature convergence, and the GA + PSO method in [63] where fuzzy rules are applied to choose between

GA individuals and PSO individuals. The method presented in this paper falls within this third group.

4. B-spline curves

Let $\mathcal{U} = \{0 \leq a = u_0, u_1, u_2, \ldots, u_{r-1}, u_r = b\}$ be a nondecreasing sequence of real numbers on the interval $[a, b]$ called *knots*. $\mathcal{U}$ is called the *knot vector*. For each sequence of knots as above, we can construct the *i-th B-spline basis function $N_{i,k}(t)$ of order k* (or equivalently, degree $k-1$), defined by the Cox–de Boor recursive relations:

$$N_{i,1}(t) = \begin{cases} 1 & \text{if } u_i \leq t < u_{i+1} \\ 0 & \text{otherwise} \end{cases} \qquad (i = 0, \ldots, r-1) \tag{3}$$

and

$$N_{i,k}(t) = \frac{t - u_i}{u_{i+k-1} - u_i} N_{i,k-1}(t) + \frac{u_{i+k} - t}{u_{i+k} - u_{i+1}} N_{i+1,k-1}(t) \tag{4}$$

for $i = 0, \ldots, r-k$ with $k > 1$. Note that in (4) the knots are not only in the numerator but also in the denominator. This means that $N_{i,k}(t)$ is a nonlinear function of $\mathcal{U}$ (although, since $\mathcal{U}$ is often given a priori, the notation $N_{i,k}(t, \mathcal{U})$ is rarely used). Note also that *i*-th B-spline basis function of order 1 in Eq. (3) is a piecewise constant function with value 1 on the interval $[u_i, u_{i+1})$, called the *support* of $N_{i,1}(t)$, and zero elsewhere. This support can be either an interval or reduce to a point, as knots u_i and u_{i+1} must not necessarily be different. If necessary, the convention $0/0 = 0$ in Eq. (4) is applied.

A *B-spline curve of order k* (or degree $k-1$) is a piecewise polynomial parametric curve given by:

$$\mathbf{C}(t) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,k}(t) \tag{5}$$

where $\{\mathbf{P}_i\}_{i=0,\ldots,n}$ are vector coefficients called *control points* as they roughly determine the shape of the curve, and $N_{i,k}(t)$ are the basis functions defined above. Note that in this paper vectors are denoted in bold. Without loss of generality, parameter t can be assumed to take values on the interval $[a, b] = [0, 1]$. For a proper definition of a B-spline curve in Eq. (5), the following relationship must hold: $r = k + n$ (see [45] for further details).

The knot vectors can be classified into two groups. The first one is the *uniform knot vector*; in it, each knot appears only once and the distance between consecutive knots is always the same. As a consequence, each basis function is similar to the previous one but shifted to the right according to such a distance. A qualitatively different behavior is obtained when any of the knots appears more than once (this case is usually referred to as *non-uniform knot vector*). The most common case of non-uniform knot vectors consists of repeating the end knots as many times as the order while interior knots appear only once (such a knot vector is called *non-periodic knot vector*). In general, a B-spline curve does not interpolate any of its control points; interpolation only occurs when end knots are repeated as many times as the order of the curve. In such a case, the B-spline curve does interpolate the first and last control points [25,45], that is, $\mathbf{P}_0 = \mathbf{C}(0)$ and $\mathbf{P}_n = \mathbf{C}(1)$. This is achieved by setting: $u_0 = u_1 = \cdots = u_{k-1} = a$, $u_{n+1} = u_{n+2} = \cdots = u_r = b$. Since they are the most common in computer design and manufacturing, in this work we will restrict ourselves to non-periodic knot vectors. Note however that our method does not preclude any other kind of knot vectors to be used instead.

5. IMCH-GAPSO method for B-spline curve reconstruction

5.1. The problem

Let us suppose that we are provided with a set of (possibly noisy) data points, $\{\mathbf{Q}_j\}_{j=1,\ldots,m}$, obtained through a laser scanning process, as it is commonly done in reverse engineering for design and manufacturing. Our goal is to obtain a B-spline curve, $\mathbf{C}(t)$, defined as in Eq. (5), that approximates the data points in the least-squares sense. The order of the curve, k, and the number of control points, $n+1$, are freely given by the user. Choice of k is important because low-order polynomials give little flexibility in controlling the shape of the curve while high-order polynomials can introduce unwanted wiggles and require more computation; therefore, too low and too high values for k should be avoided. A classical choice is to consider fourth-order B-splines and so will we in this paper. Note, however, that our method does not depend on the value of k. The number of control points defines the degrees of freedom available to change the shape of the curve, and also, the number of spans for the piecewise polynomial curve. Such a number is a problem-dependent parameter; as a rule of thumb, a higher number of control points is required for twisted shapes while fewer control points are needed for smooth shapes. On the other hand, it is advisable to keep this number as low as possible in order to minimize our storage and computation needs. In this context, the curve reconstruction process consists of three major steps that have to be solved:

(1) *data parameterization*: in this step, a set of parametric values $\{t_j\}_{j=1,\ldots,m}$ for the data points are obtained so that each $t_j \in [0, 1]$ is associated with $\mathbf{Q}_j$, $j = 1, \ldots, m$.
(2) *knot placement*: in this step, we obtain a suitable choice of knot vector $\mathcal{U}$ (defined as in the previous section) for the B-spline curve.
(3) *least-squares minimization*: in this step, control points of the B-spline curve are calculated by minimizing the least-squares error, Q, defined as the sum of squares of the residuals:

$$\begin{aligned} Q &= \min \left[\sum_{j=1}^{m} (\mathbf{Q}_j - \mathbf{C}(t_j))^2 \right] \\ &= \min \left[\sum_{j=1}^{m} \left(\mathbf{Q}_j - \sum_{i=0}^{n} \mathbf{P}_i N_{i,k}(t_j) \right)^2 \right] \end{aligned} \tag{6}$$

Note that the objective function Q depends on the B-spline coefficients and the knots. This problem is well known to be a continuous multimodal and multivariate nonlinear minimization problem. Note also that this problem has a large number of unknowns for large sets of data points, a case that happens very often in practice.

5.2. The proposed method

In this paper we introduce a new hybrid evolutionary approach to address the B-spline curve reconstruction problem described in previous paragraphs. Our proposal is summarized in Fig. 1. Our input (a given set of data points obtained from a laser scanning process) is to be fitted with an unknown B-spline curve of a certain order. To do so, we use an iterative scheme based on the hybridization of two classical bio-inspired techniques: genetic algorithms and particle swarm optimization (in purple and pink in Fig. 1, respectively). These techniques are applied to data parameterization and knot placement, respectively. Then, the fitting approximating curve is calculated (red column in Fig. 1) by LSQ (least-squares) through either SVD (singular value decomposition),

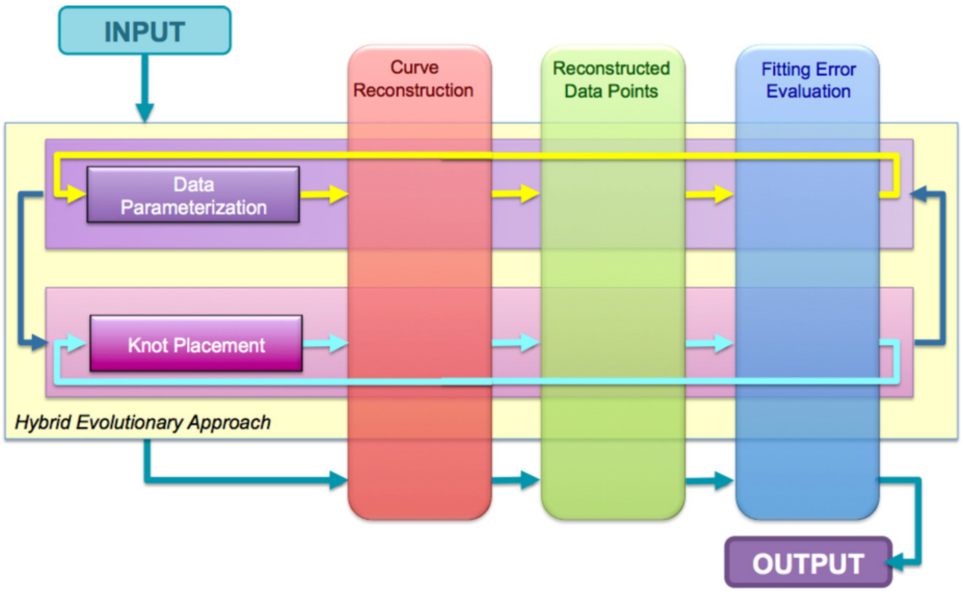

Fig. 1. Graphical workflow of the proposed method.

QR decomposition, a modification of LU method, or other techniques (see Section 5.3 for details). Evaluation of this fitting curve at parameter values yields reconstructed data points (in green), which are then compared with the original ones in order to compute the fitting error (in blue). This pipeline is then repeated iteratively until a termination criterion (such as a prescribed error threshold or a fixed number of iterations) for GA and PSO is attained.

A great novelty of our approach is that these algorithms are mutually coupled in the sense that the output of one system is used as the input of the other and vice versa. This coupling, also repeated iteratively, is needed to take into account the mutual interplay between the parameterization of data points and the choice of knot vector. The stopping criterion is that the difference between the values of both the parameters and the knots in two consecutive iterations is less than a threshold value ϵ, chosen as $\epsilon = 10^{-6}$ in this paper. This iterative mutually coupled hybrid GA–PSO system eventually returns the fittest vector of parameters accounting simultaneously for data parameterization and knot placement.

The individual steps of this method are described in detail in next paragraphs.

5.2.1. *Data parameterization*

First step consists of obtaining an adequate parameterization of data points. We perform this task by using a genetic algorithm, as described in Section 3.1. In order to do so, some aspects must be previously considered. First of all, a typical GA requires two elements to be defined prior to its use: (1) a genetic representation of each potential solution of the problem and (2) a measure of the quality of the solution (the fitness function). Since in this step we are interested in the assignment process of parameter values to data points, we propose the use of a real-coded genetic algorithm in which the genetic representation (chromosome) of an individual will be a real m-dimensional vector, where each coordinate represents the parameter value assigned to a data point.

As initial population we consider a randomly generated set of parameter vectors. To widen the search area of the algorithm it is desirable that the population size be large; however the computation time increases as this parameter rises, so a trade-off between both considerations is actually required. In general, a population of 100 individuals is used in the examples reported in this paper. We tested our results for populations up to 1000 individuals and obtained very similar results in all cases, meaning that our choice for this parameter is adequate. On the other hand, some standard parameterizations can be included in order to speed up the convergence of the process, such as the chord length method:

$$t_0 = 0, \quad t_m = 1, \quad t_j = t_{j-1} + \frac{|\mathbf{Q}_j - \mathbf{Q}_{j-1}|}{\sum_{l=1}^{m} |\mathbf{Q}_l - \mathbf{Q}_{l-1}|}$$

or the centripetal method [45]:

$$t_0 = 0, \quad t_m = 1, \quad t_j = t_{j-1} + \frac{\sqrt{|\mathbf{Q}_j - \mathbf{Q}_{j-1}|}}{\sum_{l=1}^{m} \sqrt{|\mathbf{Q}_l - \mathbf{Q}_{l-1}|}}$$

for $j = 1, \ldots, m - 1$. Regarding the fitness function to measure the quality of our assignment, it is the error function of the fitting process given by Eq. (6). Note that evaluation of this fitness function requires to specify a knot vector. Initially, it is chosen randomly and then optimized in the second step of our method. The algorithm then uses three genetic operators to obtain new populations of individuals: selection, crossover and mutation. In our case, the selection operator is implemented as the classical biased roulette wheel with slots weighted in proportion to individual fitness values. For crossover we use an one-point operator that randomly selects a crossover point within an individual, then swaps the two parent chromosomes to the right from this point and eventually sorts the obtained vectors to produce two new offsprings. This crossover operator is applied with probability $p_c = 0.9$ for the examples in this paper, although variations for this parameter in the range [0.8, 0.98] yield similar results. This is not the case for the computation times; our choice for this parameter provides the best performance in our experiments while computation times increase as p_c goes to the end points of this interval. Finally, as a mutation method we select the position h with worst fit error in the parameter vector of the solution and change the value of the selected parameter by the arithmetic mean of the previous and next parameters in the vector, that is, $t_h = (t_{h-1} + t_{h+1})/2$. Note that $t_{h-1} < t_h < t_{h+1}$, hence no sorting method is required. This mutation operator is applied with probability $p_m = 0.2$ in our examples. Variations of p_m within the interval [0.05, 0.35] yield similar results, once again at the expense of higher computation times for the extreme values in that interval. In our experiments, this effect is more noticeable for the leftmost end point but not dramatically. Finally, our termination criterion is that of not improving the solution after 10 consecutive generations.

5.2.2. *Knot placement*

In this step we seek to determine automatically a good choice of $\mathcal{U}$ yielding an extremely accurate data fitting. According to our previous assumptions, this problem consists in determining the internal knots $\{u_k, u_{k+1}, \ldots, u_n\}$. They are computed by using a global particle swarm optimization approach, as described in Section 3.2. To do this, we need to specify some PSO parameters:

- *Initial population, N*: unless otherwise stated, an initial population size of 100 individuals or particles has been considered. We also tested our method for populations up to 1000 particles and obtained similar results. All particles have been initialized with uniformly distributed random values on the interval [0, 1].
- *Inertia coefficient*, *w*: although we initially considered a fixed value for *w*, several studies reported in the literature show that it is convenient to change it dynamically, starting from a high value, which corresponds to a system where particles perform extensive exploration, and gradually reducing it to a lower value where the system would be better at homing into local optima [46]. We applied this strategy in this paper, starting at the initial value $w = 0.9$ and reducing it gradually until the final value $w = 0.4$. As expected, our computation times improved with respect to those for fixed values for *w*.
- *Social and cognitive parameters, γ_1, γ_2*: some references provide useful indications about good values for these coefficients [7,61]. Following them, both parameters have been set to $\gamma_1 = \gamma_2 = 2$ in this work, which are within the feasible domain of convergence for this method. Furthermore, variation of their values in the range $\gamma_1 + \gamma_2 < 4(w+1)$ leads to similar convergence rates, in good agreement with previous theoretical results of other authors [12,46,61].
- *Termination criterion*: we set a fixed number of iterations as our termination criterion. In general, a total of 10 iterations is used in this paper. We tested all our examples for larger numbers of iterations (up to 500) and found that they do not modify our results significantly, indicating that convergence to optimal values has already been achieved for such number of iterations.
- *Fitness function*: the fitness function is the error function given by (6), already used in the previous step for the same purpose.

5.2.3. *Least-squares minimization*

In this step we fit data points to a B-spline curve $\mathbf{C}(t)$ given by Eq. (5), which can be rewritten in matrix notation as:

$$\mathbf{Q} = \mathbf{N} \cdot \mathbf{P} \tag{7}$$

where $\mathbf{Q} = (\mathbf{Q}_1, \ldots, \mathbf{Q}_m)^T$, $\mathbf{P} = (\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n)^T$ is the vector of control points, and $\mathbf{N} = \{N_{i,k}(t_j)\}_{j=1,\ldots,m;i=0,\ldots,n}$ is the matrix of sampled basis functions computed by using the knot vector obtained in previous step. System (7) is overdetermined since *m* is generally assumed to be much larger than *n* and, therefore, it has usually no solution. So, alternatively, we compute $\mathbf{C}(t)$ by using the least-squares method. This method tries to determine the coefficients **P** that fit the equations better in the sense of solving the minimization problem:

$$Q = \min_{\mathbf{P}} ||\mathbf{Q} - \mathbf{N} \cdot \mathbf{P}||^2 \tag{8}$$

where $||\cdot||$ is the Euclidean norm. The necessary condition for **P** to be the solution of (8) is that:

$$\mathbf{N}^T \cdot \mathbf{N} \cdot \mathbf{P} = \mathbf{N}^T \cdot \mathbf{Q} \tag{9}$$

which leads to the normal equation:

$$\mathbf{M} \cdot \mathbf{P} = \mathbf{R} \tag{10}$$

where $\mathbf{M} = \left[\sum_{j=1}^{m} N_{l,k}(t_j) N_{i,k}(t_j)\right]$ and $\mathbf{R} = \left[\sum_{j=1}^{m} \mathbf{Q}_j N_{l,k}(t_j)\right]$ for *i*, $l = 0, \ldots, n$.

The algebraic solution of (9) is given by: $\mathbf{P} = \mathbf{N}^+ \cdot \mathbf{Q}$, where $\mathbf{N}^+$ denotes the Moore–Penrose pseudoinverse of **N**. If the matrix **N** has rank $n+1$, **M** is non-singular (or equivalently, $\mathbf{N}^T \cdot \mathbf{N}$ is invertible) and (10) defines **P** uniquely. Furthermore, an explicit formula for $\mathbf{N}^+$ is available as $\mathbf{N}^+ = (\mathbf{N}^T \cdot \mathbf{N})^{-1} \cdot \mathbf{N}^T = \mathbf{M}^{-1} \cdot \mathbf{N}^T$. Note that $\mathbf{N}^+$ is then a left inverse of **N**. Otherwise, $\mathbf{N}^+$ is defined by choosing **P** to minimize $||\mathbf{P}||^2$ among the solutions of (9).

5.3. *Implementation issues*

All computations in this paper have been performed on a 2.4 GHz. Intel Core 2 Duo processor with 4 GB of RAM. The source code has been implemented by the authors in the native programming language of the popular scientific program *Matlab*, version 2010b. In our opinion, *Matlab* is a very suitable tool for this task: it is fast and provides reliable, well-tested routines for efficient matrix manipulations. It also contains a bulk of resources regarding the solving of systems of equations. This feature proved to be very valuable in case of ill-conditioned matrices, i.e., with too large (or even infinite) condition number. This is a situation that can actually happen in practice, for instance, when one or several singular values in SVD decomposition are null or very near to zero. Advisable answer to this problem is to set reciprocals of such singular values to zero. *Matlab* command `svd` handles this situation for us. Another problem is that, although matrix **M** is nonnegative, symmetric and $2k-1$ banded, its computation explicitly to solve (10) is not computationally efficient and often a source of numerical rounding errors. A better alternative is to use the QR decomposition of **N**. In case **M** is well-conditioned and positive definite (that is, it has full rank), (10) can be solved directly by using the Cholesky decomposition. To this purpose, **M** is written as $\mathbf{M} = \mathbf{U}^T \cdot \mathbf{U}$ where **U** is an upper triangular matrix. After replacing this decomposition in (10), the solution is obtained in two stages, a forward substitution step for $\mathbf{U}^T \cdot \mathbf{Y} = \mathbf{N}^T \cdot \mathbf{Q}$, and then a backward step for $\mathbf{U} \cdot \mathbf{P} = \mathbf{Y}$. In practice, *Matlab* performs these steps automatically and provides us with the solution of such process in just a single execution. Similarly, when LU decomposition is used instead, *Matlab* command `lu` returns a suitable matrix factorization regardless the sparsity of the matrix, although different (mostly LAPACK and UMFPACK) routines are invoked in each case. In this paper, SVD has been primarily used since it provides the best numerical answer in the sense of least-squares for those cases in which the exact solution is not possible. To this aim, matrix **M** is decomposed as the matrix product $\mathbf{M} = \mathbf{U} \cdot \mathbf{\Phi} \cdot \mathbf{V}^T$ where **U** is a column-orthogonal matrix, $\mathbf{\Phi}$ is a diagonal matrix with positive or zero elements ϕ_k called the singular values and **V** is a square orthogonal matrix. Furthermore, since **M** is square, its inverse can readily be obtained as: $\mathbf{M}^{-1} = \mathbf{V} \cdot [\mathrm{diag}(1/\phi_k)] \cdot \mathbf{U}^T$. Also, *Matlab* provides us with the command `mldivide` to solve the equation (7) for both squared and non-squared systems (by using Gaussian elimination with partial pivoting and least-squares techniques, respectively). Depending on the general structure of matrix **N**, this command applies specialized LAPACK and BLAS routines to get the best possible solution to this system. Besides, *Matlab* provides excellent graphical options and optimized code for input/output interaction and high performance computations.

6. Experimental results

To evaluate the performance of our approach, we have applied it to several examples from different fields. To keep the paper at manageable size, we discuss here only four of them. Example 1 focuses on obtaining the mathematical expression of the baseball seam curve from a set of scanned data points, while examples 2, 3 and 4 discuss the fitting of data to parallel curves on a cell phone model, section curves of a paint spray gun, and helical curves on two different pieces, respectively. These three types of curves are very useful for generation of tool-path trajectories, an important issue in manufacturing.

These examples have been carefully chosen for two reasons: firstly, to show the diversity of situations to which our algorithm can be applied, and secondly, because they correspond to difficult and interesting real-world problems rather than to academic examples. We remark, however, that our method can also be applied to many other problems in computer design and manufacturing, such as those discussed, for instance, in [6,44,47,65]. As we will show below, our method provides a very accurate solution to these problems in terms of B-spline curves. Finally, a comparison of our method with previous approaches is also reported.

6.1. *Example 1: baseball seam curve*

The mathematical description of the curve traced out on the surface of a baseball by its seam has been the object of interest by several authors and an intriguing mathematical issue at its own (see, for instance, [1,35,60]). In this example, we obtain a solution to this problem in terms of a B-spline curve obtained through our hybrid method. The initial input consists of a set of 1127 scanned data points from the real baseball shown in Fig. 2 (top-left). Such data points are displayed in Fig. 2(top-right). The IMCH-GAPSO method is applied to a cubic B-spline curve with 28 control points and the set of parameters for GA and PSO described in Sections 5.2.1 and 5.2.2, respectively. The resulting curve is shown in Fig. 2 (bottom-left). As the reader can see, the method performs very well, with a root-mean-square error (RMSE) between the original and the reconstructed data points of $(6.9, 6.7, 2.1) \times 10^{-6}$ in this case and a CPU time of 3.7 s. Fig. 2 (bottom-right) shows the reconstructed curve and its underlying surface, displayed with a transparency factor $\alpha = 0.8$ for better visualization of the seam curve. Depending on our choice of curve parameters (particularly, the number of control points, n), fitting errors may vary from $10^{-2} - 10^{-3}$ for $n < 10$ to 10^{-6} for values of n larger than 24. It is worthwhile to mention that a very large number of control points, say $n \geq 60$, does not improve the quality of the solution significantly but increases the complexity of the model (i.e., the number of parameters needed for the reconstructed curve). This example shows that our method is able to obtain the mathematical expression of a B-spline curve from only the data points with a high level of accuracy.

6.2. *Example 2: parallel curves for tool-path generation in manufacturing*

The generation of CNC (computer numerically controlled) tool-paths to produce a smooth surface is a central issue in any sculptured surface machining process. In this technology, a sequence of cutter-contact points are traced by milling cutters by following a pattern of tracing or scanning usually called tool-path topology [38,43]. Those tool-path patterns can be grouped into four types: serial-pattern, radial-pattern, strip-pattern and contour-pattern [6]. The first two groups, intended for machining an area, consist of trajectories on surfaces that are (either locally or globally) at given distances from a generator. In particular, parallel curves are often used in CNC-machining to ensure that the space between adjacent tool-paths is kept constant in either the three-dimensional space (i.e., on the surface) or in the surface parametric domain [44]. For instance, geodesic parallels have been applied to tool-path generation in zig–zag finishing with 3-axis machining and ball-end cutter so that the scallop-height (the cusp height of the material removed by the cutter) becomes constant [56,58], thus optimizing the size of the cutter location data and, consequently, reducing the machining time. Other examples can be found in [6,44].

Fig. 3 shows a real-world example: a cell phone model scanned by Geomagic [20] and stored as an IGES file freely available online years ago. The model is comprised of six complicated NURBS surfaces with 18×18 control points each. The physical object has been scanned at given distances so as to obtain a collection of 23 sets of data points corresponding to as many different parallel sections. These data sets have been fitted to cubic B-splines by using our IMCH-GAPSO approach. As a result, we obtained a collection of 23 parallel curves for zig–zag machining at fixed distances, displayed in Fig. 3 (left). The number of data points for each curve varied from 676 (for the shortest curve) to 1239, while the number of control points varied from 7 to 12 for the 23 curves. The RMSE between the original points and the reconstructed points is of order 10^{-7} in each coordinate, a clear indication of the excellent fitting of the 3D data points to the B-spline curves in all cases. Computation times varied from 1.18 to 2.79 s for each curve. Fig. 3 (right) shows the B-spline parallel curves projected onto the underlying surface for illustrative purposes.

6.3. *Example 3: section curves*

Cross-sections of surfaces are very important in many applied fields [5]. For example, in the medical area it is possible to reconstruct and display the external surface of the organ under investigation from a set of parallel slices corresponding to different levels. Similarly, in computer design, many objects are usually designed by defining a number of cross-sections; the shape of the surface in-between is computed by using some interpolation scheme. The cross-section of a surface can be obtained through several methods, including ordinary differential equations (ODEs) solved by numerical integration, as in [49]. This yields a set of data points, from which a fitting B-spline curve can be reconstructed. Fig. 4 (left) shows a real paint spray gun comprised of several NURBS surfaces stored in an IGES file. Intersecting this model with a normal plane passing through its center, we obtain a collection of two data sets corresponding to the outer and inner section curves, with a total of 1482 and 623 data points, respectively. We apply our hybrid method to obtain a B-spline representation of such curves. Fig. 4 (right) shows our results for fourth-order B-spline curves with 60 and 25 control points for the outer and inner curves, respectively. In this example, a population of 200 individuals for GA and 150 particles for PSO are used. The RMSE errors are of order 10^{-7} in both cases, with computation times of 12.5 and 5.7 s, respectively. This example shows that our method performs very well for both open curves and closed curves and is able to adapt to complicated shapes with several concavity changes.

6.4. *Example 4: helical curves for tool-path generation in manufacturing*

Helical tool-path topology is widely used in high-speed machining and rapid prototyping [6,44]. In reverse engineering, helical curves appear during the scanning process when the object to be manufactured is rotated so that the vector of light from the scanner to the object maintains a constant angle with respect to a fixed vector (the axis of the helical curve). One example is shown in Fig. 5, where a cylindrical object is rotated along its vertical axis as the beam light moves upwards at a constant velocity. In this example, a collection of 481 data points (on the left) are obtained. Application of our method to this case returns a cubic B-spline curve with 34 control points, shown in Fig. 5 (middle). In this case, we obtain a RMSE between the original and the reconstructed data points of $(0.73, 0.21, 1.34) \times 10^{-5}$ with CPU time of 7.2 s. Fig. 5 (right) shows

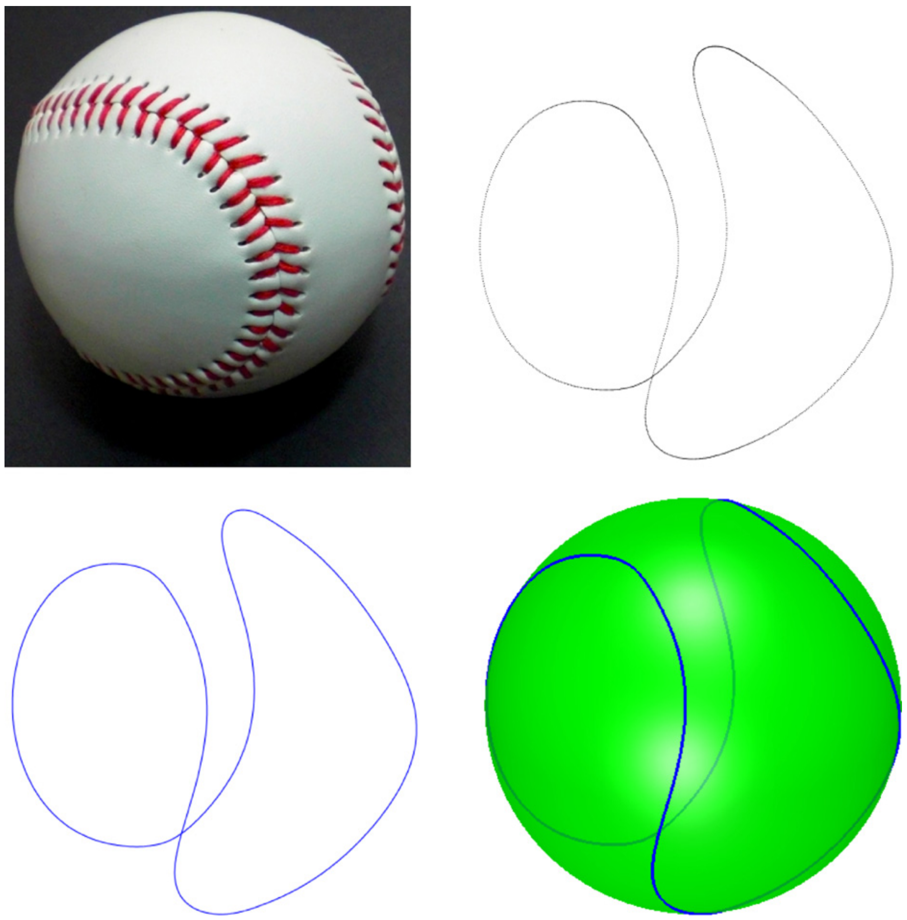
Fig. 2. Baseball seam curve reconstruction: (t-l) original baseball; (t-r) scanned data points of the baseball seam; (b-l) reconstructed B-spline curve; (b-r) reconstructed curve on the baseball surface (partial transparency is applied).

the reconstructed curve on its underlying surface. Once again, partial transparency is applied for better visualization.

Occasionally, real models are so complicated that some of their parts cannot be scanned properly. An example is shown in Fig. 6, where the central part of the piece is comprised of two connected branches whose inner part is not accessible to a laser scanner. In such cases, alternative procedures are needed to generate helical curves. One possibility is to solve an initial-value problem of ordinary differential equations obtained by geometrical arguments from the fact that the helical curve exhibits a constant angle with its axis [50]. In general, the resulting system of ODEs has not analytical solution, so a numerical method (such as Runge–Kutta and the like) to integrate it must be used instead. This yields a sequence of data points on the surface to be manufactured rather than a mathematical curve. This severe limitation can be overcome by applying our method to the set of points obtained by numerical integration in order to get a B-spline representation of such a helical curve. This process is depicted in Fig. 6: the original set of 799 data points (left) is fitted by using a cubic B-spline curve with 56 control points (middle). Fig. 6 (right) highlights the complexity of the helical trajectory: starting at a point on the lower cone, it twists upwardly on one of the branches until it suddenly crosses to the another one through the middle bridge, twists again on that branch and finally moves up on the upper cone until it reaches the top boundary of the surface. Despite this complex oscillatory behavior, we obtain a very good fitting of data points, with RMSE between the original and the reconstructed data points of $(3.26, 4.18, 7.36) \times 10^{-5}$ with CPU time of 17.8 s. In this example, a population of 300 individuals for GA and 200 particles for PSO are used. To our knowledge, no other method provides a B-spline solution to this problem so far with this extremely high level of accuracy.

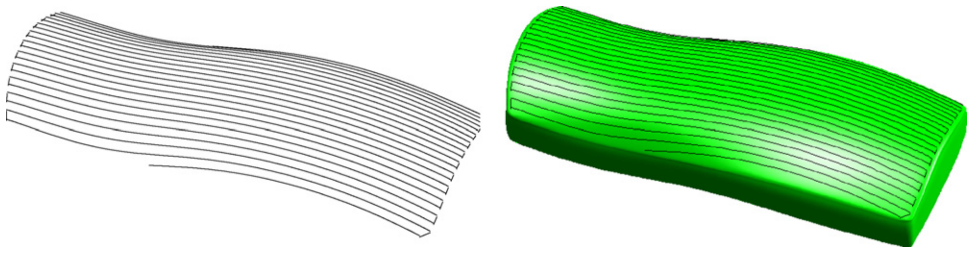
Fig. 3. Parallel curves from data points on a cell phone model for zig–zag tool-path generation: (left) reconstructed curves; (right) curves on the surface.

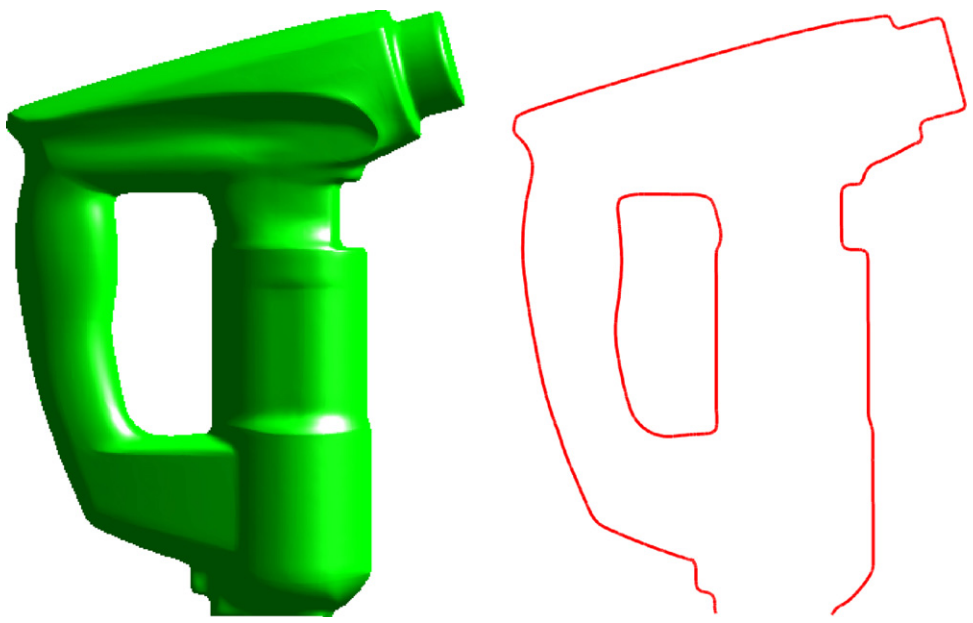

Fig. 4. Section curves from data points of a paint spray gun model: (left) spray gun model; (right) reconstructed inner and outer section curves.

6.5. *Comparison with other approaches*

To the best of our knowledge, no hybrid evolutionary approach has ever been applied to the B-spline curve reconstruction problem discussed in this paper. This lack of previous references prevents us from providing a detailed comparison with other hybrid methods. But even comparison with non-hybrid evolutionary approaches is difficult since, as we already mentioned in Section 2, they have barely been applied to B-spline curve reconstruction so far. Remarkable contributions in the field are the papers in [17,55,62,68–70]. They are, however, extremely limited: with the exception of the work in [70] (that is restricted to closed curves), all these papers deal with explicit functions only, while the parametric case is not addressed. An even more critical drawback is that all are strictly focused on the knot placement problem and, therefore, they require a good parameterization of data points a priori in order to work properly. As a consequence, they fail for the examples discussed in this paper, which are not uniformly sampled. These results are a clear indication of the originality and opportunity of the present contribution.

Although they do not fall exactly within the scope of this paper, we have also considered non-evolutionary methods for B-spline

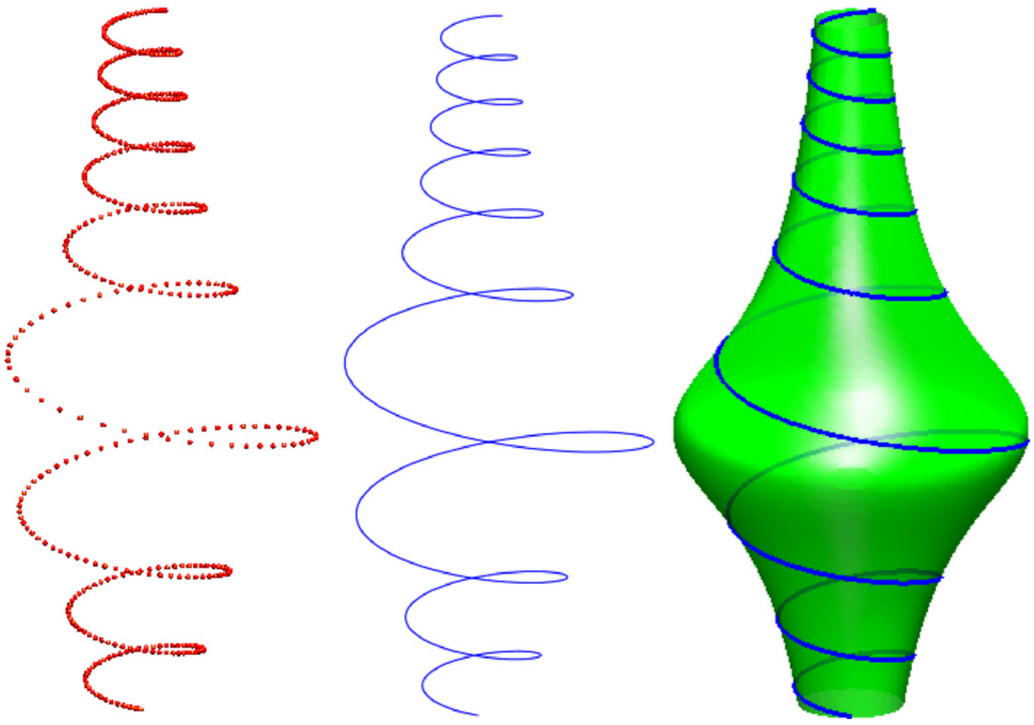

Fig. 5. B-spline curve reconstruction of a helical tool-path trajectory: (left) scanned data points; (middle) reconstructed curve; (right) reconstructed curve on the surface (partial transparency is applied).

Table 1
Comparison of the proposed method with other (non-evolutionary) methods for B-spline curve reconstruction.

Authors, year and reference	Method	Fitting error	# control points	# data points	Free knots	CPU time (s)	Additional comments
		10^{-3}	25				The method requires to give an initial curve close to the shape of the target curve.
Yang et al. (2004) [67]	Iterative optimization based on SDM (square distance minimization)			N.R.	×	2–3	
		10^{-3}	10–50				It also requires pre-calculation of the distance field.
		10^{-3}	13				
		10^{-4}	60				
		10^{-4}	57				
Li et al. (2005) [34]	Adaptive knot placement based on discrete curvature and lowpass filtering	10^{-2}	N.R.	N.R.	√	N.R.	The method is strongly affected by discretization errors. Very simplified version: data parameterization not computed.
		10^{-1}	6	32		1.5	
		10^{-1}	11	102		3	
Wang et al. (2006) [66]	Iterative optimization based on SDM	10^{-1}	8	472	×	3	Very simplified version: knot placement not computed.
		10^{-1}	59	2656		8.1	
Park et al. (2007) [42]	Adaptive refinement using dominant points	10^{-1} to 10^{-3}	50–150	501	×	2–7	Very simplified version: knot placement not computed.
		10^{-3}	110	3765		N.R.	
		10^{-6}	28	1127		3.7	The method carries out all steps of the curve fitting process at full extent.
Our method Gálvez and Iglesias (2012)	Iterative mutually coupled hybrid genetic algorithms–particle swarm optimization (ICMH-GAPSO)	10^{-7}	7–12	676–1239	√	1.2–2.8	No pre-/post-processing required.
		10^{-7}	25	623		5.7	
		10^{-7}	60	1482		12.5	
		10^{-5}	34	481		7.2	
		10^{-5}	56	799		17.8	

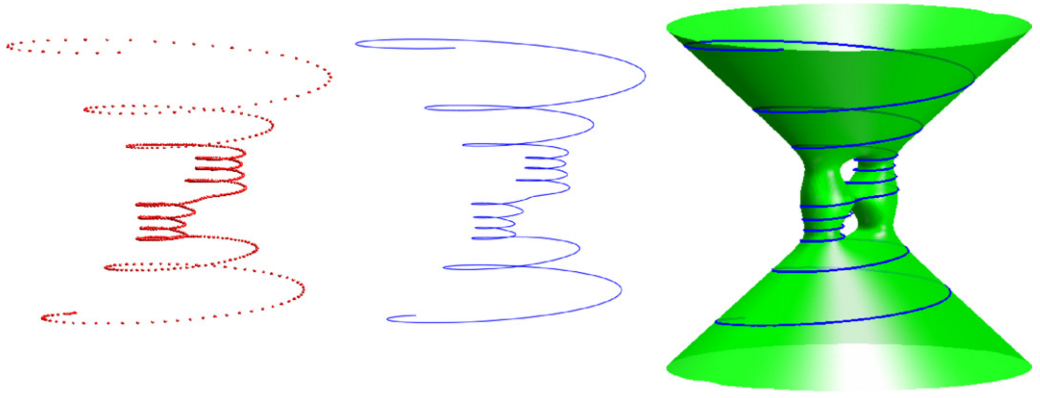

Fig. 6. B-spline curve reconstruction of a helical tool-path trajectory: (left) scanned data points; (middle) reconstructed curve; (right) reconstructed curve on the surface (partial transparency is applied).

curve approximation to enrich our discussion. In particular, we have selected for comparison those methods that meet the two following criteria: (1) they have been published during the last decade and (2) they provide numerical information about fitting errors in their examples, thus allowing quantitative comparison with our approach. This leads to the methods in [34,42,66,67]. Comparative results are summarized in Table 1. The different methods are arranged in rows and sorted in chronological order. For each method, the following items are reported in columns: method employed, fitting errors, number of control points, number of data points, whether or not free knots are allowed (answer *true* is marked with a check ($\surd$); otherwise with symbol $\times$), computation times (in seconds) and some additional comments, used to clarify the CPU times given in the previous column. Wherever any data is missing in the original source, it is indicated by N.R. (*N*ot *R*eported).

The work in [34] reports an adaptive knot placement method based on the discrete curvature of data points. The method is applied to three industrial examples: a section of blade turbine and two examples from reverse engineering applications in the automotive industry, a car hood and a car caudal region section. Fitting mean errors are of order 10^{-2} in all cases. The work in [42] yields tolerance errors of order 10^{-1} to 10^{-3} for an open fitting curve obtained from 501 data points through an adaptive refinement method using dominant points for a number of control points varying in the range 50–150. They also applied their method to a closed curve of 110 control points but, unfortunately, no fitting errors are reported in that case. Finally, the methods in [66,67] use iterative optimization schemes based on the square distance minimization approach. Fitting mean errors in [67] are of order 10^{-3} for three examples, with 25 control points, a number of control points varying in the range 10–50 and 13 control points, respectively, and of order 10^{-4} for two more complicated examples with 60 and 57 control points, respectively. Similarly, the method in [66] reports RMSE values of 10^{-1} for an open curve with 10 control points and two (relatively simple) closed curves with 6 and 11 control points, respectively. For comparison, we obtain RMSE values in the range 10^{-5} to 10^{-7} for more complicated examples, showing that our method outperforms previous approaches in terms of fitting errors.

7. Conclusions and future work

This paper reports the first case of application of hybrid evolutionary systems to B-spline curve reconstruction. In this problem, a massive set of noisy data points obtained from physical objects by technologies such as 3D laser scanning are to be fitted to a B-spline curve. This issue appears ubiquitously in various fields, particularly in reverse engineering for manufacturing. Solving the general problem is very difficult and challenging, because many (qualitatively different) sets of parameters must be calculated simultaneously, leading to a high-dimensional multivariate and multimodal nonlinear optimization problem. Due to their ability to combine the strengths of several meta-heuristics into a single framework, hybrid evolutionary approaches appear to be adequate to deal with the complex, multifaceted nature of this problem. However, existing hybrid approaches are still unsuitable for our problem and, hence, a new hybrid architecture is required. In this context, this paper introduces a new hybrid evolutionary approach called IMCH-GAPSO that combines two classical bio-inspired techniques: genetic algorithms and particle swarm optimization, accounting for data parameterization and knot placement, respectively. The novelty of our approach is that GA and PSO are mutually coupled so that the output of one system is used as the input of the other and vice versa. This coupling, designed to take into account the mutual interplay between data parameterization and knot placement, is repeated iteratively until a termination criterion is attained.

Major advantages of our method can be summarized as follows:

- *Generality*: our method is very general; it can be applied to any set of unconstrained data points. The underlying functions of data can be open or closed curves of any type and degree. For instance, in the examples reported in this paper, the method has been applied to reconstruct parallel curves (example 2), section curves (example 3) and helical curves (example 4).
- *Completeness*: this is one of the most remarkable features of our method. Unlike many previous methods, it addresses all steps of the fitting curve pipeline, namely, data parameterization, knot placement, and least-squares minimization. We do assume neither a given parameterization nor a knot vector. Data parameters and internal knots in our method are truly free variables of the problem. Moreover, the method does not require further pre- or post-processing.
- *Good performance*: to evaluate the performance of our approach, it has been applied to several illustrative examples of data points from real-world applications in manufacturing. Our experimental results show that our approach performs very well, being able to reconstruct with very high accuracy extremely complicated shapes, unfeasible for reconstruction with current methods.
- *Accuracy*: as shown in Table 1, fitting errors of our examples are much lower than those of previous works, even although our examples are more difficult and challenging. In fact, our method outperforms existing B-spline curve reconstruction methods in terms of fitting errors.

Main limitations of our method are the computation times and the parameter tuning. Our method is not very fast; emphasis is on accuracy. It is not significantly slow either. In general, our method compares well with existing methods regarding the computation times, even though most of them do not perform all the steps of the fitting process as does our method. On the other hand, as usual with evolutionary methods, parameter tuning is a problem-dependent issue, meaning that other problems might require a different set of values for the parameters of the method.

Future work includes the development of other coupling strategies for the data parameterization-knot placement interplay, the development of alternative hybrid approaches and the parallelization of this method for optimal performance. We also want to extend our method to rational B-spline curves, where the existence of additional parameters (weights) can modify our procedure of selection of optimal parameters. Application of our hybrid method to other real-world problems is also part of our future work.

Acknowledgments

This research has been supported by the Computer Science National Program of the Spanish Ministry of Education and Science, Project Ref. #TIN2006-13615 and #TIN2012-30768 and the University of Cantabria.

References

[1] D. Allison, R. Diaz, N. Miller, Generalized baseball curves: three symmetries and you're in!, Loci 1 (2009), Freely available from: http://mathdl.maa.org/mathDL/23/?pa=content&sa=viewDocument&nodeId=2866.
[2] P.J. Angeline, Evolutionary optimization versus particle swarm optimization Evolutionary Programming VII. Lecture Notes in Computer Science, vol. 1447, 1998, pp. 601–610.
[3] T. Back, Evolutionary Algorithms in Theory and Practice, Oxford Univ. Press, NY, 1996.
[4] C.A. de Boor, J.R. Rice, Least squares cubic spline approximation. I: fixed knots, CSD TR 20, Purdue University, Lafayette, IN, 1968; C.A. de Boor, J.R. Rice, Least squares cubic spline approximation. II: variable knots, CSD TR 21, Purdue University, Lafayette, IN, 1968.
[5] E. Castillo, A. Iglesias, Some characterizations of families of surfaces using functional equations., ACM Transactions on Graphics 16 (3) (1997) 296–318.
[6] B.K. Choi, R.B. Jerard, Sculptured Surface Machining. Theory and Applications, Kluwer Academic Publishers, Dordrecht/Boston/London, 1998.
[7] M. Clerc, J. Kennedy, The particle swarm: explosion stability and convergence in a multi-dimensional complex space, IEEE Transactions on Evolutionary Computation 6 (1) (2002) 58–73.
[8] M. Crampin, R. Guifo, G.A. Read, Linear approximation of curves with bounded curvature and a data reduction algorithm, Computer-Aided Design 17 (6) (1985) 257–261.
[9] P. Dierckx, Curve and Surface Fitting with Splines, Oxford University Press, Oxford, 1993.
[11] R.C. Eberhart, Y. Shi, Comparison between genetic algorithms and particle swarm optimization Evolutionary Programming VII. Lecture Notes in Computer Science, vol. 1447, 1998, pp. 611–616.
[12] R.C. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Proc. CEC, San Diego, CA, 2000, pp. 84–88.
[13] R.C. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in: Proceedings of the 2001 Congress on Evolutionary Computation, 2001, pp. 81–86.
[14] A.P. Engelbretch, Fundamentals of Computational Swarm Intelligence, John Wiley and Sons, Chichester, England, 2005.
[15] A.A. Esmin, G. Lambert-Torres, G.B. Alvarenga, Hybrid evolutionary algorithm based on PSO and GA mutation, in: Proceedings of 6th International Conference on Hybrid Intelligent Systems, 2006, pp. 57–62.
[16] G. Farin, Curves and Surfaces for CAGD, 5th ed., Morgan Kaufmann, San Francisco, 2002.
[17] A. Gálvez, A. Iglesias, Efficient particle swarm optimization approach for data fitting with free knot B-splines, Computer-Aided Design 43 (12) (2011) 1683–1692.
[18] A. Gálvez, A. Iglesias, Iterative two-step genetic-algorithm-based method for efficient polynomial B-spline surface reconstruction, Information Sciences 182 (1) (2012) 56–76.
[19] A. Gálvez, A. Iglesias, Particle swarm optimization for non-uniform rational B-spline surface reconstruction from clouds of 3D data points, Information Sciences 192 (1) (2012) 174–192.
[20] Geomagic web site: http://www.geomagic.com.
[22] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
[23] E.A. Grimaldi, F. Grimacia, M. Mussetta, P. Pirinoli, R.E. Zich, A new hybrid genetical—swarm algorithm for electromagnetic optimization., in: Proceedings of International Conference on Computational Electromagnetics and its Applications, 2004, pp. 157–160.
[24] J.H. Holland, Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor, 1975.
[25] J. Hoschek, D. Lasser, Fundamentals of Computer Aided Geometric Design, A.K. Peters, Wellesley, MA, 1993.
[26] A. Isheila, J.P. Gonneta, D. Joannica, J.F. Fontaine, Systematic error correction of a 3D laser scanning measurement device, Optics and Lasers in Engineering 49 (1) (2011) 16–24.
[27] S. Jeong, S. Hasegawa, K. Shimoyama, S. Obayashi, Development and investigation of efficient GA/PSO-hybrid algorithm applicable to real-world design optimization, IEEE Computational Intelligence Magazine 4 (3) (2009) 36–44.
[28] L. Jing, L. Sun, Fitting B-spline curves by least squares support vector machines, in: Proc. of the 2nd. Int. Conf. on Neural Networks & Brain, IEEE Press, Beijing, China, 2005, pp. 905–909.
[29] C.F. Juang, A hybrid of genetic algorithm and particle swarm optimization for recurrent network design, IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics 34 (2004) 997–1006.
[30] D.L.B. Jupp, Approximation to data by splines with free knots, SIAM Journal of Numerical Analysis 15 (1978) 328–343.
[31] Y.T. Kao, E.E. Zahara, A hybrid genetic algorithm and particle swarm optimization for multimodal functions, Applied Soft Computing 8 (2008) 849–857.
[32] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, Perth, Australia, 1995, pp. 1942–1948.
[33] J. Kennedy, R.C. Eberhart, Y. Shi, Swarm Intelligence, Morgan Kaufmann Publishers, San Francisco, CA, 2001.
[34] W. Li, S. Xu, G. Zhao, L.P. Goh, Adaptive knot placement in B-spline curve approximation, Computer-Aided Design 37 (2005) 791–797.
[35] F. López-López, Question #48. Is there a physical property that determines the curve that defines the seam of a baseball? American Journal of Physics 64 (9) (1996) 1097.
[36] T. Lyche, K. Morken, Knot removal for parametric B-spline curves and surfaces, Computer-Aided Design 4 (1987) 217–230.
[37] T. Lyche, K. Morken, A data-reduction strategy for splines with applications to the approximation of functions and data, IMA Journal of Numerical Analysis 8 (1988) 185–208.
[38] S. Marshall, J.G. Griffiths, A new cutter-path topology for milling machines, Computer Aided Design 26 (3) (1994) 204–214.
[39] M. Mitchell, An Introduction to Genetic Algorithms (Complex Adaptive Systems), MIT Press, 1998.
[40] A. Mohammadi, M. Jazaeri, A hybrid particle swarm optimization-genetic algorithm for optimal location of SVC devices in power system planning, in: Proceedings of 42nd International Universities Power Engineering Conference, 2007, pp. 1175–1181.
[41] H. Park, An error-bounded approximate method for representing planar curves in B-splines, Computer Aided Geometric Design 21 (2004) 479–497.
[42] H. Park, J.H. Lee, B-spline curve fitting based on adaptive curve refinement using dominant points, Computer-Aided Design 39 (2007) 439–451.
[43] S.C. Park, M. Chang, Tool path generation for a surface model with defects, Computers in Industry 61 (1) (2010) 75–82.
[44] N.M. Patrikalakis, T. Maekawa, Shape Interrogation for Computer Aided Design and Manufacturing, Springer Verlag, Heidelberg, 2002.
[45] L. Piegl, W. Tiller, The NURBS Book, Springer Verlag, Berlin Heidelberg, 1997.
[46] R. Poli, J. Kennedy, T. Blackwell, Particle swarm optimization—an overview, Swarm Intelligence 1 (2007) 33–57.
[47] H. Pottmann, S. Leopoldseder, M. Hofer, T. Steiner, W. Wang, Industrial geometry: recent advances and applications in CAD, Computer Aided Design 37 (2005) 751–766.
[48] M.J.D. Powell, Curve fitting by splines in one variable, in: J.G. Hayes (Ed.), Numerical Approximation to Functions and Data, Athlone Press, London, 1970.
[49] J. Puig-Pey, A. Gálvez, A. Iglesias, A new differential approach for parametric-implicit surface intersection, Lectures Notes in Computer Science 2657 (2003) 897–906.
[50] J. Puig-Pey, A. Gálvez, A. Iglesias, Helical curves on surfaces for computer-aided geometric design and manufacturing, Lectures Notes in Computer Science 3044 (2004) 771–778.
[51] A. Razdan, Knot Placement for B-Spline Curve Approximation, Arizona State University, Tempe, AZ, 1999.
[52] J.R. Rice, The Approximation of Functions, Vol. 2, Addison-Wesley, Reading, MA, 1969.
[53] J. Robinson, S. Sinton, Y.R. Samii, Particle swarm, genetic algorithm, and their hybrids: optimization of a profiled corrugated horn antenna., in: Proceedings of the IEEE International Symposium in Antennas and Propagation Society, 2002, pp. 314–317.
[54] N. Ru, Y. Jianhua, A GA and particle swarm optimization based hybrid algorithm., in: Proceedings of the IEEE Congress on Evolutionary Computation, 2008, pp. 1047–1050.
[55] M. Sarfraz, S.A. Raza, Capturing outline of fonts using genetic algorithms and splines, in: Proc. of Fifth International Conference on Information Visualization IV'2001, IEEE Computer Society Press, 2001, pp. 738–743.
[56] R. Sarma, D. Dutta, The geometry and generation of NC tool paths, Journal of Mechanical Design: ASME Transactions 119 (1997) 253–258.

[57] M. Settles, T. Soule, Breeding swarms: a GA/PSO hybrid., in: Proceedings of Genetic and Evolutionary Computation Conference, GECCO'05, ACM Press, 2005, pp. 161–168.
[58] K. Suresh, D.C.H. Yang, Constant scallop-height machining of free-form surfaces, Journal of Engineering for Industry: ASME Transactions 116 (1996) 253–259.
[59] R. Thangaraj, M. Pant, A. Abraham, P. Bouvry, Particle swarm optimization: hybridization perspectives and experimental illustrations, Applied Mathematics and Computation 217 (12) (2011) 5208–5226.
[60] R. Thompson, Designing a baseball cover, College Mathematics Journal 29 (1998) 48–61.
[61] I.C. Trelea, The particle swarm optimization algorithm: convergence analysis and parameter selection, Information Processing Letters 85 (2003) 317–325.
[62] E. Ulker, A. Arslan, Automatic knot adjustment using an artificial immune system for B-spline curve approximation, Information Sciences 179 (2009) 1483–1494.
[63] F. Valdez, P. Melin, O. Castillo, Evolutionary method combining particle swarm optimization and genetic algorithms using fuzzy logic for decision making, in: Proceedings of the IEEE International Conference on Fuzzy Systems, 2009, pp. 2114–2119.
[64] T. Varady, R.R. Martin, J. Cox, Reverse engineering of geometric models—an introduction, Computer Aided Design 29 (4) (1997) 255–268.
[65] T. Varady, R.R. Martin, Reverse Engineering, in: G. Farin, J. Hoschek, M. Kim (Eds.), Handbook of Computer Aided Geometric Design, Elsevier Science, 2002.
[66] W.P. Wang, H. Pottmann, Y. Liu, Fitting B-spline curves to point clouds by curvature-based squared distance minimization, ACM Transactions on Graphics 25 (2) (2006) 214–238.
[67] H.P. Yang, W.P. Wang, J.G. Sun, Control point adjustment for B-spline curve approximation, Computer-Aided Design 36 (2004) 639–652.
[68] F. Yoshimoto, M. Moriyama, T. Harada, Automatic knot placement by a genetic algorithm for data fitting with a spline., in: Proc. of Shape Modeling International'99, IEEE Computer Society Press, 1999, pp. 162–169.
[69] F. Yoshimoto, T. Harada, Y. Yoshimoto, Data fitting with a spline using a real-coded algorithm, Computer Aided Design 35 (2003) 751–760.
[70] X. Zhao, C. Zhang, B. Yang, P. Li, Adaptive knot placement using a GMM-based continuous optimization algorithm in B-spline curve approximation, Computer-Aided Design 43 (2011) 598–604.