Applying Factored Evolutionary Algorithms to the B-Spline Knot Selection Problem

Eve Ginsberg
University of Virginia
Charlottesville, VA, USA
adk6dr@viriginia.edu

Jordan Schupbach
Montana State University
Bozeman, MT, USA
jordan.schupbach@montana.edu

John Sheppard
Montana State University
Bozeman, MT, USA
john.sheppard@montana.edu

Nicholas Turk
Georgia Institute of Technology
Atlanta, GA, USA
nturk8@gatech.edu

***Abstract*—Stochastic search algorithms are an effective method for solving the B-spline knot selection problem. Unfortunately, they often require a large number of (in this case, expensive) fitness function evaluations to find good solutions. In an attempt to mitigate this issue, we apply a modern cooperative coevolutionary algorithm – Factored Evolutionary Algorithms (FEAs) – to the B-spline knot selection problem. We demonstrate FEA's performance on a variety of benchmark functions, and we compare FEA to traditional stochastic search methods. We also propose an FEA-specific method to evaluate Mean Squared Error (MSE) more efficiently, thus reducing the theoretical runtime.**

I. Introduction

Fitting a curve to data (i.e. nonlinear regression) is an important and foundational problem for many fields, as we often obtain data which follow some unknown functional form. B-spline basis functions are a popular choice to form a basis for approximating unknown functions due to their efficiency (with respect to number of parameters) and ability to adapt to different functional forms. Adaptation is achieved by the selection of knot points (referred to as the knot-selection problem), which determine the underlying function space allowed to fit the data and hence the quality of fit for a particular dataset/knot-vector combination.

A. B-spline Regression

B-spline basis functions are piecewise polynomial functions of a given degree d defined by a nondecreasing sequence called a knot vector: $\mathbf{k} = (\mathbf{k}_0, \mathbf{k}_1, ..., \mathbf{k}_p)$ [1]. B-spline basis functions can then be defined recursively through the Cox-de Boor formulation [2]. The base case is:

$$B_{i,0}(x) = \begin{cases} 1 & \text{if } \mathbf{k}_i \le x < \mathbf{k}_{i+1} \\ 0 & \text{otherwise} \end{cases}.$$

The general case for the i-th basis function of degree d is:

$$B_{i,d}(x) = \frac{x - \mathbf{k}_i}{\mathbf{k}_{i+d} - \mathbf{k}_i} B_{i,d-1}(x) + \frac{\mathbf{k}_{i+d+1} - x}{\mathbf{k}_{i+d+1} - \mathbf{k}_{i+1}} B_{i+1,d-1}(x).$$

A B-spline curve C of order $d+1$ (degree d) with p knot points is then defined as a linear combination of the basis functions:

$$C(x) = \sum_{i=1}^{p} \theta_i B_{i,d}(x). \tag{1}$$

The functional form of the B-spline basis functions and curves gives properties that $C(x)$ is a piecewise polynomial function of degree d over each interval $(\mathbf{k}_i, \mathbf{k}_{i+1})$, is continuous at the knot locations $\mathbf{k}$ (for $d > 0$) and has $d - 2$ continuous derivatives.

B-spline regression splines use this functional form to fit a curve to data $\mathcal{D} = \{\mathbf{x}_i, \mathbf{y}_i\}_{i=1}^n$ according to the model

$$\mathbf{y}_i = f(\mathbf{x}_i) + \boldsymbol{\epsilon}_i = \sum_{i=1}^{p} \boldsymbol{\theta}_i B_{i,d}(x_i) + \boldsymbol{\epsilon}_i.$$

The parameters of this model is estimated by minimizing MSE

$$\hat{\boldsymbol{\theta}} = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} (\hat{\mathbf{y}}_i - \mathbf{y}_i)^2, \tag{2}$$

whose solution is obtained by solving the normal equations, resulting in

$$\hat{\boldsymbol{\theta}} = (\mathbf{B}^\top \mathbf{B})^{-1} \mathbf{B}^\top \mathbf{y} \tag{3}$$

where $\mathbf{B}_{i,j} = B_{j,d}(x_i)$. This is a linear optimization problem, which for B-spline regression, the design matrix $\mathbf{B}$ is potentially very sparse (depending on the number and order of B-spline bases) enabling the system to be solved efficiently with sparse linear solvers.

B. Knot Selection

In the B-spline knot selection problem, we find the model coefficients $\hat{\boldsymbol{\theta}}$ **and** the knot vector $\mathbf{k}^{opt}$ that minimize MSE:

$$\mathbf{k}^{opt}, \hat{\boldsymbol{\theta}} = \underset{\mathbf{k},\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2. \tag{4}$$

Note that it is possible for other criteria to be used to determine the optimal knot vector (e.g. AIC, BIC, GCV, C_p). Also note that because $\mathbf{k}$ is nondecreasing, the solution space is constrained to the p-simplex:

$$S_p[a, b] = \{\mathbf{k} \in \mathbb{R}^d : a = \mathbf{k}_1 < \mathbf{k}_2 < \cdots < \mathbf{k}_p = b\}.$$

Allowing the knot-vector to vary, turns a simple linear optimization problem (Eqn 2) into a nonlinear optimization problem (Eqn 4), called the full-functional problem [3], with a potentially large number of stationary points [4]. In particular, Jupp [5] showed that the "lethargy" property is intrinsic to the knot selection problem, which has the consequence that the objective function has many stationary points, is non-convex, and has poor convergence for gradient based algorithms when solutions are near the boundary, which is often the case. Further, Zhou and Shen [6] described the knot confounding problem, which arises from the interdependence of knots on the resulting quality of fit. This problem manifests as a problem for step-wise insertion/removal algorithms where knots cannot be added, removed, or moved without modifying nearby knots or severely degrading fit.

This interdependence may also manifest as the "two steps forward, one step back problem" in stochastic search algorithms [7], where some dimensions move closer to the optimal solution while others move away, contributing to slower convergence rates and potentially suboptimal solutions. Cooperative PSO (CPSO) was proposed as a way to mitigate this phenomenon [7]. For this reason, cooperative evolutionary algorithms (CCEA) are a natural choice for solving the knot selection problem. In this work, we apply a Factored Evolutionary Algorithm (FEA) [8], which is a generalized class of CCEA, which include methods like CPSO [7] and CCGA [9].

II. Literature Review

Determining how best to solve the knot selection problem is a well-studied field that has put forth many potential methods. Each method has some way of determining both the optimal number of knots and the optimal placement of those knots. Early methods date back to the late 1960s and early 1970s. For example, de Boor and Rice proposed the `SWEEP` and `OPT` algorithm [10], which alternate between adding knots and then repositioning one at a time.

Various step-wise forward and backward selection procedures have also been proposed that use a variety of selection criteria (e.g. MSE, AIC, BIC, GCV, C_p) to guide the insertion and removal of knots. This includes work on `TURBO` [11], `MARS` [12], `LOGSPLINE`, `POLYMARS` [13], and the insertion/removal/relocation algorithm of Zhou and Shen [6].

There are also various methods based on geometric heuristics. Park and Li [14] proposed a two-stage approach where in the first stage, "dominant points" are selected and in the second knot positions are optimized to balance inter-segment shape index distance. Both Li et al. [15] and Michel and Zidna [16] use heuristics based on estimates of discrete local curvature for the addition of knots. Razdan uses arc length and curvature estimates to select points of interest [17]. Aguilar et al. also rely on estimates of curvature for a knot readjustment scheme [18]. Some methods place knots where estimates of the fourth derivative are largest (e.g., [19]). Yeh et al. [20] also make use of derivatives as a heuristic for knot selection.

There is a growing body of work that use and adapt stochastic search methods. Miyata and Shen [21] and Pittman [22] used Genetic Algorithms (GA). Iglesias and Galvez [23] and Mohanty and Fahnestock [24] use Particle Swarm Optimization (PSO). Luo et al. [25] used Differential Evolution (DE). Galvez and Iglesias used the Firefly algorithm [26] and Galvez et al. used the elitist clonal selection algorithm [27]. Finally, Miyata and Shen [21], [28] used evolutionary algorithms with simulated annealing. Stochastic search algorithms often have good theoretical guarantees for finding an optimal solution and work well in practice, but they can be expensive computationally. Our work most closely relates to this body of research by using factored evolutionary algorithms to solve the knot selection problem. By using FEA, we investigate whether CCEAs can be more efficient than traditional stochastic search algorithms at solving the knot selection problem.

III. Methods

We explore three traditional stochastic search algorithms as base algorithms to FEA: PSO, DE, and GA. These algorithms are well known stochastic search algorithms that have in prior research been successfully applied to a wide variety of optimization problems, including the knot selection problem [29], [23], [25]. The novelty of our method is in the use of FEA wrapped around these three methods.

A. Factored Evolutionary Algorithms

FEA is a relatively new class of stochastic search algorithms that subdivides the problem into overlapping "factors" corresponding to subsets of the variables optimized [8]. Each factor is given its own optimization routine (often population based), and the factors compete with and share information with one another to iterate towards an optimal solution by maintaining a global solution $\mathbf{G}$ also referred to as the context vector.

Given a parameter set $\mathbf{k} = \{\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_d\}$ with index set $\mathcal{I}^d = \{1, \ldots, d\}$, the factor architecture $\boldsymbol{\mathcal{F}}$ is a collection of subsets of $\mathbf{k}$ such that $\boldsymbol{\mathcal{F}}_i \subset \mathbf{k}$ for each $\boldsymbol{\mathcal{F}}_i \in \boldsymbol{\mathcal{F}}$, and $\cup_{i=1}^{|\boldsymbol{\mathcal{F}}|} \boldsymbol{\mathcal{F}}_i = \mathbf{k}$. Associated with each factor is an index set $\boldsymbol{\mathcal{I}}_i = \{a \in \mathcal{I}^d : \mathbf{k}_a \in \boldsymbol{\mathcal{F}}_i\}$ and an optimization algorithm $\boldsymbol{\mathcal{S}}_i$, also referred to as a subpopulation, that searches the parameter space defined by the factor $\boldsymbol{\mathcal{F}}_i$. In the other direction, associated with each variable $\mathbf{k}_j$ is the set of factors containing that variable $\boldsymbol{\mathcal{O}}_j = \{\boldsymbol{\mathcal{F}}_k \in \boldsymbol{\mathcal{F}} : \mathbf{k}_j \in \boldsymbol{\mathcal{F}}_k\}$ called the "overlap." Additionally, there are the subpopulations associated with the factors in the overlap set, referred to as "overlapping subpopulations" and denoted $(\boldsymbol{\mathcal{OS}})_j = \{\boldsymbol{\mathcal{S}}_k \in \boldsymbol{\mathcal{S}} : \mathbf{k}_j \in \boldsymbol{\mathcal{F}}_k\}$. After initializing the global context $\mathbf{G}$ and each subpopulation $\boldsymbol{\mathcal{S}}_i$, the FEA algorithm repeats three steps until convergence: **update**, **compete**, and **share**.

During the **update** step (the for loop in Alg. 1), each subpopulation $\boldsymbol{\mathcal{S}}_i$ is optimized relative to the objective function $f(\mathbf{k})$ allowing only the variables associated to its factor $\boldsymbol{\mathcal{F}}_i$ to vary while the remaining variables $\mathbf{r}_i = \boldsymbol{\mathcal{F}}_i \setminus \mathbf{k}$ are held to the values given by the global context $\mathbf{G}$. During the **compete** step (Alg. 2), variables in $\mathbf{k}$ are iterated through in a permuted order $\mathbf{p} = perm(\mathcal{I})$. The subpopulations associated to the factors in overlap set $\boldsymbol{\mathcal{O}}_{\mathbf{p}_j}$, namely $(\boldsymbol{\mathcal{OS}})_{\mathbf{p}_j}$ are considered for updating variable $\mathbf{k}_{\mathbf{p}_j}$ in $\mathbf{G}$. The value of $\mathbf{k}_{\mathbf{p}_j}$ in $\mathbf{G}$ is replaced by the $\mathbf{k}_{\mathbf{p}_j}$ from the best solution found among the subpopulations in $(\boldsymbol{\mathcal{OS}})_{\mathbf{p}_j}$, so long as that solution has higher fitness than $\mathbf{G}$. Finally, during the **share** step (Alg. 3), each subpopulation $\boldsymbol{\mathcal{S}}_i$ is updated so the nonfree variables $\mathbf{r}_i$ coincide with $\mathbf{G}$.

It is natural to consider "linear" factor architectures with some amount of overlap for the B-spline knot selection problem. For example, a set of variables $\mathbf{k} = \{\mathbf{k}_1, \ldots, \mathbf{k}_d\}$ may be broken into a linear factor architecture

Algorithm 1 FEA

```
Input: Objective function $f$, optimization algorithm $\mathcal{A}$, factor architecture $\mathcal{F}$, domain $\mathcal{D}$
Output: Global context solution $\mathbf{G}$
  $\mathbf{G} \leftarrow$ initializeGlobal($\mathcal{D}$)
  Sort($\mathbf{G}$)
  $\boldsymbol{\mathcal{S}} \leftarrow$ initializeSubpops($f, \mathcal{F}, \mathcal{A}$)
  repeat
    for all $\boldsymbol{\mathcal{S}}_i \in \boldsymbol{\mathcal{S}}$ do
      repeat
        $\boldsymbol{\mathcal{S}}_i$.updateIndividuals()
      until Termination criterion is met
    end for
    $\mathbf{G} \leftarrow$ Compete($f, \boldsymbol{\mathcal{S}}$)
    Share($\mathbf{G}, \boldsymbol{\mathcal{S}}$)
  until Termination criterion is met
  return $\mathbf{G}$
```

$\boldsymbol{\mathcal{F}} = \{\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3\}, \{\mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}, \ldots, \{\mathbf{k}_{d-2}, \mathbf{k}_{d-1}, \mathbf{k}_d\}\}$ with a factor size of 3 and maximum overlap size of 3. This architecture may mitigate the knot confounding and "two steps forward, one step back" problems previously described as it allows a subsequence of knots to be changed simultaneously within the subpopulations. In addition, it enables performance improvements by allowing "partial" fitness function evaluations. Because we know the knots not associated with a given factor do not change, we only need to partially update $\mathbf{B}$ when evaluating Eqn. 2 and we can use the coefficients from the previous iteration as a starting point for iterative solvers.

IV. Experiments

We conducted experiments to compare the performance of FEA to traditional stochastic search algorithms on a variety of benchmark functions. We did this by running the B-spline knot selection problem on each base algorithm (PSO, DE, and GA) and the FEA version of each. We used the Mean Squared Error (MSE) of the B-spline approximation as our metric for these comparisons and considered six benchmark functions, two levels of sample size and three levels of noise.

Five of the six benchmark functions were taken from literature ([29] and [30]) illustrated in Figure 1. In particular, we chose the "Blocks", "Bumps", "HeaviSine" and "Big Spike" functions exactly as described in these papers. We also modified the doppler function; instead using $f(t) = \sin\left(\frac{20}{t+0.3}\right)$. For the sixth benchmark, We created our own functions by producing randomly generated B-spline curves. We generated our random functions using either a Beta or Uniform distribution of 30, 40, and 50 knot points and a normal distribution of $\boldsymbol{\theta}$ to produce the random B-spline curves, (i.e. $\mathbf{k} \sim \textsc{Beta}(a, b)$ *or* $\mathbf{k} \sim \textsc{Uniform}(0, 1)$ and $\boldsymbol{\theta} \sim \textsc{Normal}(\mu, \sigma^2)$ with $\mathbf{y}_i(x) = \sum_{i=1}^{p} \theta_i \beta_{i,d}(x) + \boldsymbol{\epsilon}_i)$ with parameters set to $a = 1$, $b = 3$, $\mu = 0$, and $\sigma^2 = 1$.

For each benchmark function, we sampled points uniformly along the function (i.e. $x \sim \textsc{Uniform}(0, 1)$, adding normally distributed noise to those points (i.e. $y = f(x) + \epsilon\ ,\quad \epsilon \sim N(0, \sigma^2)$), using sample sizes of $n = 2000$ and 5000. The noise was sampled from a normal distribution with $\mu = 0$, and a σ^2 that depended on the range of the given benchmark function. For a function with range $r = range(f)$, we used standard deviations of $\sigma = \frac{r}{20}$, $\frac{r}{12.5}$, and $\frac{r}{5}$. Finally, we allowed each algorithm to run for 150,000 fitness function evaluations.

We gave each of the traditional algorithms a population size of 1000 and tuned the base algorithms' other parameters, including the number of knots, through Bayesian optimization [31]. By using Bayesian optimization to tune the number of knots, we avoided some of the issues with incremental addition and removal of knot points. FEA had two more parameters to tune, namely the factor size and overlap for the factor architecture, which were also tuned via Bayesian optimization. The additional number of parameters to tune in FEA means that tuning the algorithm is more difficult than tuning the base stochastic search algorithms alone. That said, we allowed FEA's tuning to run for the same amount of iterations as the

Algorithm 2 FEA Compete

```
Input: Objective function $f$, optimization algorithm $\mathcal{A}$, factor architecture $\mathcal{F}$, domain $\mathcal{D}$, context $\mathbf{G}$, subpopulations $\boldsymbol{\mathcal{S}}$
Output: Updated context $\mathbf{G}$
  $\mathbf{p} \leftarrow perm(\mathcal{I}^d)$
  for j = 1 to $d$ do
    $(\boldsymbol{\mathcal{OS}})_{\mathbf{p}_j} \leftarrow \{\boldsymbol{\mathcal{S}}_k \in \boldsymbol{\mathcal{S}} : \mathbf{x}_j \in \boldsymbol{\mathcal{F}}_k\}$
    bestVal $\leftarrow \mathbf{G}[\mathbf{p}_j]$
    GCopy $\leftarrow$ G
    sort(GCopy)
    bestFit $\leftarrow f$(GCopy)
    $\mathbf{q} \leftarrow perm(\mathcal{I}^{|(\boldsymbol{\mathcal{OS}})|})$
    for $i = 1$ to $|(\boldsymbol{\mathcal{OS}})|$ do
      $currVal \leftarrow (\boldsymbol{\mathcal{OS}})_{\mathbf{p}_j}[\mathbf{q}_i]$.getBestSolution()$[\mathbf{p}_j]$
      $\mathbf{G}[\mathbf{p}_j] \leftarrow currVal$
      GCopy $\leftarrow$ G
      sort(GCopy)
      if $f$(GCopy) $< bestFit$ then
        $bestVal \leftarrow currVal$
        $bestFit \leftarrow f$(GCopy)
      end if
    end for
    $\mathbf{G}[\mathbf{p}_j] \leftarrow bestVal$
  end for
  sort($\mathbf{G}$)
  return $\mathbf{G}$
```

Algorithm 3 FEA Share

```
Input: Objective function $f$, context $\mathbf{G}$, subpopulations $\boldsymbol{\mathcal{S}}$
Output: Updated subpopulations $\boldsymbol{\mathcal{S}}$
  for all $\boldsymbol{\mathcal{S}}_i \in \boldsymbol{\mathcal{S}}$ do
    $\boldsymbol{\mathcal{S}}_i$.contextVector $\leftarrow$ G
    $\boldsymbol{\mathcal{S}}_i$.updateDomain()
    $\boldsymbol{\mathcal{S}}_i$.updateFitness()
  end for
  return
```

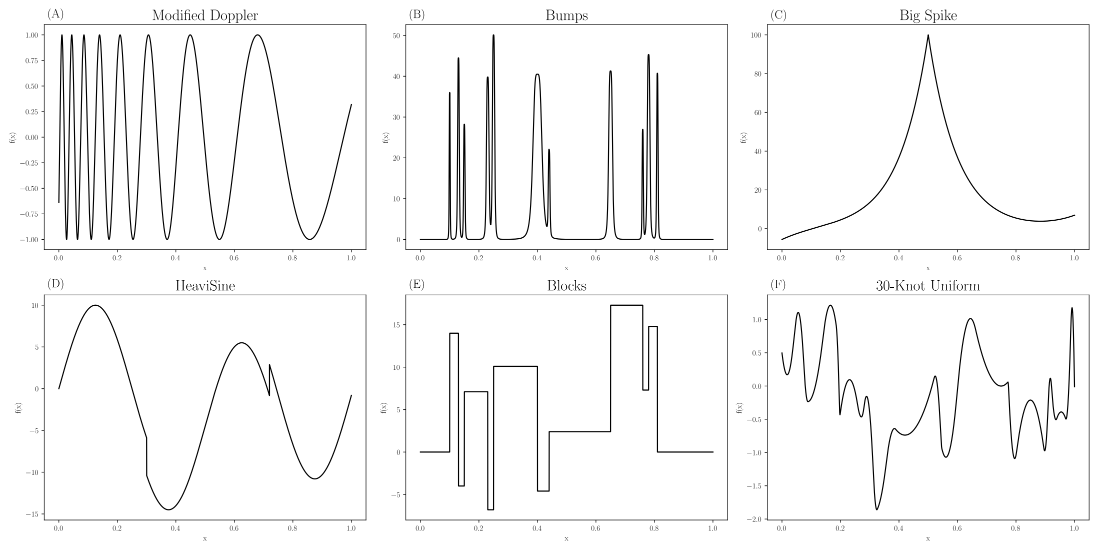

Fig. 1. Benchmark functions used in experiments. Bottom-right (F) is the random benchmark function with 30-knots, drawn from a uniform distribution.

TABLE I
EXPERIMENTAL RESULTS FOR BENCHMARK FUNCTIONS WITH $n = 5000$, $\sigma = \frac{r}{5}$

Function	DE	GA	PSO	DE FEA	GA FEA	PSO FEA
BigSpike	27.9663	27.0731	**26.5264**	27.5446	26.8739	26.6234
Blocks	3.0990	1.6143	1.5496	**1.4277**	1.4696	1.5751
Bumps	19.5134	6.6819	6.8302	**6.2533**	6.2911	8.0621
HeaviSine	1.4825	1.5047	1.4710	**1.4393**	1.4933	1.4869
MDoppler	0.0130	0.0102	**0.0097**	0.0100	0.0107	0.0108
Unif30	0.0470	0.0348	0.2384	0.0348	0.0344	**0.0339**
Unif40	0.0417	0.0506	0.0330	**0.0310**	0.0311	0.0321
Unif50	0.0738	0.1397	0.0556	0.0486	0.0490	**0.0481**
Beta30	0.0426	0.0482	**0.0341**	**0.0341**	0.0344	0.0345
Beta40	0.0901	0.1267	0.0699	0.0515	**0.0502**	0.0534
Beta50	0.0860	0.1170	0.0600	**0.0459**	0.0463	0.0545

base stochastic search algorithms.

V. RESULTS

Results of our experiments comparing FEA to the base algorithms are provided in Table I with the best for each function in bold. We only present the results for the large sample size ($n = 5000$) and large noise level ($\sigma = \frac{r}{5}$) as we saw similar patterns across all sample size and noise levels. The results show FEA outperforms or is competitive with the best performing algorithm on all of the benchmark functions. Additionally, the DE and GA variants of FEA outperformed the PSO variant. It is interesting to note that DE performed worst on the benchmark functions while the FEA variant of DE usually performed best. Alternatively, the base PSO algorithm was fairly competitive with all of the other algorithms while the FEA variant appears to have degraded its performance several cases. Interestingly, as the dimensionality of the problem increased in the random function benchmarks, FEA algorithms appeared to perform better compared with the base algorithms. The experiment scripts [1], a python FEA implementation [2] and modifications to FEA for the B-spline knot selection problem [3] are publicly available.

VI. DISCUSSION

The B-spline knot selection problem is a difficult optimization problem due to the large number of local optima in the objective function and the large search space. Traditional stochastic search algorithms have been used to solve this problem effectively, but they are often slow and can get stuck in local optima. Our experiments show that FEA is competitive with or outperforms baseline stochastic search algorithms on the B-spline knot selection problem for the benchmark functions we considered. This was achieved with each algorithm being given the same fitness function evaluation budget. We have also shown that FEA can be modified to allow for partial fitness function evaluations, which can lead to potential performance gains. We believe this is a promising approach to the B-spline knot selection problem and that further research in this area is warranted, to investigate how or when CCEAs such as FEA may perform better than the base algorithms.

[1]https://github.com/jordanschupbach/feareuExperiments

[2]https://github.com/jordanschupbach/pyFEA

[3]https://github.com/jordanschupbach/pyBsplineFEA

References

[1] C. De Boor, *A Practical Guide to Splines*. Springer-Verlag, 1978.

[2] ——, "On calculating with b-splines," *Journal of Approximation Theory*, vol. 6, no. 1, pp. 50–62, 1972.

[3] G. H. Golub and V. Pereyra, "The differentiation of pseudo-inverses and nonlinear least squares problems whose variables separate," *SIAM Journal on Numerical Analysis*, vol. 10, no. 2, pp. 413–432, 1973.

[4] J. R. Rice, "On the degree of convergence of nonlinear spline approximation," in *Approximations with Special Emphasis on Spline Functions*. New York: Academic Press, 1969, pp. 349–365.

[5] D. L. Jupp, "Approximation to data by splines with free knots," *SIAM Journal on Numerical Analysis*, vol. 15, no. 2, pp. 328–343, 1978.

[6] S. Zhou and X. Shen, "Spatially adaptive regression splines and accurate knot selection schemes," *Journal of the American Statistical Association*, vol. 96, no. 453, pp. 247–259, 2001.

[7] F. Van den Bergh and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[8] S. Strasser, J. Sheppard, N. Fortier, and R. Goodman, "Factored evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 2, pp. 281–293, 2016.

[9] M. A. Potter and K. A. De Jong, "A cooperative coevolutionary approach to function optimization," in *International conference on parallel problem solving from nature*. Springer, 1994, pp. 249–257.

[10] C. De Boor and J. R. Rice, "Least squares cubic spline approximation, ii-variable knots," Department of Computer Science, Purdue University, Tech. Rep. CSD TR 21, 1968.

[11] J. H. Friedman and B. W. Silverman, "Flexible parsimonious smoothing and additive modeling," *Technometrics*, vol. 31, no. 1, pp. 3–21, 1989.

[12] J. H. Friedman, "Multivariate adaptive regression splines," *The Annals of Statistics*, vol. 19, no. 1, pp. 1–67, 1991.

[13] C. J. Stone, M. H. Hansen, C. Kooperberg, and Y. K. Truong, "Polynomial splines and their tensor products in extended linear modeling: 1994 wald memorial lecture," *The Annals of Statistics*, vol. 25, no. 4, pp. 1371–1470, 1997.

[14] H. Park and J.-H. Lee, "B-spline curve fitting based on adaptive curve refinement using dominant points," *Computer-Aided Design*, vol. 39, no. 6, pp. 439–451, 2007.

[15] W. Li, S. Xu, G. Zhao, and L. P. Goh, "Adaptive knot placement in b-spline curve approximation," *Computer-Aided Design*, vol. 37, no. 8, pp. 791–797, 2005.

[16] D. Michel and A. Zidna, "A new deterministic heuristic knots placement for b-spline approximation," *Mathematics and Computers in Simulation*, vol. 186, pp. 91–102, 2021.

[17] A. Razdan, "Knot placement for b-spline curve approximation," Tempe, AZ: Arizona State University, Tech. Rep. No Number, 1999.

[18] E. Aguilar, H. Elizalde, D. Cárdenas, O. Probst, P. Marzocca, and R. A. Ramirez-Mendoza, "An adaptive curvature-guided approach for the knot-placement problem in fitted splines," *Journal of Computing and Information Science in Engineering*, vol. 18, no. 4, p. 041013, 2018.

[19] R. N. Goldman and T. Lyche, *Knot insertion and deletion algorithms for B-spline curves and surfaces*. SIAM, 1992.

[20] R. Yeh, Y. S. Nashed, T. Peterka, and X. Tricoche, "Fast automatic knot placement method for accurate b-spline curve fitting," *Computer-Aided Design*, vol. 128, p. 102905, 2020.

[21] S. Miyata and X. Shen, "Adaptive free-knot splines," *Journal of Computational and Graphical Statistics*, vol. 12, no. 1, pp. 197–213, 2003.

[22] J. Pittman, "Adaptive splines and genetic algorithms," *Journal of Computational and Graphical Statistics*, vol. 11, no. 3, pp. 615–638, 2002.

[23] A. Gálvez and A. Iglesias, "Efficient particle swarm optimization approach for data fitting with free knot B -splines," *Computer-Aided Design*, vol. 43, no. 12, pp. 1683–1692, Dec. 2011.

[24] S. D. Mohanty and E. Fahnestock, "Adaptive spline fitting with particle swarm optimization," *Computational Statistics*, vol. 36, no. 1, pp. 155–191, 2021.

[25] J. Luo, H. Kang, and Z. Yang, "Knot placement for b-spline curve approximation via l-infinity, 1-norm and differential evolution algorithm," *Journal of Computational Mathematics*, vol. 40, no. 4, pp. 592–609, 2021.

[26] A. Gálvez and A. Iglesias, "Firefly algorithm for explicit b-spline curve fitting to data points," *Mathematical Problems in Engineering*, vol. 2013, no. 1, p. 528215, 2013.

[27] ——, "Efficient particle swarm optimization approach for data fitting with free knot b-splines," *Computer-Aided Design*, vol. 43, no. 12, pp. 1683–1692, 2011.

[28] S. Miyata and X. Shen, "Free-knot splines and adaptive knot selection," *Journal of the Japan Statistical Society*, vol. 35, no. 2, pp. 303–324, 2005.

[29] F. Yoshimoto, T. Harada, and Y. Yoshimoto, "Data fitting with a spline using a real-coded genetic algorithm," *Computer-Aided Design*, vol. 35, no. 8, pp. 751–760, 2003.

[30] D. L. Donoho and I. M. Johnstone, "Adapting to unknown smoothness via wavelet shrinkage," *Journal of the American Statistical Association*, vol. 90, no. 432, pp. 1200–1224, 1995.

[31] J. Snoek, H. Larochelle, and R. P. Adams, "Practical bayesian optimization of machine learning algorithms," *Advances in Neural Information Processing Systems*, vol. 25, 2012.