Constrained Evolutionary Optimization by Approximate Ranking and Surrogate Models

Thomas Philip Runarsson

Science Institute, University of Iceland tpr@hi.is

Abstract. The paper describes an evolutionary algorithm for the general nonlinear programming problem using a surrogate model. Surrogate models are used in optimization when model evaluation is expensive. Two surrogate models are implemented, one for the objective function and another for a penalty function based on the constraint violations. The proposed method uses a sequential technique for updating these models. The quality of the surrogate models is determined by their consistency in ranking the population rather than their statistical accuracy. The technique is evaluated on a number of standard test problems.

1 Introduction

In engineering design optimization expensive mathematical or physical models are common. Surrogate models are numerical or physical simplifications of these models, respectively. These simplifications are essentially inexpensive models. Surrogate models have been used in optimization [11], also known as approximate models [1] and meta-models [9]. In optimization surrogate models are based on scarce samples of the expensive model.

In evolutionary optimization there is an increased interest in applying surrogate models in place of expensive fitness models. A recent survey of fitness approximation in evolutionary computation is presented in [2] and a framework for evolutionary optimization established in [3]. Engineering design optimization problems commonly include constraints which are often more costly to evaluate than the objective. The handling of general nonlinear constraints in evolutionary optimization using surrogate models has not received much attention. This problem is tackled here by introducing two surrogate models: one for the objective function and another for a penalty function based on the constraint violations. The proposed method uses a sequential technique for updating these models. The quality of the surrogate models are determined by their consistency in ranking the population rather than their statistical accuracy or confidence. At each generation the surrogate models are updated and at least one expensive model evaluation is performed. The key new idea here is to evaluate the accuracy of the model by observing how it changes the behavior of the evolutionary algorithm (EA). That is, how does the surrogate model influence selection?

The paper is organized as follows. In section 2 an effective EA for the general nonlinear programming problem is described. In section 3 the surrogate models

implemented are presented. The performance of these models is evaluated on the sphere model. In section 4 the most effective way of sampling an expensive model is investigated. In section 5 the heuristic method of approximate ranking is described which will help determine how often the expensive model needs to be sampled. This is followed by a detailed experimental study of the proposed method on 13 benchmark functions in section 6. The paper concludes with a discussion and summary.

2 Constrained evolutionary optimization

Consider the general nonlinear programming problem formulated as

$$\text{minimize} \quad f(\mathbf{x}), \quad \mathbf{x} = (x_1, \ldots, x_n) \in \mathcal{R}^n, \tag{1}$$

where $f(\mathbf{x})$ is the objective function, $\mathbf{x} \in \mathcal{S} \cap \mathcal{F}$, $\mathcal{S} \subseteq \mathcal{R}^n$ defines the search space bounded by the parametric constraints $\underline{x}_i \leq x_i \leq \overline{x}_i$, and the feasible region $\mathcal{F}$ is defined by

$$\mathcal{F} = \{\mathbf{x} \in \mathcal{R}^n \mid g_j(\mathbf{x}) \leq 0 \quad \forall\ j\}, \tag{2}$$

where $g_j(\mathbf{x})$, $j = 1, \ldots, m$, are inequality constraints (equality constraints may be approximated by inequality constraints). Using a penalty function approach the constraint violations are treated as a single function,

$$\phi(\mathbf{x}) = \sum_{j=1}^{m} \max[0, g_j(\mathbf{x})]^2. \tag{3}$$

In [7] an effective algorithm for solving nonlinear programming problems was introduced. This algorithm is described by the pseudocode in fig. 1. The algorithm is essentially an improved version of the (μ, λ) evolution strategy (ES) using stochastic ranking presented in [6]. The algorithm in fig. 1 also uses stochastic ranking, which balances the influence of the penalty and objective function in determining the overall ranking of the population. In particular the population of individuals, of size λ, are ranked from best to worst, denoted $(\mathbf{x}_{1;\lambda}, \ldots, \mathbf{x}_{\mu;\lambda}, \ldots, \mathbf{x}_{\lambda;\lambda})$, and only the best μ are selected. The further modification, presented here, is simply the attempt to replace both $\phi(\mathbf{x})$ and $f(\mathbf{x})$ by surrogate models. The surrogate models only influence the ranking, the remainder of the algorithm is unchanged.

3 Nearest neighborhood regression

Deciding on a general surrogate model for optimization, especially when taking the no-free lunch theorems [10] into consideration, is difficult. Commonly used surrogate models include, among others, Kriging, polynomial regression and radial basis function [2]. Perhaps the most simple and transparent surrogate model is the nearest neighbor (NN) regression model, that is

$$\hat{h}(\mathbf{x}_i) = h(\mathbf{y}_j) \text{ where } j = \underset{k=1,\ldots,\ell}{\operatorname{argmin}} \|\mathbf{x}_i - \mathbf{y}_k\| \tag{4}$$

1 *Initialize*: $\sigma'_k = (\overline{\mathbf{x}}_k - \underline{\mathbf{x}}_k)/\sqrt{n}$, $\mathbf{x}'_k = \underline{\mathbf{x}}_k + (\overline{\mathbf{x}}_k - \underline{\mathbf{x}}_k)\mathbf{U}_k(0,1)$, $k = 1, \ldots, \lambda$
2 **for** $t := 1$ to T **do** *(generational loop)*
3 *evaluate*: $f(\mathbf{x}'_k)$, $\phi(\mathbf{x}'_k), k = 1 \ldots, \lambda$
4 *rank the* λ *points and copy the best* μ *in their ranked order*:
5 $(\mathbf{x}_i, \sigma_i) \leftarrow (\mathbf{x}'_{i;\lambda}, \sigma'_{i;\lambda})$, $i = 1, \ldots, \mu$
6 **for** $k := 1$ to λ **do** (*replication*)
7 $i \leftarrow \mod(k-1, \mu) + 1$ (*cycle through the best* μ *points*)
8 **if** $(k < \mu)$ **do** (*differential variation*)
9 $\sigma'_k \leftarrow \sigma_i$
10 $\mathbf{x}'_k \leftarrow \mathbf{x}_i + \gamma(\mathbf{x}_1 - \mathbf{x}_{i+1})$ (*if out of bounds retry using standard mutation*)
11 **else** (*standard mutation*)
12 $\sigma'_{k,j} \leftarrow \sigma_{i,j} \exp\left(\tau' N(0,1) + \tau N_j(0,1)\right)$, $j = 1, \ldots, n$
13 $\mathbf{x}'_k \leftarrow \mathbf{x}_i + \sigma'_k \mathbf{N}(0,1)$ (*if out of parametric bounds then retry*)
14 $\sigma'_k \leftarrow \sigma_i + \alpha(\sigma'_k - \sigma_i)$ (*exponential smoothing* [5])
15 **od**
16 **od**
17 **od**

Fig. 1. The improved (μ, λ) ES using the differential variation (lines $8-11$) performed once for each of the best $\mu - 1$ points ($\gamma \approx 0.8$). $U(0,1)$ is a uniform random number in $[0,1]$ and $N(0,1)$ is normally distributed with zero mean and variance one. $\tau' \propto 1/\sqrt{2n}$, $\tau \propto 1/\sqrt{2\sqrt{n}}$ and $\alpha \approx 0.2$, see also [5,7].

where $\mathbf{Y} \equiv \{\mathbf{y}\}_{j=1}^{\ell}$ are the set of points which have been evaluated using the expensive model $h(\mathbf{y})$. The approximate model passes exactly through the function values at the given points $\mathbf{Y}$. For this reason points evaluated using either the surrogate or expensive model may be compared directly. An obvious refinement of the nearest neighbor regression model is to weight the contribution of κ neighbors according to their distance to the query point $\mathbf{x}_i$. This is achieved by calculating the weighted average of the κ nearest neighbors,

$$\hat{h}(\mathbf{x}_i) = \frac{\sum_{j=1}^{\kappa} v_i h(\mathbf{y}_j)}{\sum_{j=1}^{\kappa} v_j} \tag{5}$$

where $v_j = 1/\|\mathbf{x}_i - \mathbf{y}_j\|^2$ is the distance-weighted contribution of the κ nearest neighbors. If there exists a neighbor j where $\|\mathbf{x}_i - \mathbf{y}_j\| < \varepsilon$ (a small value), then $\hat{h}(\mathbf{x}_i)$ is assigned $h(\mathbf{y}_j)$, that is, use (4).

The convenience in using nearest neighborhood regression is that learning consists of simply storing points, evaluated using the expensive model, in $\mathbf{Y}$. Each time a point is added the model is improved. If all the points are used, $\kappa = \ell$ in (5), the surrogate is called a *global model* and if only the nearest points are used a *local model*. Determining which model is most appropriate is problem dependent. For example, consider the case when the expensive model is the sphere model ($h(\mathbf{x}) = \sum_{k=1}^{n} x_k^2, n = 100$) and one is interested in the surrogate model resulting in the greatest rate of progress [8] towards the optimum. The progress rate may be simulated as follows: sample the expensive model space ℓ

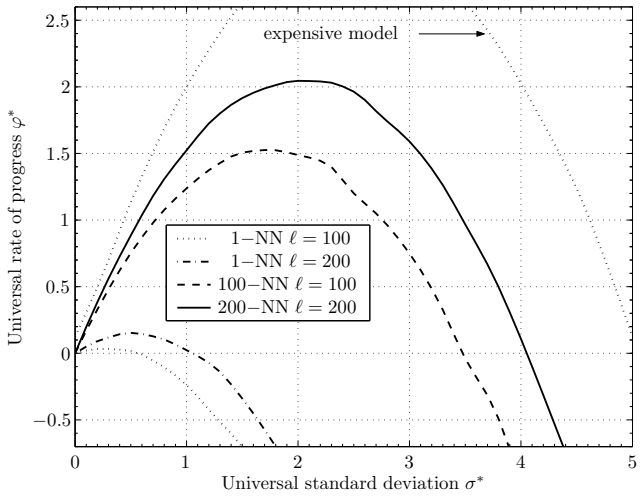

Fig. 2. The simulated progress rates for the local and global nearest neighbor regression models for the sphere model using a $(1, 100)$ evolution strategy.

times, construct a surrogate model, and compute the progress. The experiment is repeated 10.000 times and the expectation taken. The progress rates for a $(1, 100)$ ES using the surrogate models (4) and (5) for $\kappa = \ell$ is given in fig. 2. For the sphere model the weighted average of all nearest neighbors is the better surrogate model. Furthermore, it is clear that as the design space is sampled more densely, $\ell = 200$ rather than $\ell = 100$, the better the approximation and hence the progress. However, expensive models prohibit such an approach and so a sequential strategy, which regularly refines the surrogate model, is more suitable. Evolutionary algorithms are sequential (or generational) in nature and so this approach can be readily adapted.

4 Sampling strategy

In this section an attempt is made to answer the following question. Given λ offspring which should be evaluated using the expensive model so that progress toward the optimum is maximized? As in the previous section the answer is sought via simulation. First of all, the individuals from previous generations are in $\mathbf{Y}$. For the $(1, \lambda)$ progress rate simulation this implies that initially $\mathbf{Y} = \mathbf{x}_o$. Three different sampling strategies are investigated for selecting the offspring, $\mathbf{x}_i \notin \mathbf{Y}$, to add to the set $\mathbf{Y}$. The first simply selects the best individual according to the surrogate model, the second selects an individual at random, and the third strategy is to select the offspring that is the furthest distance from its closest neighbor in $\mathbf{Y}$. The last strategy is motivated by the fact that the individual furthest from its nearest neighbor in $\mathbf{Y}$ is the worst approximated. The algorithm used to perform this progress rate simulation is presented in fig. 3. The progress

```
1  Initialize: σ = σ*(r/n), r = ‖x* − x_o‖
2  for j := 1 to M do (Monte Carlo Simulation)
3    Y = x_o, x_i = x_o + N_i(0, σ²), i = 1 ..., λ
4    for k := 2 to (ℓ + 1) do sample points
5      compute ĥ(x_i) using (5), i = 1 ..., λ
6      y_k = x_i, x_i ∉ Y and i sampled according to a strategy
7      evaluate h(y_k) (expensive model)
8    od
9    φ_j = r − ‖x* − x_{1:λ}‖
10 od
11 Return expectation: φ* = (n/r) Σ_{j=1}^{M} φ_j/M
```

Fig. 3. Computer simulation used to estimate the progress rates for the different sampling strategies. $M = 10.000$ and $\mathbf{x}^*$ is the optimum.

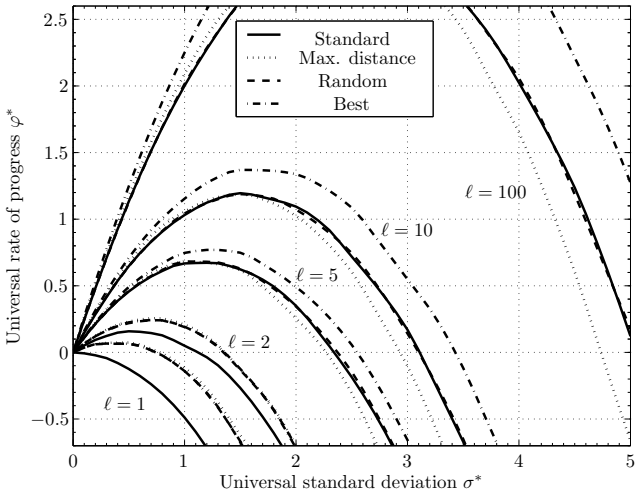

Fig. 4. $(1, 2\ell)$ progress rate for the three different sampling strategies, compared with standard method $(1, \ell)$, and different values of ℓ. $M = 10.000$.

rate simulations using $\lambda = 2\ell$ are presented in fig. 4 using the three different sampling strategies and various values of ℓ ($\lambda = 2\ell$) on the sphere model ($n = 100$). Furthermore, the standard progress rate is given for the $(1, \ell)$ strategy using the expensive model directly. For small values of ℓ and λ all sampling strategies result in a greater progress than the standard method. However, as ℓ and λ increase sampling the best point, according to the surrogate, is the only sampling strategy which results in greater progress than the standard approach.

```
1  approximate: f̂(x'_k), φ̂(x'_k), k = 1..., λ
2  rank and determine the parent set X_1 ≡ {x'_{i:λ}}_{i=1}^μ
3  y_j ← argmin_{x'_{i:λ}} i for x'_{i:λ} ∉ Y, j ← j + 1 (approximated best point)
4  evaluate: f(y_j), φ(y_j) (expensive model evaluation)
5  for i := 2 to λ do
6    approximate: f̂(x'_k), φ̂(x'_k), k = 1..., λ
7    determine new parent set X_i ≡ {x'_{i:λ}}_{i=1}^μ
8    if X_{i-1} ≠ X_i do (the parent set has changed)
9      y_j ← argmin_{x'_{i:λ}} i for x'_{i:λ} ∉ Y, j ← j + 1
10     evaluate: f(y_j), φ(y_j)
11   else (parent set remains unchanged)
12     break (exit for loop)
13   od
14 od
```

Fig. 5. The *approximate ranking* procedure where initially $Y \neq \emptyset$.

5 Approximate ranking

In the previous section different sampling strategies were investigated. The results suggest that it would be most beneficial, in terms of progress, to evaluate the best individuals. Therefore, this is the strategy adopted. However, the number of samples to be evaluated using the expensive model was not established. Clearly one would like to minimize the total number of expensive model evaluations and yet retain a good quality surrogate model.

From the EA's perspective as long as a good approximate estimate of the parent set is found there is no need to call the expensive model. Therefore, the simple heuristic proposed is that the surrogate model is approximately correct as long as the parent set does not change when the surrogate model is improved. As a result the following approximate ranking method, shown in fig. 5, is used to determine indirectly the number expensive fitness evaluations needed at any given generation. The maximum number evaluations per generation is therefore λ and the minimum number is 1.

6 Experimental study

In the following experiment 13 nonlinear programming problems from the literature [6,7] are studied. The improved ES in fig 1, denoted ιES, is compared with a surrogate version of the same algorithm, denoted $\hat{\iota}$ES. In the surrogate version lines 3–4 in fig. 1 are replace with the approximate ranking described in fig. 5. The only additional modification is that the set $\mathbf{Y}$ has a fixed size ℓ so that any new point added to the set automatically overwrites the oldest point. The idea here is that the old points are furthest away from the current population. The set $\mathbf{Y}$ should, however, be large enough to accept all new point at any given

generation, that is $\ell \geq \lambda$. Clearly, the smaller ℓ becomes the faster the surrogate model is evaluated. There exist also a number of fast implementations of the NN algorithm, see for example [4].

The experimental setup is the same as in [6,7] where all experiments are run for a fixed number of generations, $T = 350000/\lambda$ with the exception of problem `g12` which is run for $T = 35000/\lambda$ generations. As a first experiment the improved $(15, 100)\iota$ES is compared with its surrogate counterpart using $\ell = 100$ and $\ell = 200$ respectively. In table 1 the results, for 30 independent runs, using the simple nearest neighbor regression (4) as the surrogate model is given. The quality of solutions found by surrogate version are similar to the one using the expensive model evaluations. The mean number of expensive model evaluations needed to locate the best solution may be found in the column labeled `feval` along with its standard deviation. The number of expensive model evaluation has been reduced for all problems, but only significantly for functions `g01`, `g08`, `g09` and `g12`. Using a set size of $\ell = \lambda$ is also sufficiently large.

It is interesting to observe how the number of expensive fitness evaluations changes per generation due to the approximate ranking. This is shown in fig. 6 for two test functions based on an average of 100 independent runs. Also shown is the corresponding mean best objective value. For `g01` the entire population must be evaluated using the expensive model at the beginning of the run. The number is reduced once the population enters the feasible region, then remains constant for some time at 40% of the population and then falls again toward the end of the run. For `g02` the number of expensive function evaluations increases towards the end of the run.

When using the surrogate model (5) with $\kappa = \ell$ a poor global search performance is observed. For this reason it is decided to use $\kappa = 10$ and the entire

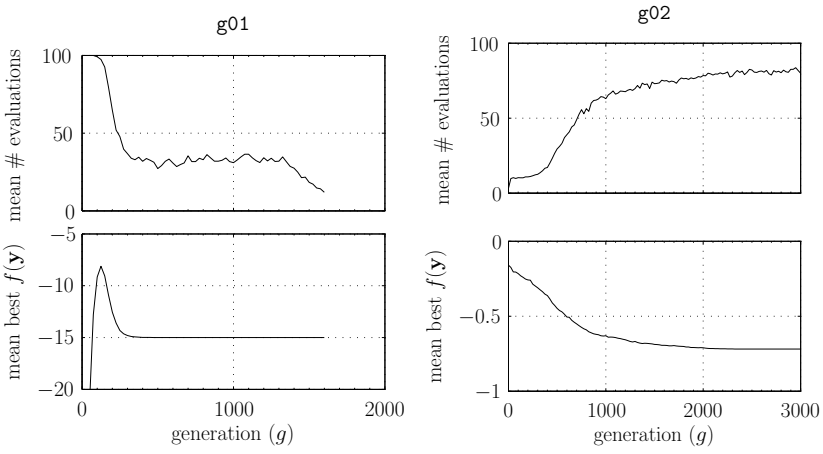

Fig. 6. Top plots shows the mean number of expensive model evaluation per generation and the bottom the corresponding mean best objective function value.

Table 1. Statistics for 30 independent runs using a $(15, 100)$ ES and its surrogate version using 1−NN.

	objective					feval	
	best	median	mean	st. dev.	worst	mean	std
g01 − ιES	−15.000	−15.000	−15.000	3.6E−16	−15.000	122163	5062
îES($\ell = \lambda$)	−15.000	−15.000	−15.000	0.0E+00	−15.000	67341	6116
îES($\ell = 2\lambda$)	−15.000	−15.000	−15.000	3.2E−16	−15.000	61783	7283
g02 − ιES	−0.803619	−0.760456	−0.753209	3.7E−02	−0.609330	348606	2578
îES($\ell = \lambda$)	−0.803617	−0.708854	−0.707586	5.7E−02	−0.570960	209416	36582
îES($\ell = 2\lambda$)	−0.803619	−0.744804	−0.731058	6.4E−02	−0.527944	211497	37225
g03 − ιES	−1.001	−1.001	−1.001	1.7E−05	−1.001	324206	27783
îES($\ell = \lambda$)	−1.001	−1.001	−1.001	3.2E−06	−1.001	275094	7661
îES($\ell = 2\lambda$)	−1.001	−1.001	−1.001	5.7E−07	−1.001	278894	6373
g04 − ιES	−30665.539	−30665.539	−30665.539	2.2E−11	−30665.539	68023	6004
îES($\ell = \lambda$)	−30665.539	−30665.539	−30665.539	7.3E−12	−30665.539	53815	2669
îES($\ell = 2\lambda$)	−30665.539	−30665.539	−30665.539	7.3E−12	−30665.539	53216	2479
g05 − ιES	5126.497	5126.497	5126.497	5.8E−12	5126.497	62976	3388
îES($\ell = \lambda$)	5126.497	5126.497	5126.497	2.0E−12	5126.497	60844	3174
îES($\ell = 2\lambda$)	5126.497	5126.497	5126.497	1.4E−07	5126.497	62814	3783
g06 − ιES	−6961.814	−6961.814	−6961.814	6.4E−12	−6961.814	55203	2900
îES($\ell = \lambda$)	−6961.814	−6961.814	−6961.814	3.6E−12	−6961.814	46524	3349
îES($\ell = 2\lambda$)	−6961.814	−6961.814	−6961.814	3.6E−12	−6961.814	46424	2305
g07 − ιES	24.306	24.323	24.337	4.1E−02	24.635	347393	12789
îES($\ell = \lambda$)	24.308	24.336	24.375	7.6E−02	24.591	237195	41850
îES($\ell = 2\lambda$)	24.307	24.326	24.337	3.5E−02	24.450	231089	44286
g08 − ιES	−0.095825	−0.095825	−0.095825	4.2E−17	−0.095825	64863	41349
îES($\ell = \lambda$)	−0.095825	−0.095825	−0.095825	8.8E−18	−0.095825	2504	1957
îES($\ell = 2\lambda$)	−0.095825	−0.095825	−0.095825	7.6E−18	−0.095825	2634	1766
g09 − ιES	680.630	680.630	680.630	7.4E−04	680.635	264120	82602
îES($\ell = \lambda$)	680.630	680.631	680.632	2.4E−03	680.638	145955	23840
îES($\ell = 2\lambda$)	680.630	680.630	680.631	1.8E−03	680.637	133523	17545
g10 − ιES	7049.404	7064.109	7082.227	4.2E+01	7258.540	304066	86127
îES($\ell = \lambda$)	7050.290	7071.520	7086.310	4.0E+01	7191.870	283637	75378
îES($\ell = 2\lambda$)	7049.620	7099.250	7118.900	7.1E+01	7367.780	295030	65746
g11 − ιES	0.750	0.750	0.750	1.8E−15	0.750	47046	2968
îES($\ell = \lambda$)	0.750	0.750	0.750	1.1E−16	0.750	39289	2453
îES($\ell = 2\lambda$)	0.750	0.750	0.750	1.1E−16	0.750	38566	3004
g12 − ιES	−1.000000	−1.000000	−1.000000	0.0E+00	−1.000000	19726	1462
îES($\ell = \lambda$)	−1.000000	−1.000000	−1.000000	0.0E+00	−1.000000	4200	900
îES($\ell = 2\lambda$)	−1.000000	−1.000000	−1.000000	0.0E+00	−1.000000	4216	921
g13 − ιES	0.053942	0.053942	0.111671	1.4E−01	0.438804	197606	117774
îES($\ell = \lambda$)	0.053942	0.053942	0.143746	1.6E−01	0.438902	93302	14605
îES($\ell = 2\lambda$)	0.053942	0.053942	0.182229	1.8E−01	0.181425	90768	12624

experiment is repeated. This result is given in table 2. Using the 10-NN regression reduces the number of expensive function evaluations for test function g12 even further but now there are some difficulties in locating feasible solutions for test

Table 2. Statistics for 30 independent runs using a $(15, 100)$ ES and its surrogate version using 10−NN.

	objective					feval	
	best	median	mean	st. dev.	worst	mean	std
g01 − ιES	−15.000	−15.000	−15.000	3.6E−16	−15.000	122163	5062
îES($\ell = \lambda$)	−15.000	−15.000	−15.000	0.0E+00	−15.000	64320	6219
îES($\ell = 2\lambda$)	−15.000	−15.000	−15.000	3.2E−16	−15.000	60522	7030
g02 − ιES	−0.803619	−0.760456	−0.753209	3.7E−02	−0.609330	348606	2578
îES($\ell = \lambda$)	−0.792608	−0.731221	−0.706970	6.8E−02	−0.552092	254828	9817
îES($\ell = 2\lambda$)	−0.785266	−0.679370	−0.679007	5.6E−02	−0.564173	257185	8281
g03 − ιES	−1.001	−1.001	−1.001	1.7E−05	−1.001	324206	27783
îES($\ell = \lambda$)	−1.000	−1.001	−1.001	6.8E−05	−1.001	259112	19095
îES($\ell = 2\lambda$)	−1.000	−1.001	−1.001	1.1E−04	−1.001	257262	15739
g04 − ιES	−30665.539	−30665.539	−30665.539	2.2E−11	−30665.539	68023	6004
îES($\ell = \lambda$)	−30665.539	−30665.539	−30665.539	7.3E−12	−30665.539	51369	3354
îES($\ell = 2\lambda$)	−30665.539	−30665.539	−30665.539	7.2E−12	−30665.539	50733	3361
g05 − ιES	5126.497	5126.497	5126.497	5.8E−12	5126.497	62976	3388
îES($\ell = \lambda$)	5126.497	5126.497	5126.497	8.8E−08	5126.497	59657	2555
îES($\ell = 2\lambda$)	5126.497	5126.497	5126.497	5.6E−12	5126.497	58923	3288
g06 − ιES	−6961.814	−6961.814	−6961.814	6.4E−12	−6961.814	55203	2900
îES($\ell = \lambda$)	−6961.814	−6961.814	−6961.814	3.6E−12	−6961.814	41787	2773
îES($\ell = 2\lambda$)	−6961.814	−6961.814	−6961.814	3.6E−12	−6961.814	43462	2574
g07 − ιES	24.306	24.323	24.337	4.1E−02	24.635	347393	12789
îES($\ell = \lambda$)	24.308	24.334	24.358	6.2E−02	24.531	234441	50231
îES($\ell = 2\lambda$)	24.307	24.325	24.331	3.0E−02	24.476	209696	43488
g08 − ιES	−0.095825	−0.095825	−0.095825	4.2E−17	−0.095825	64863	41349
îES($\ell = \lambda$)	−0.095825	−0.095825	−0.095825	1.3E−17	−0.095825	18801	17018
îES($\ell = 2\lambda$)	−0.095825	−0.095825	−0.095825	1.1E−17	−0.095825	14188	12716
g09 − ιES	680.630	680.630	680.630	7.4E−04	680.635	264120	82602
îES($\ell = \lambda$)	680.630	680.630	680.630	8.1E−04	680.634	124675	15821
îES($\ell = 2\lambda$)	680.630	680.630	680.631	2.3E−03	680.643	122770	16603
g10 − ιES	7049.404	7064.109	7082.227	4.2E+01	7258.540	304066	86127
îES($\ell = \lambda$)	7054.928	7952.319	7697.312	9.1E+02	10282.820	170499	76428
îES($\ell = 2\lambda$)	7119.248	7545.843	7373.358	5.2E+01	8909.700	160202	64015
g11 − ιES	0.750	0.750	0.750	1.8E−15	0.750	47046	2968
îES($\ell = \lambda$)	0.750	0.750	0.750	1.1E−16	0.750	39148	2469
îES($\ell = 2\lambda$)	0.750	0.750	0.750	1.1E−16	0.750	39894	2493
g12 − ιES	−1.000000	−1.000000	−1.000000	0.0E+00	−1.000000	19726	1462
îES($\ell = \lambda$)	−1.000000	−1.000000	−1.000000	0.0E+00	−1.000000	2742	598
îES($\ell = 2\lambda$)	−1.000000	−1.000000	−1.000000	0.0E+00	−1.000000	2536	477
g13 − ιES	0.053942	0.053942	0.111671	1.4E−01	0.438804	197606	117774
îES($\ell = \lambda$)	0.053942	0.053942	0.18223	1.8E−01	0.438836	102921	31912
îES($\ell = 2\lambda$)	0.053942	0.053942	0.20789	1.9E−01	0.438809	93012	13840

function g10 where only 18/30 and 26/30 feasible solutions are found for $\ell = 100$ and 200 respectively. In general there is no improvement over using just the simple nearest neighbor regression, i.e. 1-NN.

7 Summary

A new approach using surrogate models for global optimization has been presented and tested on some general nonlinear programming problems. A simple heuristic is proposed where the surrogate model is said to be sufficiently accurate if any improvement in the surrogate does not change the parent set $\{\mathbf{x}'_{i;\lambda}\}_{i=1}^{\mu}$ at any given generation.

It is clear that the best surrogate model will depend on the properties of the expensive model. For the sphere model the distance-weighted NN regression model is more appropriate than a simple nearest-neighbor model. However, for the general nonlinear programming problems the $1-$NN model seems more appropriate. The sampling methods also influence search performance. It was found that expensive model evaluation of the best approximated points was the best strategy for the sphere model. This is not necessarily the best strategy for other surrogate and expensive models. The idea of using approximate ranking is, however, equally applicable to other surrogate models and sampling strategies. This and applications to real world problems will be the topic of future research.

References

1. J.-F. M. Barthelemy and R.T. Haftka. Approximation concepts for optimum structural design – A review. *Stuctural Optimization*, 5:129–144, 1993.
2. Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 2003.
3. Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5), October 2002.
4. J. McNames. A fast neartest neighbor algorithm based on a principle axis search tree. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9), September 2001.
5. T. P. Runarsson. Reducing random fluctuations in mutative self-adaptation. In *Parallel Problem Solving from Nature VII (PPSN-2002)*, volume 2439 of *LNCS*, pages 194–203, Granada, Spain, 2002. Springer Verlag.
6. T. P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, September 2000.
7. T. P. Runarsson and X. Yao. Search biases in constrained evolutionary optimization. *IEEE Transactions on System, Man, and Cybernetics: Part C*, (to appear, see http://www.hi.is/~ tpr), 2004.
8. H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, New-York, 1995.
9. T. W. Simpson, J. Peplinski, P. N. Koch, and J. K. Allen. On the use of statistics in design and the implications for deterministic computer experiments. In *Design Theory and Methodology - DTM'97*, number DETC97/DTM-3881, Sacramento, CA, 1997. ASME.
10. D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.
11. S. Yesilyurt and A. T. Patera. Surrogates for numerical simulations; optimization of eddy-promoter heat exchangers. *Comp. Methods Appl. Mech. Engr.*, 121:231–257, 1995.