Evolving Radial Basis Function Networks via GP for Estimating Fitness Values using Surrogate Models

Ahmed Kattan and Edgar Galvan

***Abstract*— In real-world problems with candidate solutions that are very expensive to evaluate, Surrogate Models (SMs) mimic the behaviour of the simulation model as closely as possible while being computationally cheaper to evaluate. Due to their nature, SMs can be seen as heuristics that can help to estimate the fitness of a candidate solution without having to evaluate it. In this paper, we propose a new SM based on Genetic Programming (GP) and Radial Basis Function Networks (RBFN), called *GP-RBFN Surrogate*. More specifically, we use GP to evolve both: the structure of a RBF and its parameters. The SM evolved by our algorithm is tested in one of the most studied NP-complete problem (MAX-SAT) and its performance is compared against RBFN Surrogate, GAs, Random Search and (1+1) ES. The results obtained by performing extensive empirical experiments indicate that our proposed approach outperforms the other four methods in terms of finding better solutions without the need of evaluating a large portion of candidate solutions.**

I. Introduction

Many real-world problems require experiments to run for days, weeks or even months to complete due to the nature of the problem. For instance, most engineering design problems need simulations to evaluate design objectives along with constraints as function of design variables. Thus, it is necessary to consider and simulate several scenarios. For many of these problems, one single simulation could take several hours, or even days to finish. As a result of this, the simulation of several scenarios turns out to be almost an impossible task to carry out.

Moreover, many optimisation problems are black-box problems, i.e., problems of unknown class, and they are possibly mathematically ill-behaved (e.g., discontinuous, non-linear, non-convex). Possible ways of dealing with such optimisation problems include the use of a high-performance computing technology with multi-threading programming, or an approximation model that approximates a given objective function.

Surrogate Models (SMs), also known as response surface models, are approximation models, that mimic the behaviour of the simulation model as closely as possible while being fast surrogates for time-consuming computer simulations. In a nutshell, SMs work by running simulations at a set of points and fitting response surfaces to the resulting input-output data. In this paper, we will refer to a solution in the problem's search space and its real-fitness value as a data-point. In Section III, we further describe this.

Some commonly used approximation functions include: Polynomial Regression, Artificial Neural Networks, Radial Basis Function Networks (RBFNs) and Support Vector Machines [5].

There are some interesting works in the area of SMs. For instance, Lim et al. [9] proposed a generalised surrogate-assisted evolutionary framework for optimisation of problems that are computationally expensive to evaluate. The authors introduced the idea of employing several on-line local SMs which are constructed using data points that lie in the vicinity of an initial guess. The improved solutions generated by the local search are used to replace the original individual(s). In their work, the framework has been presented with single objective optimisation and multi-objective optimisation.

Lian et al. [8] proposed an enhancement for the standard GA by using local surrogate search to expedite its convergence. The model uses GA to generate a population of individuals and rank them with a real function. Then, a gradient-based local search is performed on the SM to find new promising solutions. Both, the GA and local search are alternatively used under a trust-region framework until the optimum is found. The trust-region framework is used to assure that the surrogate's solutions are converging towards the original problem.

More recently, Moraglio and Kattan [10] showed that SMs can be naturally generalised to encompass combinatorial spaces based in principle on any arbitrarily complex underlying solution representation by generalising their geometric interpretation from continuous to general metric spaces. An illustrative example is given related to Radial Basis Function Networks (RBFNs), which can be successfully used as surrogate models to optimise combinatorial problems defined on the Hamming space associated with binary strings. The authors illustrated the methodology with the well-known NK-landscape problem.

These works are just few examples that show the potential of SMs to be used as accurate approximation models. That is, they can be seen as heuristics that can help to estimate the fitness of a candidate solution without having to evaluate it.

In this paper we take a step further on this direction and propose a new SM that is obtained using Genetic Program-

Ahmed Kattan is with the Evolutionary Design and Optimisation Group, Computer Science Department, Um Al-Qura University, Saudi Arabia and Edgar Galvan is with the Distributed Systems Group, School of Computer Science and Statistics, Trinity College Dublin, email: Ajkattan@uqu.edu.sa, edgar.galvan@scss.tcd.ie.

978-1-4673-1509-8/12/$31.00 ©2012 IEEE

ming (GP) [7], [12] and RBFN [1]. More specifically, GP has been used to control the magnitude of the RBFs' output in the RBFN in such a way to minimise the surrogate's prediction errors (more of this is presented in Section II). Thus, the goals of this paper are twofold:

- To improve standard RBFN surrogate model by using GP to tune its parameters, and
- To show how SMs can successfully been used to evolve solutions without the need of evaluating a large number of candidate solutions.

This paper is structured as follows. In the following section we describe standard Radial Basis Function Networks. Section III describes our approach (i.e., evolving SMs via GP and RBFNs). Section IV presents the experimental setup used to conduct our experiments followed by results and discussion, presented in Section V. Finally, Section VI draws some conclusions.

II. Radial Basis Function Networks

There are a number of known approaches to learn a function that belongs to a certain class of functions from existing data-points [1] (i.e., finding a function in that class that interpolates and best fits the data-points according to some criteria). Some of these include Genetic Programming (GP), RBFN Interpolation, Artificial Neural Networks and Gaussian Process Regression (also known as Kriging) [2].

Gaussian Process Regression is a very powerful method with a solid theoretical foundation, which not only can make a rational extrapolation about the location of the global optimum, but also gives an interval of confidence about the prediction made. RBFN Interpolation is conceptually simpler than Gaussian Process Regression and can extrapolate the global optimum from the known data-points. In this paper, we focus on RBFNs as surrogate models.

GP is a powerful method for approximating unknown functions. Thus, in this work, we take advantage of this and explore the possibility of improving SMs using GP to control the accuracy of the RBFN.

A. RBFN Representation

Radial Basis Function Networks (RBFNs) can be seen as a variant of artificial neural network that uses radial basis functions as activation functions [1]. Typically, RBFN consists of three layers: input, hidden, and output layer. The relationship between the input and the hidden layer is determined by the RBF activation function. The nodes in the output layer usually perform a simple summation for the linear weights of these activations. RBFNs have successfully been used in function approximation, time series prediction, and control [1].

A radial basis function (RBF) is a real-valued function $\phi : \mathbf{R}^n \rightarrow \mathbf{R}$; its value depends only on the distance from some point c, called *centre*, so that $\phi(\mathbf{x}) = \phi(\|x_q - \mathbf{c}\|)$. The point c is a parameter of the function and the point x_q is the query point to be estimated. The norm is usually Euclidean, so $\|x_q - \mathbf{c}\|$ is the Euclidean distance between c and x_q. Since we use a generalised RBFN (fully described in [10]), the Euclidean distance has been replaced with a metric distance that naturally encompass the GA representation of our optimisation problems (details are presented later in this section). There are several types of RBF functions, including: Gaussian, Multiquadric, Inverse Quadratic and Inverse Multiquadric. In this paper we use the Gaussian functions of the form:

$$\phi(x) = \exp(-\beta \|x_q - \mathbf{c}\|^2)$$

where $\beta > 0$ is the width parameter. Radial basis functions are typically used to build function approximations of the form:

$$y(x) = w_0 + \sum_{i=1}^{N} w_i \, \phi(\|x_q - \mathbf{c}_i\|) \tag{1}$$

Thus, $y(x)$ is used to approximate the real-objective function when evaluating an individual. The approximating function $y(x)$ is represented as a sum of N radial basis functions, each associated with a different centre c_i, a different width β_i, and weighted by an appropriate coefficient w_i, plus a bias term w_0. In principle, any continuous function can be approximated with arbitrary accuracy by a sum of this form, if a sufficiently large number N of radial basis functions is used. The bias w_0 can be set to the mean of the values of the known data-points from the training set that are used to train the surrogate model, or set to 0.

B. Training

Training the RBFNs requires to find, as stated before, three parameters: (a) the centres $\mathbf{c}_i$, (b) the values of w_i in such a way that the predictions on the training set minimises the errors and, finally, (c) the RBF width parameters β_i.

The centers can be chosen to coincide with the known data-points and evaluate them with the real fitness evaluation. The β value can be either fixed for all N linear RBFs (global) or it can be customised for each RBF (local). Typically, the β value is set as $1/D^2$ where D is the mean pairwise distance between data-points in the nearest n neighbours from known data-points set [2]. In this work, we let GP to find the "best" global β value for all RBFs in the RBFN (more on this is explained in Section III).

The value of β controls the radius of each RBF (spreading on the search space to cover all the other centres), so that each known function value at a centre can potentially contribute significantly to the prediction of the function value of any point in space. Finally, the weights vector can be calculated by solving the system of N simultaneous linear equations in w_i by requiring that the unknown function predicts exactly the known data-points. Formally, we have:

$$y(\mathbf{x}_i) = b_i, \; i = 1 \ldots N.$$

[1]Data-point is the pair of solution and its real fitness value.

[2]Note that the distance definition depends on the underlying representation of the problem. Since in this paper we aim to build surrogate model for GA search, we use the Hamming distance.

Setting $g_{ij} = \phi(||\mathbf{x}_j - \mathbf{x}_i||)$, the system can be written in a matrix form as $\mathbf{Gw} = \mathbf{b}$ where $\mathbf{b}$ is a vector of the true fitness values of the data-points that have been used to train the surrogate. The matrix $\mathbf{G}$ is non-singular [3], because we guarantee the points $\mathbf{x}_i$ are distinct, so the weights w can be solved by simple linear algebra:

$$\mathbf{w} = \mathbf{G}^{-1}\mathbf{b}$$

The value of the bias term w_0 in Equation 1 is set to the mean value of the known data-points, i.e., the mean of vector $\mathbf{b}$. In this way, the predicted function value of a point which is out of reach of the influence of all centres is by default set to the average of their function values.

C. Interpolation

The RBFNs can be naturally generalised from continuous spaces to any representation [10]. This can be done by considering distances defined directly on the underlying representation. The generalisation is possible because the representation of RBFN, their training, and the prediction do not depend directly on the representation, but depends only on the distances between solutions [10].

Once the RBF parameters are determined, the model is ready to estimate the fitness of any unseen point. Thus, the fitness $f(x)$ of unknown point x_q in the search space is predicted by weighted linear combination of:

$$f(x) = w_0 + \sum_{i=1}^{N}[w_i * \phi(d(x_q, \mathbf{c}_i))]$$

where, w_i is a vector of weights that has been calculated during the training phase and ϕ is the kernel function which is defined in Equation 1. Finally, $d(x_q, \mathbf{c}_i)$ is the distance between the new point x_q and the training set points c_i.

III. GP-RBFN Surrogate

There are two main problems associated with training RBFN. Firstly, determining the optimal architecture and secondly, determining the optimal parameters [3]. In this work, we let the GP to evolve the RBFN parameters. In particular, we search for the best β value i.e., RBF radius parameter.

To do so, we train the RBFN on a small set of solutions evaluated using the real objective function (called the *known-points* set) as described in Section II-B. As mentioned previously, we let the GP to decide the β value. The trained RBFN (without calculating its β) is then passed to GP as a function in the form:

$$\sum_{i=0}^{N} w_i * exp(-\beta\|x_q - \mathbf{c}_i\|)$$

we refer to this function as ***RBF node***. Thus, this can be seen as passing a collection of RBF functions to the GP together with its weights parameters. For each RBF node, GP allocates a β value (randomly selected) from the interval $(0, 1]$. Note that all RBF nodes (in the same tree or in different trees) have the same parameters. The only difference is in the β values. Thus, GP has the freedom to modify the β value in each RBF node and also combines it with constants values to control the magnitude of its output in such a way to minimise the prediction errors on the known-points set.

TABLE I
PRIMITIVE USED IN OUR WORK.

Primitive	*Arity*	*Input type(s)*	*Output type*
Plus, Minus, Div, Mul	2	Real number	Real number
RBF	1	Unseen solution	Real number
Constants [-1,1]	0	N/A	Real number

The aim is to evolve a program that can generalise its predictions on unseen solutions. For this, GP is supplied with primitives that allows it to build SMs. Table I shows the primitive set used by our GP system. The trees shown in Figure 1 provide an example of two evolved 'surrogate program' by means of a GP system.

Thus, when a RBFN is executed to estimating the fitness of an unseen point (as defined in Equation 1) we replace the RBFs with an evolved expression. So, an evolved expression, for example, may contain RBF node multiplied by a constant (Figure 1 (a)), or it can be a single RBF node with a suitable β value (Figure 1 (b)). Also, it might be the case that GP decides to evolve an expression that uses multiple RBF nodes combined with division or multiplication, for example.[4]

More formally, let the surrogate known-points set be c_n where $n \in \{1, 2, ..., N\}$ and let the i^{th} individual in the population denoted as I_i. Finally, let x_q be an unseen point in the problem's search space. Thus, any program in the population can estimate the fitness of an unseen solution as follows:

$$EstimatedFitness(I_i, x_q) = w_0 + I_i(c_n) \qquad (2)$$

where w_0 is the bias term defined in Equation 1 (mean value of known-points set).

Similar to the traditional procedure of surrogate model based optimisation (SMBO) [6], [10] the best evolved program is used as a SM in a generational GA (referred in this paper as *fast GA*, because it does not use the real fitness function, instead it uses a cheap surrogate model). Thus, the role of the fast GA in the SMBO procedure is to infer the location of a promising solution of the problem using an evolved SM.

The best solution found by the SM after performing a fast GA run is then passed to the real objective function to calculate its real fitness, updating the known-points and re-training the system. This process iterates until a maximum number of evaluations is reached, as outlined in Algorithm 1. Note that the system adds a random solution to the known-points set if the surrogate suggests a solution that is already present in the known-points set. This can happen because the points in the training set act as centres of the evolved RBF expressions (they are aligned with the peak of the Gaussian).

[3]A singular matrix does not have a matrix inverse if and only if its determinant is 0.

[4]We observed that in most cases GP evolves an expression that uses single RBF node combined with a constant value using an arithmetic operator.

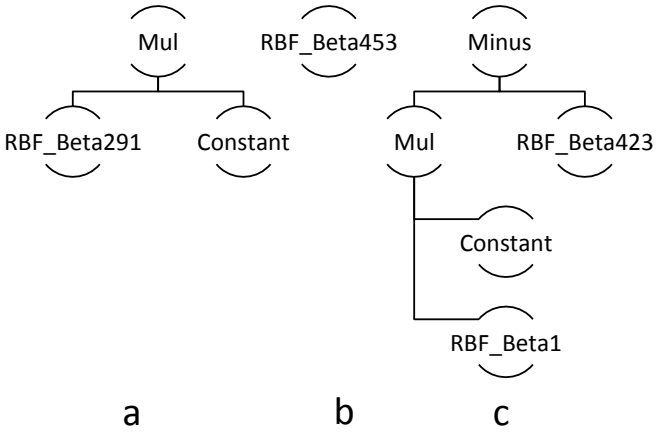

Fig. 1. Three examples of evolved programs. Each program corresponds to one of three types of expressions (read text in Section V).

The approximating function is the sum of these evolved RBFs. Depending on the β values (randomly allocated by GP for each RBF node), the approximating function would tend to have local and global optima exactly in the centres of the RBFs, i.e., the known-points. Then when the fast GA finds an optimum, this may happen to be a point in the training set! Generally, this tends to happen when the β values are not small enough. However, when β is large enough, the RBFs are spread and overlapping, and the approximating function tends to be unimodal with the position of the peak influenced by the sum of all the RBFs, and it is not a training point. Another reason for finding the same training point as optimum is that the training point might correspond to the best of the real function (which was already suggested by the SM in previous iterations). This indicates that the problem has already been solved.

A. GP Training

To train the GP system to evolve surrogate models, we use a simple two-rounds cross-validation scheme. The first round of cross-validation involves partitioning the known-points set into two separated equal subsets, performing the training on one subset (training set), and validating the GP individuals on the other subset (testing set). In the second round of cross-validation, partitions are swapped and the validation results are averaged over the rounds.

Note that once the GP generates a surrogate program, it passes it to the fast GA (as fitness measure) to find promising points and then update the known-points set. Thus, at each iteration the system adds a new point to the known-points set. Then GP generates a new surrogate program to maintain the new knowledge about the under-sampled search space.

B. Fitness Function

The fitness evaluation is designed to encourage GP to evolve programs that minimise the absolute prediction errors on the training set. It is worth mentioning that programs which produce low prediction errors are not necessary able to approximate the real objective function and may not be able to guide the search to optimal solutions. In fact, preliminary experiments showed that when the fitness function is guided by the average absolute error of predictions, GP tends to evolve programs that produce numbers close to the training set average. This bias the system to evolve programs that produce the average value of the training set. To alleviate this problem, we also measure the correlation between predictions produced by an individual and the real fitness values in the training set values. Also, the tree size is considered in the fitness evaluation to encourage GP to evolve small programs (hence fast to evaluate).

More specifically, let the individual I_i prediction for an unseen solution x_q be the function $Estimation(I_i, x_q)$, defined in Equation 2, and the total number of training points

Algorithm 1: GP-RBFN Surrogate Search.

```
1  Known-points-Set = Generate-Solution(initial-set-size);
2  Expensive-Evaluation(Known-points-Set);
3  GP-RBFN-Surrogate.Train(Known-points-Set);
4  while Expensive-evaluation-budget Not Finished do
5  |  P = Fast-GA-Run(population-size, generations,
   |  GP-RBFN-Surrogate);
6  |  P is best solution found by a cheap
   |  GA run
7  |  if Promising-Point is unique then
8  |  |  Expensive-Evaluation(P);
9  |  |  Known-points-Set.add(P);
10 |  end
11 |  else
12 |  |  Known-points-Set.add(Random Unique Point);
13 |  end
14 |  GP-RBFN-Surrogate.Train(Known-points-Set);
15 end
```

be N. We have that the prediction error (PE) can be defined as follows:

$$PE = \frac{\sum_{n=0}^{N} abs(Estimation(I_i, x_q) - RealFitness(x_q))}{N}$$

The correlation between predictions and the known-points set values can be defined as the function $Corr(\hat{f(t)}, f(t))$ where $f(t)$ is the function that returns the real fitness values for solutions (Equation 3) and $\hat{f(t)}$ is the approximation function that returns the estimated fitness values as defined in Equation 2. Finally, the tree size of the individual I_i can be represented as the function $Size(I_i)$. Putting all together, we calculate the fitness function (F_f) as follows:

$$F_f = PE + [Corr(\hat{f(t)}, f(t)) \times -1] + Size(I_i)$$

Thus, here we aim to minimise the fitness function as to evolve better programs. The reason of using $Corr(\hat{f(t)}, f(t)) \times -1$ is to encourage the system to produce positive correlation while at the same time minimise the fitness function.

IV. Experimental Setup

The Boolean satisfiability problem, also known as SAT, is one of the most studied NP-complete problems (e.g., see [4], [13]) and it has been used as an illustrative example to show that our model is able to improve the standard RBFN surrogate models. Thus, our aim is not to effectively solve the SAT problem itself, but rather to show that our approach can achieve better solutions using low number of evaluations than standard RBFN surrogate as well as other standard search algorithms.

The target in SAT is to determine whether it is possible to set the variables of a given Boolean expression in such a way to make the expression true. The expression is said to be satisfiable if such an assignment exists. A related problem, known as the Maximum Satisfiability problem, or MAX-SAT, consists in determining the maximum number of clauses of a given Boolean formula that can be satisfied by some assignment.

We treat MAX-SAT as an optimisation problem with the following objective function:

$$f(x) = \sum_{i=1}^{c} S_i(x) \tag{3}$$

where $S_i(x)$ is 1 if clause i is satisfied by assignment x and 0 otherwise. A clause is satisfied if at least one of the literals it contains is true. Since our MAX-SAT instances are all satisfiable, we declared a MAX-SAT problem as solved as soon as a string x such that $f(x) = 1$ was generated by the EA.

Experiments have been conducted to evaluate the proposed Surrogate Model. To do this, we chose a variety of the well-known MAX-SAT problems, which we felt were difficult enough to demonstrate the characteristics and benefits of the method. Here, the MAX-SAT problems are considered as costly objective function. It should be noticed that the aim of the present experiments is not to show that GP-RBFN can be competitive on real-world problems with expensive objective functions, rather it is to show that GP-RBFN can be, in principle, applied to such cases and that it provides meaningful results when applied to well-studied model problems on a simple discrete space. Note that RBFN surrogates are known to be successful for approximating continuous space problems. In our experiments we consider a discrete problem space to have different characteristics compared to continuous spaces. Hence, this preliminary step is necessary to validate GP-RBFN.

TABLE II
Surrogate Parameters Setting

Setting	*Value*
Expensive Evaluation	$5n$
Initial sample size	2
GP Population	50
GP Generations	20
Crossover rate	0.7
Mutation rate	0.3

In order to evaluate the performance of the SMBO algorithm under different conditions of problem size and ruggedness, we use MAX-SAT of size $n = \{20, 25, 30, 35, 40\}$. For comparison purposes, we performed extensive empirical experimentation (10 * 5 * 5 runs in total) [5].

All MAX-SAT problems were generated using a standard generator called *sgen* which was developed to generate small and difficult benchmarks for the satisfiability problem [14]. Sgen has been used for the annual SAT competition and satisfies the input requirements of that competition [14].

Tables II & III reports the GP-RBFN surrogates settings. The reason of setting the size of the initial samples of data-points to two is to prove our prime hypothesis that the surrogate model is better than random sampling at suggesting promising solutions which are better than the known-points set as it makes rational use of the available known-points set to guess the next promising point. Since, the *fast* GA search is guided by a cheap surrogate evaluation, we have been generous with its search size, as illustrated in Table III, i.e., $10n \times 10n$, where n is the number of variables in the MAX SAT problem. GP-RBFN settings are shown in Table II. The first section (top) of Table II shows the number of evaluation of the RBFN whereas the second section (bottom) shows the used GP setting to optimise the RBFN. Regarding the GP setting, we used trail and error approach to find the best settings in a way that balances between good performance and computational costs.

As a reference of performance, we compared our results against (a) *standard RBFN Surrogate* to validate our approach against the standard surrogate model, (b) *standard GA search* to verify whether our approach can find better solutions using small number of evaluation than standard

[5] 10 independent runs, 5 different settings for the number of variables ($n = \{20, 25, 30, 35, 40\}$) used for the MAX-SAT problem, and 5 different approaches (our proposed approach: GP-RBNF against standard RBFN, standard GA, (1+1) ES and Random Search).

TABLE III
FAST GA PARAMETERS SETTING

Setting	*Value*
Fast GA Population	$10n$
Fast GA Generations	$10n$
Crossover rate	0.7
Mutation rate	0.3

TABLE IV
SETTINGS USED IN THE EXPERIMENTS

Operator	*Standard GA*	*1+1 ES*	*Random Search (Uniform Distribution)*
One-Point Mutation	30%	100%	N/A
Crossover	70%	0%	N/A
Tournament size	2	N/A	N/A
Population Size	n	1	$5n$
Generations	5	$5n$	1

(expensive) GA, (c) *(1+1) Evolutionary Strategies* (based on one-point mutation) to compare our search with a standard search algorithm and finally, (d) *Random search* to validate our main hypothesis that surrogate is better than random sampling as it makes rational use of the available information and point out the next promising points in the search space.

For a fair comparison, all algorithms search the given problem within exactly the same number of expensive evaluations, i.e., $5n$, as illustrated in Table IV. Standard GA searches the given problem using n population through 5 generations (thus $5n$ in total). We used tournament selection size 2. As for Standard RBFN surrogate we used the same settings as in the proposed model (see upper section in Table II and Table III). The radius parameter β, has been set to $1/D^2$ where D is the mean distance of the closest 50% neighbours from the known-points set.

V. RESULTS AND DISCUSSION

A. GP-RBFN vs. Competitors

In order to test how reliable is our approach in estimating fitness values without the need of evaluating them, we used the best-of-run solution in each run to compare the performance. Every time the surrogate uses the real fitness function, we count this as a real evaluation. The number of these evaluations is the real search. Thus, to make a fair comparison we give the same number of these real evaluations to all the algorithms used in this study. Note that, giving the algorithms in the comparison same number of expensive evaluations is similar as giving them the same amount of computational time to search the problem. In this paper, we preferred to make a fair comparison based on number of expensive evaluations rather than using the time because it is easier to control.

As mentioned previously, the aim is not to effectively solve the SAT problem itself, rather than to show that our approach can achieve better solutions using small number of evaluations than its competitors. We compared the performance (both in terms of finding the global optimum and the average of the best solutions) of our proposed approach (GP-RBFN) versus four different approaches (Standard RBFN, Standard GA, Random Search and (1+1) ES). Table V reports these results.

Our proposed approach, GP-RBFN surrogate, outperformed its competitors in four (out of five) instances of the MAX-SAT problem. For instance, if we focus our attention on the best fitness values, we can clearly see that GP-RBFN surrogate model achieved the best results (indicated in boldface in Table V) in most of the cases compared with the other four approaches. Only in one case (40 variables) GP-RBFN comes in second place after standard GA. If we further continue analysing these results, we can see that our approach was the only one to find the global optima in two (out of five) instances of the MAX-SAT problem (i.e., 20 and 30 variables). Global optima found by our approach are underlined in Table V. Notice also how our proposed approach suffers from less variations (i.e., standard deviation denoted by std.) in most of the cases compared to other four competitors. If we, now, turn our attention on the average of the best solutions (third column of Table V), we can further corroborate that our approach, GP-RBFN, remains as the best approach compared to the rest of the approaches used for comparison purposes. For instance, Standard RBFN and GA always come on the second place in all instances. This is not surprise because our approach has the advantage that GP allows to evolve β values that fits the problem, as explained in Section II.

We suspect the reason that our approach did not outperform its competitors in 40 variables may be because we over trained the model. Consequently, if the SM gains more diversity, the result could be better once the problem size is even larger. This aspect will be further investigated in future research.

To further verify the significance of our results, a Kolmogorov-Smirnov two-sample test [11] has been performed on the test-case results produced by the best evolved system in each run for all pairs of systems under test and for all five test cases. Table VI reports the *p-value* for the tests. As one can see, in all cases our system is statistically significantly superior to all its competitors at the standard 5% significance level except for GP-RBFN vs. GA when $N = 40$. Note that our experiments show that GA has outperformed GP-RBFN in terms of best achieved solution in this particular case. Here, the high p-value indicates that it is unlikely that GA will outperform GP-RBFN in future

TABLE VI
KOLMOGOROV-SMIRNOV TEST.

	GP-RBFN vs. Standard RBFN Surrogate	GP-RBFN vs. GA	GP-RBFN vs. (1+1) ES	GP-RBFN vs. Random Search
N = 20	0.031	0.031	1.89E-05	1.70E-04
N = 25	1.89E-05	1.89E-05	1.89E-05	1.89E-05
N = 30	0.031	1.70E-04	1.89E-05	1.89E-05
N = 35	0.0012	0.0012	1.89E-05	1.89E-05
N = 40	1.70E-04	0.111	1.70E-04	1.70E-04

TABLE V

SUMMARY OF 50 INDEPENDENT RUNS (10 RUNS FOR EACH MAX-SAT INSTANCE).

		GP-RBFN Surrogate			RBFN Surrogate			Standard GA			(1+1) ES			Random Search		
Vars.	Opt.	Avg.	Std	Best	Avg.	Std	Best	Avg.	Std	Best	Avg.	Std	Best	Avg.	Std	Best
N = 20	91	90.00	0.60	**91.00**	89.00	0.60	90.00	88.80	1.03	90.00	87.00	0.95	89.00	83.80	2.12	86.00
N = 25	60	59.00	0.00	**59.00**	58.10	0.29	58.00	57.60	0.76	58.00	56.30	0.74	57.00	51.30	3.86	57.00
N = 30	72	70.90	0.79	**72.00**	69.50	0.48	70.00	69.00	0.43	70.00	67.10	0.79	69.00	60.10	3.78	66.00
N = 35	84	82.50	0.48	**83.00**	81.00	0.60	82.00	81.00	0.60	82.00	77.70	0.61	79.00	72.00	4.37	78.00
N = 40	96	93.90	0.28	94.00	92.2	0.57	93.00	93.5	0.63	**95.00**	88.60	0.97	91.00	68.28	3.21	86.00

*__Bold__ indicates the best results among all the approaches, underlined indicates global optima.

runs.

VI. CONCLUSIONS

In this paper, we proposed a new Surrogate Model based on Genetic Programming and Radial Basis Function Networks, called *GP-RBFN Surrogate*. We have shown how it is possible to successfully estimate the fitness of potential solutions using SMs. Because SMs are approximation functions, they highly depend on how the system explores the search space. To do so in an unbiased way, we let our GP system to evolve RBFs expressions, so there is freedom in their parameters.

The proposed model uses GP to construct better RBF expressions for the RBFN surrogate model. GP receives standard RBF associated with its weights as a primitive and uses it to construct surrogate programs. We let the GP system to find the best radius parameter value for the RBF. Evolution modifies the radius parameter of the RBF and also combines it with constants values to control the magnitude of its output in such a way to minimise the surrogate's prediction errors. For the proposed model, a fitness function was designed to encourage GP to the evolve a small and accurate RBF expressions that are able to generalise on unseen data-points.

Our proposed approach was tested in one of the most studied NP-complete problems (MAX-SAT) to validate its effectiveness. Results clearly indicate that GP-RBFN successfully uses SMs to avoid evaluating a large number of candidate solutions while finding good solutions. We verified the significance of the results by applying the statistical Kolmogorov-Smirnov two-sample test. The test shows that the GP-RBFN surrogate is statistically significantly superior to all other four approaches at the standard 5% significance level. It is fair mentioning, that our approach suffers from one problem which is the extra computational costs of evolution candidate solutions at every time it suggests a new solution.

This research can be extended in several directions. Firstly, an attractive idea is to let GP decides the weights beside the radius parameters. Another direction is exploring the possibility of letting GP finding the best number of RBFs' centres to be used in such a way to improve the surrogate's prediction.

REFERENCES

[1] A. G. Bors. Introduction of the radial basis function (rbf) networks. Technical report, Department of Computer Science, University of York, UK, 2001.

[2] A. Forrester, A. Sóbester, and A. Keane. *Engineering design via surrogate modelling: a practical guide*, volume 226. Wiley, 2008.

[3] C. Harpham, W. Dawson, and R. Brown. A review of genetic algorithms applied to training radial basis function networks. *Neural Comput. Appl.*, 13:193–201, September 2004.

[4] H. H. Hoos and T. Stützle. Local search algorithms for sat: An empirical evaluation. *J. Autom. Reason.*, 24:421–481, May 2000.

[5] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput.*, 9(1):3–12, 2005.

[6] D. R. Jones. A taxonomy of global optimization methods based on response surfaces. *J. of Global Optimization*, 21:345–383, December 2001.

[7] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. The MIT Press, Cambridge, Massachusetts, 1992.

[8] Y. Lian, M. sing Liou, and A. Oyama. An enhanced evolutionary algorithm with a surrogate model. 2008.

[9] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff. Generalizing surrogate-assisted evolutionary computation. *Evolutionary Computation, IEEE Transactions on*, 14(3):329 –355, 2010.

[10] A. Moraglio and A. Kattan. Geometric generalisation of surrogate model based optimisation to combinatorial spaces. In *EvoCop*, Lecture Notes in Computer Science. Springer, 2011.

[11] J. A. Peacock. Two-dimensional goodness-of-fit testing in astronomy. *Royal Astronomical Society, Monthly Notices*, 202:615–627, 1983.

[12] R. Poli, W. B. Langdon, and N. F. McPhee. *A Field Guide to Genetic Programming*. Published via `http://lulu.com` and freely available at `http://www.gp-field-guide.org.uk`, 2008. (With contributions by J. R. Koza).

[13] B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *AAAI*, pages 440–446, 1992.

[14] vor Spence. Generator of benchmarks for the satisfiability problem (sgen), April 2011.