Using a Markov network as a surrogate fitness function in a genetic algorithm

Alexander E. I. Brownlee, Olivier Regnier-Coudert, John A. W. McCall and Stewart Massie

Abstract— **Surrogate models of fitness have been presented as a way of reducing the number of fitness evaluations required by an evolutionary algorithm. This is of particular interest with expensive fitness functions where the cost of building the model is outweighed by the saving of using fewer function evaluations.**

In this paper we show how a Markov network model can be used as a surrogate fitness function in a genetic algorithm. We demonstrate this applied to a number of well-known benchmark functions and although the results are good in terms of function evaluations the model-building overhead requires a substantially more expensive fitness function to be worthwhile. We move on to describe a fitness function for feature selection in Case-Based Reasoning, which is considerably more expensive than the other benchmark functions we used. We show that for this problem using the surrogate offers a significant decrease in total run time compared to a GA using the true fitness function.

I. Introduction

Reducing the algorithm run-time is one of the major aims of work in the field of evolutionary and population based computation. Often this means aiming to reduce the number of fitness evaluations taken to find an optimum, particularly in the case of applications which require a long time for each fitness evaluation. Such evaluations may involve a complex CPU process or some real-world measurement. Consequently there has been some interest in algorithms which construct a surrogate model of the fitness function which is then used for evaluation of some individuals in the population. Examples include [1]–[5]. Dependent on the problem, the resulting reduction in time taken by fitness evaluations can more than make up for the extra cost of building the surrogate. Our aim in this work is to build on our previous research to incorporate a probabilistic model of fitness as a surrogate in a genetic algorithm. We then demonstrate our algorithm's application on a number of benchmark functions and show how it provides a substantial speed gain when applied to the feature selection problem.

In previous work [6]–[13] we have described a number of approaches using undirected probabilistic graphical models (Markov networks) within a framework called Distribution Estimation Using Markov networks (DEUM), an Estimation of Distribution algorithm [14], [15].

With the Markov network approach model-building overhead is significant, particularly as the number of variables and interactions increases. While the number of function evaluations were significantly fewer than with other algorithms in our earlier work the time taken to build and sample the probabilistic model was large. We thus turn our attention to alternative ways in which we can use the information contained within the model.

In this paper we propose using the Markov fitness model (MFM) as a surrogate for the true fitness function in a genetic algorithm. This expands on our work in [8] where we described how the MFM can be used to predict the fitness of individuals in a population. The surrogate model is incorporated into a new algorithm which we call the Markov Fitness Model Genetic Algorithm or MFM-GA. First we give the performance results for MFM-GA applied to several known benchmarks: Onemax, Checkerboard, 2D Ising and Trap-5. These results are intended as a proof-of-concept for the use of fitness modelling to illustrate its effect on the GA - the run time required for model building renders the algorithm impractical for solving problems with low-cost fitness functions such as these. We then apply MFM-GA to a problem with a highly expensive fitness function: feature selection in Case-Based Reasoning (CBR). In this problem, the reduction in fitness evaluations resulting from use of the surrogate model outweighs the model build time and we are able to show a substantial reduction in overall run time.

The rest of this paper is structured as follows. In Section II we explore existing literature on fitness modelling in evolutionary and population based algorithms. In Section III we describe the MFM, its construction and its use in predicting fitness. Section IV presents a new algorithm, the MFM-GA. Section V describes our experimental approach, then in Section VI we apply MFM-GA to a number of standard benchmark functions. In Section VII we describe the feature selection problem in more detail and then give result of experiments comparing a GA to MFM-GA. Finally in Section VIII we conclude with our analysis, conclusions and possible future work.

II. Fitness modelling

Typically an evolutionary algorithm makes use of a fitness function to evaluate possible solutions. Many approaches have been proposed which construct a model of this function in order to improve the problem solving efficiency; whether by explicitly sampling the model to generate solutions as in the example of estimation of distribution algorithms [14], [15], or by using the model in a hybrid with other evolutionary techniques. One approach is to use a model of the fitness function as a surrogate for the true fitness function,

O. Regnier-Coudert, J. McCall and S. Massie are with the IDEAS Research Institute, Robert Gordon University, Aberdeen, UK (phone: +44 (0)1224 262472; email: o.regnier-coudert@rgu.ac.uk, jm,sm@comp.rgu.ac.uk.

A. Brownlee is with ODS-Petrodata Ltd, 2nd Floor The Exchange No. 1, Market Street, Aberdeen, AB11 5PJ. 01224 597800; email: sandy@whatisthegrid.co.uk.

using the model to assign fitness to some individuals. The aim is to reduce the number of fitness evaluations required by the algorithm while maintaining the quality of solutions produced. This is beneficial in situations where evaluations are computationally expensive. In general the algorithm only needs to build a close model of the fitness function to efficiently optimise - perfectly modelling the fitness function represents a complex problem in itself. Indeed, with some fitness functions an imperfect model gives a smoothing effect on the fitness landscape making the problem simpler for the search part of the algorithm [16]. Existing work using surrogate fitness models includes [1]–[5].

There is a wide and growing body of work using fitness modelling techniques in evolutionary algorithms, in addition to the surrogate fitness model approach. A fitness model may also be used to guide standard genetic operators such as crossover and mutation [17]–[21] or to provide useful information about the fitness function. An EA may use fitness inheritance [22]–[24] (passing of fitness values from parents to offspring) to reduce the number of fitness evaluations; although this does not involve the construction of an explicit fitness model. The Learnable Evolution Model (LEM) [25] incorporates machine learning to identify features distinguishing high and low fitness individuals, in effect modelling parts of the fitness function. The algorithm presented in [26] models a contour line on the fitness landscape between high and low fitness individuals which the authors describe as a special case of LEM. Models of fitness can also be constructed by using techniques such as artificial neural networks [19]. Miquélez et al [27] describe an algorithm which groups individuals of similar fitness into classes which are then passed to Bayesian classifiers that can be sampled to generate individuals of high fitness. Schmidt and Lipson [28] use coevolution as a means to generating fitness predictors efficiently. Polynomial regression or the fitting of a response surface has also been used to construct a model of fitness [29]. Our work bears some similarity to this; the Markov fitness model is in effect a response surface for the fitness function. One key difference is that our work concentrates on discrete fitness functions whereas Zhou et al [29] concentrate on continuous fitness functions.

A comprehensive review of fitness modelling in evolutionary computation conducted in 2005 is presented in [30].

III. Fitness modelling with Markov networks

Previous publications on DEUM [11], [13] described how a Markov network is used to model the distribution of energy across the set of variables in a bitstring encoded problem. The energy $U(x)$ of an individual follows a negative log relationship with its fitness $f(x)$, so the Markov network can be used as a model of the fitness function which we call the Markov Fitness Model (MFM). For a solution $x = \{x_1, x_2, ..., x_n\}$ we can derive an equation (1) for each individual in the population in which the 0 and 1 bit values have been interpreted as -1 and +1 respectively.

$$-ln(f(x)) = U(x) = \alpha_1 x_1 + \alpha_2 x_2 + ... \alpha_n x_n + \alpha_{12} x_1 x_2 + ... \quad (1)$$

In (1) each term represents a clique on the Markov network - this may be a 1-clique such as $\alpha_1 x_1$ or a 2-clique such as $\alpha_{12} x_1 x_2$. Additional terms may be added for higher order cliques in the same way as was done for 3-cliques in [11] and 5-cliques in the experiment on the trap-5 function in Section VI-D. The description of each experiment outlines the terms used in the model (synonymous with the Markov network structure). A system of equations is formed by substituting the variable values and fitness for each individual within a population into (1) and then using a least squares approach to estimate values for the parameters α. The structure (that is, the terms which are present in the model) and the set of α values completely define the MFM. This can then be sampled to generate new individuals or used to predict the fitness of individuals as in the work described in this paper.

Predicting the fitness of individuals is simply a reversal of the process used to estimate the α values. The bitstring of a given individual is encoded as before so that for each x_i, 0 is coded as -1 and 1 coded as +1. These values are substituted into the energy function to give a predicted energy $U(x)$ for the individual. Predicted fitness is then calculated as in (2).

$$f(x) = e^{-U(x)} \quad (2)$$

In this paper we use this fitness prediction capability in place of the fitness function in a genetic algorithm.

IV. Genetic Algorithm using MFM surrogate

We now describe a hybrid algorithm which uses the MFM model to provides a surrogate fitness function. We call this the MFM-GA:

1. Initialise randomly-generated population p_1
2. Evaluate fitnesses of p_1 using true fitness function
3. Select a subset σ_1 of the population p_1
4. Compute model parameters for MFM from σ_1
5. Run standard genetic algorithm:
   - 5.1. Select a subset σ_2 of p_1
   - 5.2. Generate new population p_2 using standard crossover and mutation operators on σ_2
   - 5.3. Evaluate p_2, using Markov network surrogate

At step 5.3., the individual with the highest predicted fitness was also evaluated using the true fitness function. This fitness was not used by the algorithm – it was simply to allow us to monitor the progress of the algorithm.

The important point to note is not the choice of a genetic algorithm but that the MFM provides surrogate fitness evaluations for the GA. This approach could be taken with any population-based algorithm using a bitstring representation.

V. Experimental approach

In order to prove that the MFM can model solution fitness for a given problem, we compare a GA that evaluates the solutions using their true fitness and a GA that evaluates the solutions using the MFM surrogate fitness.

TABLE I
EXPERIMENTAL PARAMETERS

GA Settings	Values
Number of Offspring	2
Selection	Tournament
Crossover	Std 2-point Crossover, rate=0.7
Mutation	Std Single-bit Mutation, rate=0.075
Elitism	2

TABLE II
POPULATION SIZE FOR EACH PROBLEM

Problem	Problem Size	GA Pop Size	MFM Pop Size
OneMax	180	100	19800
Checkerboard	100	1024	33000
2D Ising	64	1000	21120
Trap-5	60	1500	14520
FS - Sonar	61	100	671
FS - Vehicle	19	50	209

A. GA – Settings and population size

To efficiently show that the fitness obtained by the MFM can be used as a surrogate fitness, the GA with and without the MFM surrogate is set with the same parameters. As the focus is not on the performance of the algorithms but on how close the solutions they provide are, the same settings were used for all of the experiments and are shown in Table I. Population sizes for the GA are given under *GA Pop Size* in Table II. Population sizes for the OneMax, Checkerboard2D and Trap5 problems were set as in [31].

B. MFM – Population size and Selection rate

To build the MFM, a first population of solutions is needed. This population is evaluated using the true fitness function of the problem, and the Markov network coefficients are then calculated from this. As one objective of fitness modelling is to reduce computational cost, it is important to correctly set the population size used to build the model. Using a large population will result in a long model building time, and using a small population reduces the accuracy of the model. To correctly set the population size, we followed the approach presented in [8] that defines the optimum population size to use to build a MFM as $1.1N$, where N is the number of coefficients in the model.

In [8], truncation selection (there referred to as *top selection*) was shown to improve the quality of the models, thus we use truncation selection to choose the individuals used for estimating the model parameters. Experiments were run on a range of problems to define what proportion of the population should be selected to build the model. 1% truncation was used for the benchmark functions in Section VI and 10% for the feature selection in Section VII, giving a fair balance between model quality and number of true fitness evaluations required to build it. The start population size needed to respect the $1.1N$ ratio while using a 1% selection is thus $110N$ and with 10% selection, $11N$. The *MFM Pop Size* column in Table II shows the actual population sizes used on each problem. This also represents the total number of true fitness evaluations that will be performed to solve a problem using MFM-GA.

C. MFM – Structures

The fitness modelling capability of the MFM strongly depends on the structure that is given; this was explored to an extent in [8] and more fully in [6]. The structure represents the relationships between the different variables of a problem. Known structures were used for the benchmark function experiments and are specified with the description of the benchmark function for each experiment. For the experiments on the feature selection problem a structure learning process was used and its impact on running time and number of fitness evaluations is included in the results.

VI. MFM-GA ON BENCHMARK FUNCTIONS

In this section we explore the application of MFM-GA to a variety of standard benchmark problems. The goal of this is to demonstrate that the MFM can be used in place of the true fitness function on a variety of well-known low-cost benchmarks; we would not expect any speed benefit to be gained. The fitness functions are not computationally expensive and because for all of these problems the genetic algorithm can solve the problem in few enough function evaluations it does not take much time. Consequently we expect the time taken to build the model will exceed the time saved by reducing the number of true function evaluations.

We wish to see how the genetic algorithm's run is affected by using the surrogate model instead of the true fitness function so we present run-length distributions; on the x-axis of each is the number of true or surrogate fitness function calls and on the y-axis the cumulative number of runs of each algorithm (out of 1000) which found an optimum within that number of evaluations. We also give a statistical t-test comparison of the two distributions. If the surrogate successfully simulates the fitness function we would not expect these run-lengths to be substantially different as the GA is the same in both parts of the experiment – the key difference is that the MFM-GA is not using true fitness evaluations during the run.

We also give the mean wall-clock run times for each experiment (comparable because identical machines running no other task were used). This serves to illustrate the increased run-time caused by the model building step and provides motivation for moving on to apply the algorithms to the more computationally intensive feature selection problem in Section VII. For reference, the machines running the experiments had 2.66GHz Intel Core2-Duo CPUs (though the code is single-threaded).

In this section all the experiments allow the GA part of the algorithm to run until a known globally optimal solution is found or a cap on the number of evaluations is reached.

A. Onemax

First we present the results for the genetic algorithm and MFM-GA on the 180 bit onemax problem. This is one of the simplest and most commonly used benchmarks; in this

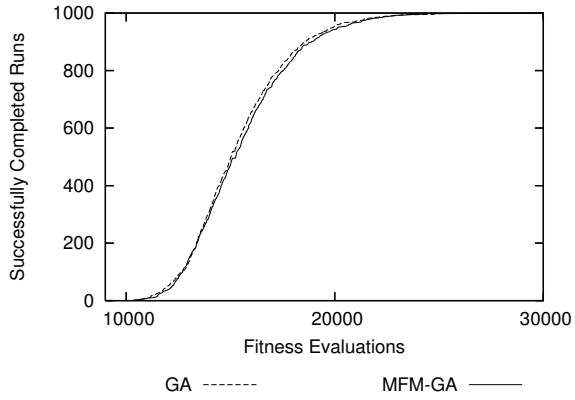

Fig. 1. Run-Length Distributions for 180-bit Onemax

TABLE III
RLD STATISTICS FOR 180-BIT ONEMAX

Mean GA True Evals	Mean MFM-GA Surrogate Evals	p-value
15403 (2480)	15544 (2538)	< 0.005

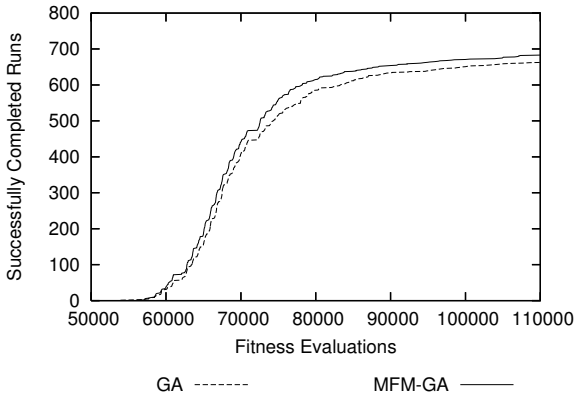

Fig. 2. Run-Length Distributions for 100-bit Checkerboard

TABLE IV
RLD STATISTICS FOR 100-BIT CHECKERBOARD

Mean GA True Evals	Mean MFM-GA Surrogate Evals	p-value
79818 (1096)	88200 (3115)	<0.005

problem the fitness is simply the number of bits with the value 1. Onemax is a classic univariate problem as there are no interactions between variables – each has an independent and equal contribution to the overall fitness. The optimal solution is all $x_i = 1$, with $f(x) = n$ with a single, global, optimum. There are no interactions between variables so the MFM includes only univariate terms.

The results in Figure 1 and Table III show the mean number of true and surrogate fitness evaluations taken by the GA and MFM-GA respectively with standard deviations in brackets. Both have similar performance in terms of true / surrogate fitness evaluations, although the p-value for the t-test between the two distributions is less than 0.005 showing the difference between the two mean run-lengths is significant. This means that MFM-GA evaluates the surrogate approximately 1% more often than the GA evaluates the fitness function. However, it can be seen that the genetic algorithm still successfully solves the problem in both cases and more importantly, the surrogate model can successfully be used in place of the true fitness function in this case.

As expected, with model building the total wall-clock time is considerably longer for 1000 runs of MFM-GA than for the GA: over 1000 runs each run of the GA experiment took a mean of 0.18s (std dev 0.18s), in contrast with 76.2s (std dev 0.18s) with MFM-GA. In addition, MFM-GA uses more evaluations of the true fitness function to build the model (given in Table II) than the GA requires for its run (showing Table III). The GA uses a mean of 15402 evaluations of the true fitness function; MFM-GA uses 19800 to build the surrogate model (though after the model is built no more evaluations are required).

B. Checkerboard

The checkerboard problem [14] introduces bivariate interactions. Chromosomes are a square number of bits in length; representing the rows of a *s* x *s* grid concatenated into one bitstring. The objective is to realise a grid with a checkerboard pattern of alternating 1s and 0s; thus each 1 should be surrounded by 0s and vice versa, not including corners. This gives us two, global, optima. Formally this is written as in (3) in which δ is 1 if $i = j$ and 0 otherwise.

$$\text{Maximise:} f(x) = 4(s-2)^2 - \sum_{i=2}^{s-1}\sum_{j=2}^{s-1}\left\{\begin{array}{r}\delta\left(x_{ij}, x_{i-1j}\right)\\ +\delta\left(x_{ij}, x_{i+1j}\right)\\ +\delta\left(x_{ij}, x_{ij-1}\right)\\ +\delta\left(x_{ij}, x_{ij+1}\right)\end{array}\right\} \tag{3}$$

The structure for the MFM includes both bivariate terms for the lattice and the univariate terms for the singleton variables. The results with the 100bit checkerboard ($s = 10$) problem in Figure 2 and Table IV are quite similar to those for onemax. The run-length distributions show MFM-GA taking significantly more surrogate function evaluations than the true evaluations required by the GA. In contrast with results for onemax, while the GA takes a mean of 79818 true evaluations, MFM-GA required 33000 evaluations (Table II) to build the surrogate model, after which no further true evaluations were used. With the more complex fitness function the run-time difference is less extreme than for onemax although the model building still adds considerable run time to the GA. Over 1000 runs of the GA each run took a mean of 1.26s (std dev 0.12s) and each run of MFM-GA took 34.2s (std dev 0.22s).

C. 2D Ising

The general Ising spin glass problem [32] is defined by an energy function over a set of spin variables $\sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_l\}$ and a set of coupling constants h and J as in (4).

$$H(\sigma) = -\sum_{i \in L} h_i \sigma_i - \sum_{i<j \in L} J_{ij}\sigma_i\sigma_j \tag{4}$$

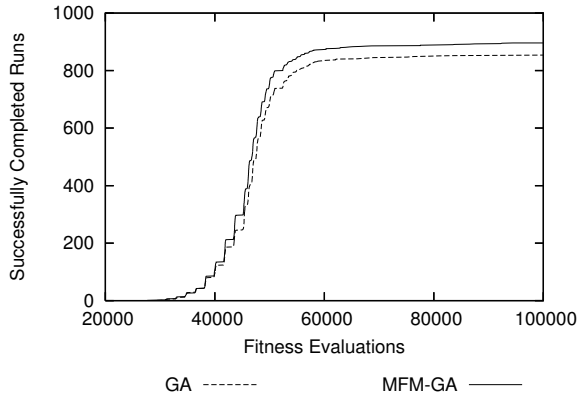

Fig. 3. Run-Length Distributions for 64-bit 2D Ising

TABLE V
RLD STATISTICS FOR 64-BIT 2D ISING

Mean GA True Evals	Mean MFM-GA Surrogate Evals	p-value
56500 (2282)	45729 (4904)	0.000

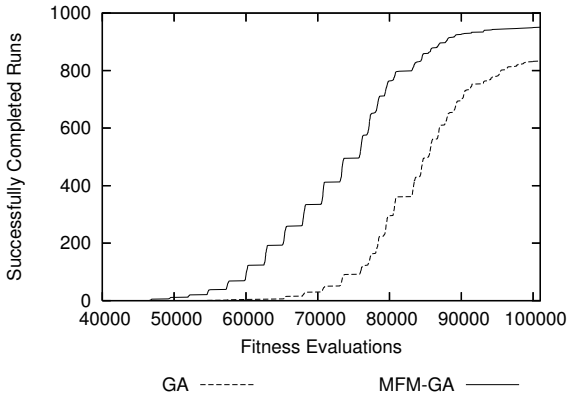

Fig. 4. Run-Length Distributions for 60-bit Trap-5

TABLE VI
RLD STATISTICS FOR 60-BIT TRAP-5

Mean GA True Evals	Mean MFM-GA Surrogate Evals	p-value
83241 (7228)	70839 (8122)	0.000

Here L is a lattice of n sites (in this experiment a 2D lattice). Each coupling constant h_i and J_{ij} relates to a single spin $\sigma_i$and pair of spins $\sigma_i$and σ_j respectively. Spin variables can be +1 or -1. With a binary encoding the coupling constants are restricted to +1 and -1 (in the general case this restriction is removed). The objective of the problem is to find a configuration of spins which minimises the energy H. The specific instances we used were produced by randomly sampling each coupling constant – the optimum for each was verified using the Spin Glass Ground Server[1] . Ising has been used for benchmarking EDAs, including [13], [33].

The structure for this problem is the same as that for checkerboard and the results, shown in Figure 3 and Table V are quite similar. We can see that MFM-GA completes using significantly fewer evaluations, and also has a higher success rate. In addition, the GA completed using a mean of 56500 true fitness evaluations, compared to 21120 evaluations required to build the surrogate for MFM-GA. However, again the model build time is considerable: the GA took a mean of 0.84s (std dev 0.24s) over 1000 runs whereas MFM-GA took a mean of 96.6s (std dev 0.22s).

D. Trap-5

The trap functions [34] are designed to deceive evolutionary algorithms into converging on a local optimum. This is particularly a problem for algorithms which do not consider interactions between variables. The trap function of order k is defined in (5). It computes fitness by dividing the bitstring into blocks of k bits.

$$f(x) = \sum_{i=1}^{n/k} trap_k(x_{b_i,1} + ... + x_{b_i,k}) \tag{5}$$

Each block $(x_{b_i,1}, +... + x_{b_i,k})$ gives a fitness, calculated as in (6), where u is the number of 1s in the block of k bits.

[1]http://www.informatik.uni-koeln.de/ls_juenger/research/sgs/sgs.html

Trap-5, which is the specific instance used in this experiment, has $k = 5$, $f_{high} = 5$ and $f_{low} = 4$.

$$trap_k(u) = \begin{cases} f_{high} & \text{if } \ u = k \\ f_{low} - u\frac{f_{low}}{k-1} & otherwise \end{cases} \tag{6}$$

The trap function is deceptive because as u increases, fitness decreases, which can lead the algorithm away from the global optimum. Algorithms which do find interactions should be able to determine that groups of k variables being switched to all 1 improves fitness. The structure of the MFM for this problem was the set of univariate terms for each bit, bivariate terms $x_i x_{i+1}, 1 \leq i \leq n$ representing the chain structure of the bitstring, and a term for each group of 5.

The results for MFM-GA and the GA are given in Figure 4 and Table VI. In contrast with the other problems, with the trap-5 problem we note an interesting difference which is worth highlighting; MFM-GA requires noticeably fewer surrogate evaluations when to locate the global optimum than true fitness evaluations required by the GA. We believe that this is a side-effect of the model detecting the information between variables. The model detects the higher fitness found in individuals with groups of 1s and ranks these higher than other individuals. As the model distributes energy across the variables based on fitness information from the whole population, it can give a much larger allocation of fitness to the optimal groups of 1s than small groups of 0s. This masks the deceptive component of the problem from the GA so it is able to more efficiently find a solution. Given that the trap function is designed to be difficult for a plain GA to solve it is surprising that its success rate is as high as 80%. We believe this is because the large population used for the GA (1500) means that it is likely each group of 5 bits will be set to 1 somewhere in the initial population, giving the algorithm a head-start. The GA required a mean of 83241 true fitness evaluations whereas MFM-GA required 14520 (Table II) to

build the surrogate after which it did not require further true fitness evaluations. The model build time for MFM-GA is again much longer than the time taken by additional fitness evaluations in the GA: 1.44s (std dev 0.12s) for the GA over 1000 runs compared to 24.6s (0.24s) for the MFM-GA.

VII. MFM-GA on Feature Selection

We now move on to the CBR feature selection fitness function. This requires a relatively large amount of CPU time for each function evaluation (1-2s for each evaluation). Consequently the large overhead required to build the MFM is, in these experiments, compensated by the reduction in the number of true fitness evaluations. First we explain the background of feature selection and how the fitness function is constructed before moving on to the experimental results.

A. Feature Selection in Case-Based Reasoning

Case-based reasoning (CBR) solves new problems based on the solutions of similar past problems. Cases describing the problem faced and the solution applied are stored in a case base for future retrieval when a similar new problem is encountered. CBR is often applied to classification tasks (e.g. for decision support or diagnostic problems) in which the problem is represented as a feature vector and the solution by a class label. In classification tasks a good feature is one that is predictive of the problem class on its own or in combination with other features. The problem is that not all features are important: some may be redundant; some may be irrelevant; and some can even be harmful to further analysis. Instance-based learners are very susceptible to irrelevant features and applying a more compact representation has been shown to successfully improve accuracy [35]. In addition, reducing the number of features increases speed of analysis and alleviates the problem that some learning algorithms break down with high dimensional data.

Feature selection can be categorised into filter and wrapper methods. Filters are data pre-processors that do not require feedback from the final learner, such as information gain or the Chi-squared score [36]. As a result they tend to be faster, scaling better to large datasets [37]. The wrapper approach uses feedback from the final learning algorithm to guide the search for a subset of features; examples include forward selection, backward elimination and GAs [38]. Generally feedback ensures wrappers select a better set of features tailored for the learning algorithm but has the disadvantage of being time consuming because feedback involves learner accuracy ascertained from cross-validation runs. On larger datasets the time taken by wrapper approaches tends to offset the improvement in results.

In this paper we introduce a hybrid approach. A solution to this problem is a bitstring where each bit represents a feature. The feature is selected for classsification if the bit is set to 1. In the other case, the feature is not selected. Initially a wrapper approach, taking feedback from a leave-one-out accuracy evaluation with a 3-Nearest Neighbour classifier, is used to generate a model of the fitness function. A filter approach is then applied to optimise the feature subset selection for the generated model. The aim is to gain the advantages of both the filter and wrapper approaches, namely: an efficient algorithm that can be applied to larger datasets than traditional wrapper methods; and selection of good feature subsets that give higher accuracies than filter methods. Two public domain classification datasets from the UCI ML repository [39] have been used to demonstrate the hybrid approach: Sonar has 60 features, 208 cases and a binary classification; while Vehicle has 18 features, 946 cases and 4 solution classes.

B. MFM Structure

Unlike with the functions in Section VI, the structure for the model can not be inferred from the definition of the fitness function. The feature selection problem uses datasets to build a case base and to evaluate the solutions and interactions between the variables depend on the dataset that is used. We used two datasets, Sonar and Vehicle, that both present different characteristics. Relationships between variables of the datasets are not defined and therefore, the exact structure of the problem is not known. Consequently, we ran a Structure Learning algorithm using Chi-Square independence tests on a population which had been evaluated using the true fitness function. This algorithm was the same as that used in DEUM-χ^2 [7]. Using a threshold of 10, the Chi-Square algorithm discovered 16 and 7 bivariate interactions in the Sonar and Vehicle datasets respectively. These were added to the full set of univariate terms and then set as the structure of the MFM before running the GA component of the algorithm.

As the results show, this structure learning process took a considerable time to run - 88-90% of the total run time. To separate structure learning times from optimisation times we ran the experiments with MFM-GA twice; once including the structure learning step, and once having already run the structure learning (as if we had a ready-known structure).

C. Experimental Procedure

Experiments on the feature selection problems were run in two steps. First, to assess the quality of the model, recording both modelled and true fitness was needed. This resulted in running the MFM-GA while evaluating in parallel the solutions generated using the true fitness. These results were compared with those of a GA running on the true fitness function only. As the run time required for this fitness function is much longer, instead of 1000 runs as before both algorithms were run 30 times and the solutions compared.

The second objective was to measure the time gained by using a surrogate fitness in place of the true fitness. To do so, MFM-GA and the GA were each run 30 times. The time taken for a run was measured, excluding the time needed to build the case base at the start (common to both). For this second set of experiments, no extra measurements that could have caused extra time were taken for either algorithm.

TABLE VII
EXPERIMENTAL RESULTS ON THE FEATURE SELECTION PROBLEMS

		Sonar		Vehicle	
Algorithm	Str Learning	Best Fitness (SD)	Time (SD)	Best Fitness (SD)	Time (SD)
GA	-	0.95 (0.06)	804 (0.042)	0.76 (0.04)	6408 (0.138)
MFMGA	Yes	0.92 (0.07)	672 (0.84)	0.73 (0.04)	1542 (0.132)
MFMGA	No	0.92 (0.04)	66 (0.012)	0.73 (0)	180 (0.024)

D. Results

In Table VII we show results for the three algorithms - GA and MFM-GA with univariate and bivariate structures - on the feature selection problem for the two case bases. The *Algorithm* column shows which algorithm each row refers to. *Str Learning* indicates whether the structure learning step was included. For each case base we have two columns: mean best fitness found over 30 runs (with standard deviation *SD*) and mean run time in seconds with standard deviation *SD*.

The results show that the MFM-GA converged on a lower fitness than the GA. An unpaired t-test between the two closest sets of results showed that the difference between the fitness found with the Sonar case base is not statistically significant ($P = 0.0799$) but the difference for the Vehicle case base ($P = 0.0052$) is. From a CBR perspective there is not one optimal solution as the feature selection problem is a balance between run time and solution quality. However, the best fitness results obtained from GA and MFM-GA are both superior to those obtained by the filter selection technique: information gain [36]. The fitness obtained when selecting the best feature subset using information gain ranking is 0.87 for sonar and 0.72 for vehicle.

The key result from these experiments is that both algorithms incorporating the MFM take less time to converge than the GA. It can be seen that the mean run-time for MFM-GA is 132s shorter than GA on Sonar and 4866s shorter on Vehicle; a substantial improvement. The difference between these mean run times is highly statistically significant ($P < 0.0001$ for both case bases). Most importantly, if the reduction in run time carries over to feature selection on larger case bases this may mean that including the MFM makes the genetic algorithm a practical option for finding solutions where it is currently infeasible.

We can also see that the structure learning stage accounts for a considerable proportion of the run time. Consequently we can say that if the structure can be determined more efficiently, or is already known for the problem, the GA incorporating the MFM as a surrogate fitness model has a good chance of offering a large performance improvement.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper we have shown that the Markov fitness model can be used as a surrogate model of fitness in a genetic algorithm. We have proposed an algorithm which incorporates the model - the MFM-GA - and have demonstrated its performance when applied to a variety of standard benchmark functions. We have seen that on the simple benchmark functions, while the algorithm works and is capable of finding a similar optimum to the standard GA with a reduction in true fitness evaluations, its run-time is impractically long due to the cost of model-building.

An interesting result was seen with the experiments applying MFM-GA to the Trap-5 function. The MFM-GA is given an advantage because model includes interactions. This may be similar to the smoothing effect described in [16].

We have also demonstrated MFM-GA on a CBR feature selection problem using two different case-bases. With this application, run time is considerably reduced by the use of the surrogate to reduce the number of fitness evaluations with the trade-off being a lower fitness of solution found. We have also shown that if the model can be supplied with a structure instead of having to learn it there is the potential for a considerable improvement in running time. This finding is important as often it is the total overall running time which is important in an application rather than finding the globally optimal solution.

There are a number of potential avenues for future research arising from this work. Initially it will be useful to find further expensive fitness functions like feature selection to determine whether similar improvements in run-time can be achieved. It will also be interesting to try improving the speed of the algorithm by using alternative structure learning approaches and by using the MFM as a surrogate for other population-based algorithms in place of the GA.

Finally an important issue with this approach is that the model is only constructed once. The obvious flaw with this is that the model will not necessarily be perfect and does not benefit from new information coming from individuals later on in the evolutionary process. The next step would be to monitor the MFM's fitness prediction capability using the fitness prediction correlation measure described in [8], and when this drops below a certain level either rebuild the model using the new population or find a way to improve the model incrementally. This will require more evaluations of the true fitness function but with the potential of an improvement in the fitness of the final solution found. Study of this approach would allow a finer balance to be made between faster run-time and better solution quality.

In summary, the results presented here and possibilities for future work show much potential for the use of the Markov fitness model as a surrogate for expensive fitness functions.

REFERENCES

[1] J. Zhang and A. C. Sanderson, *Adaptive Differential Evolution*. Springer, 2009, ch. Surrogate Model-Based Differential Evolution, pp.

83–93.

[2] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, "Generalizing surrogate-assisted evolutionary computation," *IEEE Transactions on Evolutionary Computation*, 2008.

[3] X. Llorà, K. Sastry, T.-L. Yu, and D. E. Goldberg, "Do not match, inherit: fitness surrogates for genetics-based machine learning techniques," in *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*. New York, NY, USA: ACM, 2007, pp. 1798–1805.

[4] K. Sastry, C. Lima, and D. E. Goldberg, "Evaluation relaxation using substructural information and linear estimation," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2006)*. New York, NY, USA: ACM Press, 2006, pp. 419–426.

[5] Y. S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong, "Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems," in *In Knowledge Incorporation in Evolutionary Computation*. Springer Verlag, 2004, pp. 307–332.

[6] A. E. I. Brownlee, "Multivariate Markov Networks for Fitness Modelling in an Estimation of Distribution Algorithm," Ph.D. dissertation, Robert Gordon University, Aberdeen, May 2009.

[7] A. E. I. Brownlee, J. A. W. McCall, S. K. Shakya, and Q. Zhang, "Structure Learning and Optimisation in a Markov-network based Estimation of Distribution Algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009)*. Trondheim, Norway: IEEE Press, 2009.

[8] A. E. I. Brownlee, J. A. W. McCall, Q. Zhang, and D. Brown, "Approaches to Selection and their effect on Fitness Modeling in an Estimation of Distribution Algorithm," in *Proceedings of the IEEE World Congress on Computational Intelligence (CEC 2008)*. Hong Kong, China: IEEE Press, 2008.

[9] S. K. Shakya, A. E. I. Brownlee, J. A. W. McCall, F. Fournier, and G. Owusu, "A fully multivariate DEUM algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2009)*. IEEE Press, 2009.

[10] S. K. Shakya and J. A. W. McCall, "Optimization by estimation of distribution with DEUM framework based on Markov random fields," *International Journal of Automation and Computing*, vol. 4, no. 3, pp. 262–272, 2007.

[11] A. E. I. Brownlee, J. A. W. McCall, and D. F. Brown, "Solving the MAXSAT problem using a multivariate EDA based on Markov networks," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2007) (Late Breaking Papers)*. New York, NY, USA: ACM Press, 2007, pp. 2423–2428.

[12] S. Shakya, F. Oliveira, and G. Owusu, "An application of EDA and GA to dynamic pricing," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2007)*, 2007, pp. 585–592.

[13] S. K. Shakya, J. A. W. McCall, and D. F. Brown, "Solving the Ising spin glass problem using a bivariate EDA based on Markov random fields," in *Proceedings of the IEEE World Congress on Computational Intelligence (CEC 2006)*. IEEE Press, 16-21 July 2006 2006.

[14] P. Larrañaga and J. A. Lozano, *Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation*. Boston: Kluwer Academic Publishers, 2002.

[15] J. A. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, *Towards a New Evolutionary Computation: Advances on Estimation of Distribution Algorithms (Studies in Fuzziness and Soft Computing)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[16] Z. Zhou, Y. S. Ong, M. H. Lim, and B. S. Lee, "Memetic algorithm using multi-surrogates for computationally expensive optimization problems," *Soft Comput.*, vol. 11, no. 10, pp. 957–971, 2007.

[17] C. F. Lima, K. Sastry, D. E. Goldberg, and F. G. Lobo, "Combining competent crossover and mutation operators: a probabilistic model building approach," in *Proceedings of the 2005 conference on Genetic and evolutionary computation (GECCO 2005)*. New York, NY, USA: ACM, 2005, pp. 735–742.

[18] K. Abboud and M. Schoenauer, "Surrogate Deterministic Mutation: Preliminary Results," in *Selected Papers from the 5th European Conference on Artificial Evolution*. London, UK: Springer-Verlag, 2002, pp. 104–116.

[19] Y. Jin and B. Sendhoff, "Reducing fitness evaluations using clustering techniques and neural network ensembles," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2004)*. Seattle, WA: Springer, 2004, pp. 688–699.

[20] K. Rasheed, S. Vattam, and X. Ni, "Comparison of methods for using reduced models to speed up design optimization," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2002)*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2002, pp. 1180–1187.

[21] Q. Zhang and J. Sun, "Iterated local search with guided mutation," in *Proceedings of the IEEE World Congress on Computational Intelligence (CEC 2006)*. IEEE Press, 2006, pp. 924 – 929.

[22] J.-H. Chen, D. Goldberg, S.-Y.Ho, and K.Sastry, "Fitness inheritance in multiobjective optimization," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2002)*. ACM Press, 2002, pp. 319–326.

[23] M. Pelikan and K. Sastry, "Fitness inheritance in the Bayesian optimization algorithm," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2004)*, 2004, pp. 48–59.

[24] R. E. Smith, B. A. Dike, and S. A. Stegmann, "Fitness inheritance in genetic algorithms," in *SAC '95: Proceedings of the 1995 ACM symposium on Applied computing*. New York, NY, USA: ACM Press, 1995, pp. 345–350.

[25] R. S. Michalski, "Learnable evolution model: Evolutionary processes guided by machine learning," *Machine Learning*, vol. 38, no. 1-2, pp. 9–40, 2000.

[26] P. Pošík and V. Franc, "Estimation of fitness landscape contours in EAs," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2007)*, D. Thierens, Ed., vol. 1. New York, NY, USA: ACM Press, 2007, pp. 562–569, note: ISBN 978-1-59593-697-4.

[27] T. Miquélez, E. Bengoetxea, and P. Larrañaga, "Evolutionary computation based on Bayesian classifiers," *International Journal of Applied Mathematics and Computer Science*, vol. 14, no. 3, pp. 101–115, 2004.

[28] M. D. Schmidt and H. Lipson, "Coevolution of fitness predictors," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 6, pp. 736–749, December 2008.

[29] Z. Zhou, Y. S. Ong, M. H. Nguyen, and D. Lim, "A study on polynomial regression and Gaussian process global surrogate model in hierarchical surrogate-assisted evolutionary algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2005)*, vol. 3, Sept. 2005, pp. 2832–2839 Vol. 3.

[30] Y. Jin, "A comprehensive survey of fitness approximation in evolutionary computation," *Soft Computing*, vol. 9, no. 1, pp. 3–12, 2005.

[31] S. K. Shakya, J. A. W. McCall, and D. F. Brown, "Using a Markov network model in a univariate EDA: an empirical cost-benefit analysis," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2005)*. ACM Press, 25-29 June 2005, pp. 727–734.

[32] R. Kindermann and J. L. Snell, *Markov Random Fields and their Applications*. Providence, RI: American Mathematical Society, 1980.

[33] M. Pelikan and D. Goldberg, "Hierarchical BOA solves Ising spin glasses and MAXSAT," in *Proceedings of the Genetic and Evolutionary Computation COnference (GECCO 2003)*. Springer-Verlag, 2003 2003, pp. 1271–1282.

[34] M. Pelikan, "Bayesian optimization algorithm: from single level to hierarchy," Ph.D. dissertation, University of Illinois at Urbana-Champaign, Urbana, IL, 2002.

[35] G. H. John, R. Kohavi, and K. Pfleger, "Irrelevant features and the subset selection problem," in *ICML 94: Proceedings of the Eleventh International Conference on Machine Learning*. Morgan Kaufmann Publishers Inc., 2004, p. 121129.

[36] Y. Yang and J. O. Pedersen, "A comparative study on feature selection in text categorization," in *ICML '97: Proceedings of the Fourteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1997, pp. 412–420.

[37] N. Wiratunga, I. Koychev, and S. Massie, "Feature selection and generalisation for retrieval of textual cases," in *In Proceedings of 7th European Conference on Case-Based Reasoning*. Springer-Verlag, 2004, pp. 806–820.

[38] S. Das, "Filters, wrappers and a boosting-based hybrid for feature selection," in *ICML '01: Proceedings of the Eighteenth International Conference on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2001, pp. 74–81.

[39] A. Asuncion and D. Newman, "UCI machine learning repository," 2007. [Online]. Available: http://www.ics.uci.edu/∼mlearn/MLRepository.html