

Predicting Stochastic Search Algorithm Performance using Landscape State Machines

William Rowe, David Corne and Joshua Knowles

Abstract— A Landscape State Machine (LSM) is a Markov model describing the transition probabilities between the fitness ‘levels’ of an optimization problem, when a given neighbourhood (or mutation) operator is applied. Although most optimization problems cannot be modeled precisely by an LSM, an *approximate* LSM can always be constructed by sampling, and can be used, subsequently, in place of real fitness evaluations in order to model the performance of any search algorithm using the given neighbourhood operator. In this paper, we provide empirical evidence that (a) LSMs constructed by simulated annealing-based sampling of a problem landscape make accurate models in few evaluations; (b) LSMs can accurately rank the performance of diverse algorithms including EAs with/without niching and SA; (c) the LSM approach works on diverse problems from MAX-SAT to NKp; (d) convergence of the LSM can be used as a guide to stopping the sampling phase; and, (e) a single LSM constructed using a low mutation-rate sample is sufficient to accurately rank the performance of search algorithms run at multiples of this mutation rate.

I. INTRODUCTION

The need for tuning of search algorithms to achieve good performance on specific problems has been understood by black-box optimization practitioners for many years and has been highlighted by the No Free Lunch theorems for optimization [16]. In practice, the tuning problem can be addressed using several different approaches:

- **‘Good-old Understanding’** The nature of the optimization problem and previous experiments with different algorithms are analysed thoughtfully, to support future design of specific representations, operators or search strategies (e.g. [15]).
- **Mathematical Models** The search algorithm and problem are modeled mathematically, resulting in theoretical performance curves that relate parameters like population size and mutation rate or type to best-of-population fitness (e.g. [14], [7]).
- **Landscape Statistics** Measures of problem difficulty (e.g. epistasis variance, fitness-distance correlation), made by empirical sampling of the problem landscape are used to predict the probable relative performance of different search algorithms, e.g. a hillclimber versus a genetic algorithm (see [13] for an overview).
- **Empirical Testing and Statistical Analysis** Several different algorithms are run repeatedly on the problem

and performance statistics are collected and analysed to determine the ‘winner’ (e.g., see [3]).

- **(Self-) Adaptive Parameter Tuning Algorithms** These adapt their search behaviour during run-time in response to search progress or other measures (e.g., see [6]).
- **Landscape Modelling** A model of the landscape of the optimization problem is constructed, and algorithms are tested empirically (off-line) on the model, instead of the real problem [5].

The choice of approach(es) to take depends on several factors including how much is known or knowable *a priori* about the problem (i.e. how black-box it really is), what kind of performance targets or guarantees one is aiming for, how expensive fitness¹ evaluations are, and how much time and effort can be afforded for the tuning phase. Mathematical models and ‘good-old understanding’ usually demand significant knowledge of the problem structure, while landscape statistics and empirical testing may rely on an extensive number of fitness evaluations. Adaptive parameter tuning methods tend to be the cheaper alternatives, when neither analysis nor evaluation overhead can be afforded.

Landscape modelling is a potentially promising new approach to algorithm tuning, particularly for problems where evaluation overhead is expensive and little is known about the problem structure (e.g. directed evolution, see section IV). In this approach, the fitness evaluation overhead of empirical testing is reduced by using a model of the problem for tuning the algorithms. Straightforwardly, the model could be any compact approximation of the mapping from decision space to fitness space (e.g. provided by a suitably trained neural network) but the construction of this mapping from a sample of the optimization problem by conventional means may be highly fraught. Corne et al. [5] have proposed an alternative way of modelling search landscapes — a landscape state machine (LSM) — which abstracts completely away from the decision space and only models transitions between fitness intervals. The model is very compact and can easily be constructed empirically from samples to allow subsequent tuning of algorithms at very little computational overhead. The state transition probabilities in an LSM are closely related to the ‘search kernel’ of Altenberg [1] but, whereas the latter rests on the assumption of an invertible fitness function, the LSM method was shown to estimate algorithm performance even when many different genotypes map to

William Rowe and Joshua Knowles are with the School of Chemistry, the University of Manchester, United Kingdom, (phone: +44 (0)161 3064450; email: J.Knowles@manchester.ac.uk).

David Corne is with the School of Mathematical and Computer Sciences, Heriot-Watt University, United Kingdom (email: dwcorne@macs.hw.ac.uk).

¹We use the term ‘fitness’ in the sense of objective function value, not reproductive success, throughout this paper.

each fitness interval used in the model [5].

In this paper, we build on [5] to further our understanding of LSMs, particularly their accuracy for predicting search algorithm performance, and their construction overhead in terms of real fitness evaluations used. In the next section, we first briefly describe LSMs and summarise the empirical sampling results of [5]. Next, details of our extended empirical study and our experimental set-up are given. Section III presents the results of the study, Section IV discusses our findings and Section V concludes.

II. LSM METHODOLOGY

A. LSMs and basic sampling

A landscape state machine describes a search landscape as a series of states S , each representing a group of equivalent fitness values. Associated with these states is a transition matrix T , which represents the transition probabilities between the series of states when an operator M is applied. This can be visualized easily using the theoretically derived LSM described in [5], representing a Max-Ones problem in which the candidate solution is a binary string of length $L = 5$, and the parameter which is to be maximized (the fitness value) is the number of 1s it contains (see Figure 1).

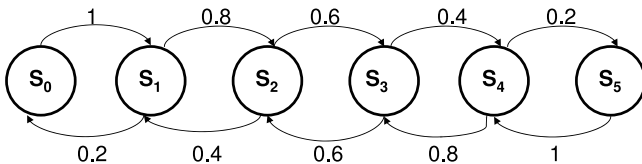


Fig. 1. Landscape state machine for Max-Ones problem, where the number of states is 6, assuming single bit flip mutations. Reproduced from [5].

If the length of the chromosome is five, there are six possible fitness values. These fitness values can be modeled as states (0-5). If a genetic algorithm operating on the chromosome produces one bit (allele) mutation per chromosome per generation, the transition probabilities between these states can be easily calculated. For example if a chromosome has a fitness of zero it can be assumed that there is a one hundred percent chance that a bit flip (0 to 1) mutation to the sequence will produce a fitness of one. The transition probability from state zero to state one is therefore one. It is straightforward to determine what the transition probabilities will be with a simple Max-Ones problem, and the model is also precise (i.e. a search algorithm using the operator M will have the same performance distribution on the LSM as on the real problem). However, for many other problems, no precise LSM exists because fitness is not Markovian. Nonetheless, an *approximate* LSM may still be derived, and used to test algorithm performance. In [5], a method of deriving an approximate LSM from a sample of a search landscape was described and tested.

For an NK -landscape problem a two-hundred state LSM was constructed, with equally spaced boundaries between the highest and lowest fitness states encountered by the

sampling algorithm. The sampling algorithm was based on a mutation-only evolutionary algorithm with elitism (the fittest individual is automatically selected into the next generation), a population size of two hundred and a tournament size of 1 (random selection), run for either 50,000 or 100,000 evaluations. Each parent and resultant child fitness state was recorded, and used to generate the transition matrix. The states of the initial population in the sampling algorithm were stored, to generate the initial states in the algorithms subsequently tested on this LSM.

This LSM was then used to assess the performance of ten different evolutionary algorithms. The rank order of the performance of these algorithms run on the landscape was compared to the rank order produced when assessing the performance of these algorithms on the real landscape. On two different NK landscapes with $N = 1000$ and $K = 2$ and $K = 5$, respectively, an LSM constructed from 100,000 evaluations delivered a promising performance at predicting the ranking of the ten EAs.

B. Simulated annealing for sampling

In [5], algorithms that reached the highest state when run on the LSM were distinguished, in terms of ranking their performance, by the number of evaluations required to reach this state. This form of discrimination may lack accuracy, and may also distort results in favour of algorithms that converge quickly, relative to an algorithm with more controlled convergence such as a simulated annealing algorithm [12].

As the sampling algorithm employed a low selection pressure, the actual range of fitness values recorded in the LSM would have been limited. This offers the potential of generating an LSM with high predictive quality with fewer evaluations than with the LSM sampling algorithm described previously. A simulated annealing-based sampling algorithm will initially allow random sampling, with increasing selection pressure as the algorithm progresses. This should stretch the range of fitness values recorded in the LSM. Algorithms subsequently run on the LSM should find it harder to reach the higher states, and reduce the reliance on discriminating between algorithms based on the number of evaluations required to reach the highest state.

We compare the performance of an LSM constructed using the method described by Corne et al., with one inferred from a run of simulated annealing. In the simulated annealing algorithm, the probability of accepting a mutation is determined by the Boltzmann acceptance criterion:

$$P(\text{accept}) = \begin{cases} 1, & \text{if } \Delta F \geq 0 \\ e^{(\Delta F/T)}, & \text{if } \Delta F < 0, \end{cases} \quad (1)$$

where $P(\text{accept})$ is the probability of accepting the transition, ΔF is the change in the objective function associated with the transition, and T is the temperature of the system.

The start and final temperatures are determined by values recorded in the construction of the LSM with the original sampling algorithm. For this algorithm, the initial value of T is set so that the the probability of accepting a mutation

from the highest state to the lowest state in the LSM is one percent.

To determine the cooling rate, the final temperature is set based on the final probability of acceptance. Here, the probability of accepting a mutation from the highest state to the second highest state is set at one percent.

C. Diverse problem types

In order to evaluate how robust the LSM tuning methodology is to diverse problem types, we extend the range of problems considered in the Corne et al study.

In all, four different types of binary-based optimization problems are investigated, giving six instances in all:

- 1) NK -landscape [10] where $N = 1000$ and $K = 2$.
- 2) NK -landscape where $N = 1000$ and $K = 5$.
- 3) NKp -landscape [2] where $N = 1000$, $K = 2$ and $p = 0.99$.
- 4) NKp -landscape where $N = 1000$, $K = 5$ and $p = 0.99$.
- 5) Flat Graph Coloring problem [9] with 100 vertices and 239 edges.
- 6) Blocks World problem [11] with 1087 variables and 13772 clauses.

The latter two problems have both been transformed into MAX-SAT form. Note: we do not show results for all instances for all of our experiments; a selection of results only is given.

D. Search algorithms including EAs with niching

In addition to ten algorithms similar to those originally tested by Corne *et al* (algorithms A–J)², eleven further algorithms are tested. Ten of these are evolutionary algorithms with local breeding on a one-dimensional grid [4]. The purpose of including these algorithms is to investigate if LSMs can still predict the performance of algorithms that explicitly try to maintain diversity in the decision space. This may be particularly challenging because an LSM does not encode any decision space information and only ‘lumps’ all equivalent fitness values together. Finally, the simulated annealing algorithm that we use to construct some of our LSMs is also included in the set of algorithms to test subsequently.

- A. Population size 10, tournament size 1 (random selection), 500 generations.
- B. Population size 10, tournament size 2, 500 generations.
- C-F. Population size 20. 235 generations, tournament sizes 1, 2, 3 and 5 respectively.
- G-J. Population size 50. 91 generations, tournament sizes 3, 5, 7 and 10 respectively.
- Niche A–J. As for A–J above, respectively, but with local breeding.
- SA. The simulated annealing algorithm — the same one as used for sampling.

²Our algorithms A–J do not use elitism (protection of the best individual) where Corne et al’s did.

In all algorithms and all problems considered in this study, a binary flip mutation probability of $1/L$ is used, where L is the length of the binary chromosome.

E. Measuring prediction accuracy with a rank-order distance

Our experiments measure the accuracy that the LSM achieves at ranking the performance of the 21 algorithms described above, on each problem. To do this for one problem, we first measure the performance of each algorithm on the *real* problem in terms of its best-objective value in each of 10 independent runs. A Fisher permutation test is next used to determine statistically significant differences between the algorithms’ performance (an alpha value of $0.05/21$ is used to correct for multiple testing). The algorithms are then ranked according to these differences, with tied algorithms sharing the sum of the rank values between them. The same ranking procedure is then carried out using the LSM in place of the problem (with the exception that 30 independent algorithm runs are used). Finally, the ‘distance’ R between the ranking of the algorithms on the real problem and on the LSM is computed, and the significance of this result is estimated using a Fisher permutation test.

F. LSM convergence/accuracy

The value of the LSM technique depends on the number of evaluations required to construct a reasonably accurate LSM. Corne *et al* noted in their study that they required more evaluations to produce a suitable LSM based on the NK -landscape, where $K = 10$ and $N = 1000$, than the landscape where $K = 5$ and $N = 1000$. This reflects that there is unlikely to be a fixed number of evaluations required to generate LSMs for all problems. In practice, we would like to stop sampling from the real landscape once the LSM has reached a certain accuracy or it has, in any case, converged. To accommodate this, the LSM can be constructed incrementally, checking the change in the transition matrix with the addition of new sample evaluations. Once the transition matrix has reached a point where the addition of new evaluations fails to contribute significantly to its overall structure, the predictive quality of the LSM should not improve further and more evaluations would be wasteful.

G. Mutation rate tuning using LSMs

Tuning of mutation rates is an important aspect of search algorithm tuning. We investigate whether it is possible to modify the way the algorithms tested interact with the LSM to predict the effect of higher mutation rates on performance. The LSMs described previously predict the performance of algorithms at the same mutation rate of the sampling algorithms used to construct them. To predict the performance of the algorithms at higher mutation rate, solutions are simply allowed to make several transitions before their new fitness is calculated and selection (or an acceptance criterion) is applied.

III. EXPERIMENTAL RESULTS

A. LSM construction by EA sampling vs SA sampling

The predictive quality of LSMs based on the simulated annealing and EA sampling algorithm from [5] were compared. The LSMs were based on 100,000 evaluations on problems 1, 2 and 3. The construction of the simulated annealing-based LSM was made up of 20 runs of the algorithm for 5000 evaluations. To compare the rankings of the algorithms on a real problem and its LSM, 10 independent runs of 100,000 evaluations were conducted on the real problem and 30 runs of 100,000 evaluations on the LSM, for each algorithm. Note: the initial populations of algorithms run on an LSM were drawn uniformly at random from the initial ‘population’ used in the construction of the LSM itself. In the case of the LSM constructed by SA, the twenty initial starting states used by the SA count as the population.

TABLE I
RANK ORDER OF PERFORMANCE OF ALGORITHMS ON *NK*-LANDSCAPE, FLAT GRAPH COLOURING AND BLOCKS WORLD OPTIMIZATION PROBLEMS. RESULTS BASED ON STATISTICAL DIFFERENCES IN BEST FITNESS ACHIEVED FROM TEN INDEPENDENT RUNS; BEST PERFORMING ALGORITHM IS LISTED FIRST.

<i>NK</i>		FGC		BW	
1	SA	1.5	niche J	2	niche F
2.5	niche J	1.5	SA	2	niche J
2.5	niche F	3.5	F	2	SA
4.5	niche I	3.5	niche F	4	niche I
4.5	F	5	niche I	6	niche B
6	niche E	7.5	niche H	6	F
8	J	7.5	E	6	niche E
8	E	7.5	I	8	B
8	niche H	7.5	J	9.5	E
10	B	10	niche E	9.5	niche H
11.5	niche B	11.5	H	11	niche D
11.5	I	11.5	B	13	niche G
13	H	14	niche G	13	J
14.5	D	14	niche B	13	D
14.5	niche D	14	D	15	I
16	niche G	16.5	G	16	H
17	G	16.5	niche D	17	G
19	niche C	19.5	niche A	19.5	niche A
19	A	19.5	C	19.5	C
19	niche A	19.5	A	19.5	A
21	C	19.5	niche C	19.5	niche C

The results from the comparison between the performance of the 21 algorithms on the LSMs constructed from the original sampling algorithms on the *NK*-landscapes, the Blocks World and Flat Graph Coloring problems produced a rank order distance of 21, 46, and 28, respectively (see Table II). Although this level of prediction is good, it is noticeable that the LSM underestimates the performance of simulated annealing on the real landscape. This is especially problematic as the simulated annealing algorithm is the best performing algorithm on the flat graph colouring problem and *NK*-landscape. The results in Table III show a better prediction accuracy for the LSMs based on the simulated annealing algorithm compared with those based on the original sampling algorithm. The performance of the algorithms run on the LSMs derived from the *NK*-landscape, the Flat

Graph Coloring and the Blocks World problem correlate very well with the performance of the algorithms on the real landscape, producing rank order distances of just 13, 12.5 and 24, respectively.

However, it is clear that there is commonality between the performances of the twenty one algorithms on the three different problems and so the above raw results do not, on their own, rule out the possibility that the LSMs merely mirror general landscape features of the three real problems, but not their specific features. To test this, we compared the rank order distances achieved by the LSM based on SA to rank order distances observed between each *pair of real problems*. The rank order differences for the pairs of real problems are given in Table IV: the best value is 32. For the LSM based on SA, the best value is 12.5 and the worst is 24. Fig. 2 gives the null distribution, i.e. the frequency that a particular rank order difference is achieved for random rankings. These frequencies were computed using a Metropolis-Hastings algorithm [8] run at a series of five temperatures³. From the plot, it follows that the ranking produced by the LSMs are between 100 and 1 million times less frequent than the best ranking differences between pairs of problems, supporting the hypothesis that the LSMs *are* modelling fine features of the problems.

We have also verified that the LSMs for these three different problems are structurally different from each other by plotting the LSM transition matrix using a colour map. This is not shown here, as it does not transfer well to grayscale, but there are clear differences in the LSMs constructed for the three different problems, again suggesting that fine-grained structure is being captured.

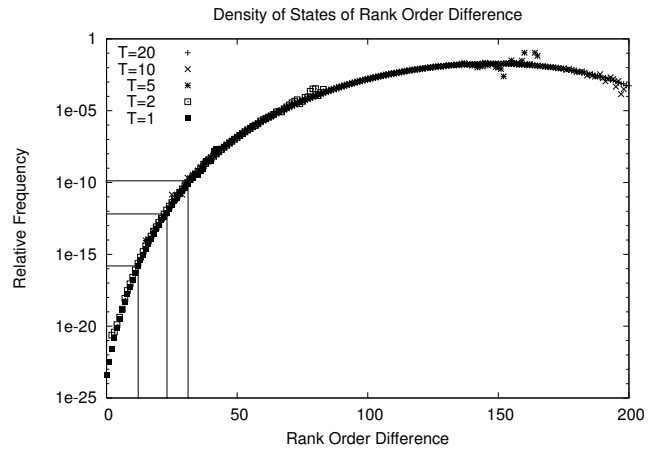


Fig. 2. Relative frequencies of rank order distances for random rankings. A ranking of 12.5 is approximately 10^6 times less frequent than a ranking of 32. A ranking of 24 is also indicated.

³We generated the random rankings by drawing a score uniformly at random in $\{1..21\}$ for each algorithm and then ranking based on these, in order to simulate the possibility of there being ties; however, this has little effect on the null distribution compared to using a randomly drawn permutation of the ranks 1 to 21, i.e. allowing no ties.

TABLE II

RANK ORDER OF PERFORMANCE OF ALGORITHMS ON LSM REPRESENTING NK -LANDSCAPE, FLAT GRAPH COLOURING AND BLOCKS WORLD OPTIMIZATION PROBLEMS, CONSTRUCTED BY THE EA DESCRIBED BY CORNE ET AL. RESULTS ARE BASED ON STATISTICAL DIFFERENCES IN BEST-FITNESS ACHIEVED OVER THIRTY INDEPENDENT TRIALS; BEST PERFORMING ALGORITHM IS LISTED FIRST. THE BOTTOM TWO ROWS SHOW THE COLUMN SUM \bar{R} OF RANK DIFFERENCES, COMPARED WITH THE RANKING ON THE REAL PROBLEM (SEE TABLE 1), AND THE ESTIMATED SIGNIFICANCE COMPARED WITH A RANDOM RANKING (SEE FIG. 2).

NK	Distance	FGC	Distance	BW	Distance
1	niche F	1.5	1.5	niche J	0.5
2	niche J	0.5	1.5	niche F	0.5
3.5	niche I	1	3.5	niche I	0.5
3.5	F	1	3.5	F	2.5
5	SA	4	5.5	niche E	0.5
6	E	2	5.5	E	4
8	niche H	0	7.5	niche H	2
8	niche E	2	7.5	SA	5.5
8	J	0	9.5	J	3.5
10	I	1.5	9.5	B	1.5
11	B	1	11	I	4
12.5	niche B	1	13	D	0
12.5	H	0.5	13	H	3
14	D	0.5	13	niche B	7
15.5	niche D	1	15	niche D	4
15.5	niche G	0.5	16	niche G	3
17	G	0	17	G	0
18.5	niche A	0.5	18.5	niche A	1
18.5	niche C	0.5	18.5	niche C	1
20.5	A	1.5	20	A	0.5
20.5	C	0.5	21	C	1.5
		21			46
		$\sim 10^{-12}$			$\sim 10^{-8}$
					28
					$\sim 10^{-11}$

B. Simulated annealing-based LSM convergence/accuracy

To investigate the convergence of the LSM transition matrix as more samples are taken, we constructed the LSM incrementally using short runs of SA. Specifically, we ran the SA twenty times for 5000 evaluations each, and after each run, the the LSM was updated using the new samples, and then used as usual to rank the performance of the 21 search algorithms. Two methods were then used to assess the LSM's convergence: (i) the rank order distance compared to the real ranking of the 21 algorithms; (ii) the change in the rank order, comparing the updated LSM with the LSM before the last update. Obviously, (i) uses 'external knowledge' of the true ranking of the algorithms, which would not be available in practical applications, whereas (ii) does not need the true rankings. We were interested to see how well (ii) correlated with (i), to see if it is possible to estimate when to stop 'training' the LSM. We repeated the experiments also using SA runs of 10000 evaluations (with the cooling schedule changed appropriately).

The rank order distance for each of the landscapes, together with the change in rank order distance, are plotted on the same axes in Figures 2 and 3. It can be seen that the LSM converges to a reasonable prediction of the performance of the algorithms within 20,000 evaluations on most of the

TABLE III

RANK ORDER OF PERFORMANCE OF ALGORITHMS ON LSM REPRESENTING NK -LANDSCAPE, FLAT GRAPH COLOURING AND BLOCKS WORLD OPTIMIZATION PROBLEMS, CONSTRUCTED BY THE SIMULATED ANNEALING ALGORITHM. RESULTS ARE BASED ON STATISTICAL DIFFERENCES IN BEST-FITNESS ACHIEVED OVER THIRTY INDEPENDENT TRIALS; BEST PERFORMING ALGORITHM IS LISTED FIRST. THE BOTTOM TWO ROWS SHOW THE COLUMN SUM \bar{R} OF RANK DIFFERENCES, COMPARED WITH THE RANKING ON THE REAL PROBLEM (SEE TABLE 1), AND THE ESTIMATED SIGNIFICANCE COMPARED WITH A RANDOM RANKING (SEE FIG. 2).

NK	Distance	FGC	Distance	BW	Distance
1.5	niche F	1	1.5	niche F	2
1.5	niche J	1	1.5	F	2
3	SA	2	3	niche J	1.5
4.5	niche I	0	4	niche I	1
4.5	F	0	5.5	SA	4
6.5	E	1.5	5.5	J	2
6.5	niche E	0.5	8	niche H	0.5
8	niche H	0	8	E	0.5
9	J	1	8	I	0.5
10.5	I	1	10	niche E	0
10.5	B	0.5	11	H	0.5
12	niche B	0.5	13	D	1
13.5	D	1	13	niche B	1
13.5	H	0.5	13	B	1.5
15	niche D	0.5	15.5	niche G	1.5
16	niche G	0	15.5	G	1
17	G	0	17	niche D	0.5
18.5	niche A	0.5	19	niche A	0.5
18.5	niche C	0.5	19	A	0.5
20	A	1	19	niche C	0.5
21	C	0	21	C	1.5
		13			12.5
		$\sim 10^{-16}$			$\sim 10^{-16}$
					24
					$\sim 10^{-12}$

TABLE IV

RANK ORDER DISTANCES BETWEEN PAIRS OF REAL PROBLEMS AND THE FISHER PERMUTATION TEST SIGNIFICANCE (SEE FIG. 2.)

Problems or problem and LSM compared	\bar{R}	Fisher
NK real landscape and FGC real landscape	32	$\sim 10^{-10}$
NK real landscape and BW real landscape	32	$\sim 10^{-10}$
FGC real landscape and BW real landscape	56	$\sim 10^{-7}$

optimization problems. This convergence is reflected by the change in the rank order of the performance of the 21 algorithms on the addition of the new evaluations in each optimization problem. This can be more clearly seen in the plots representing the performance of the LSMs constructed in 5000 evaluation steps.

The NKp -landscape where $K=2$ appears to be the most challenging for the LSM to model. After 100,000 evaluations the two forms of LSM constructed in 5000 and 10000 intervals produced rank order differences of 44 and 64.

There appears to be little difference in the performance of the LSM generated with the two sampling algorithms, although the performance of the LSM generated with the sampling algorithm constructed in 10,000 evaluation intervals on the Flat Graph Coloring problem appears to be slightly more erratic.

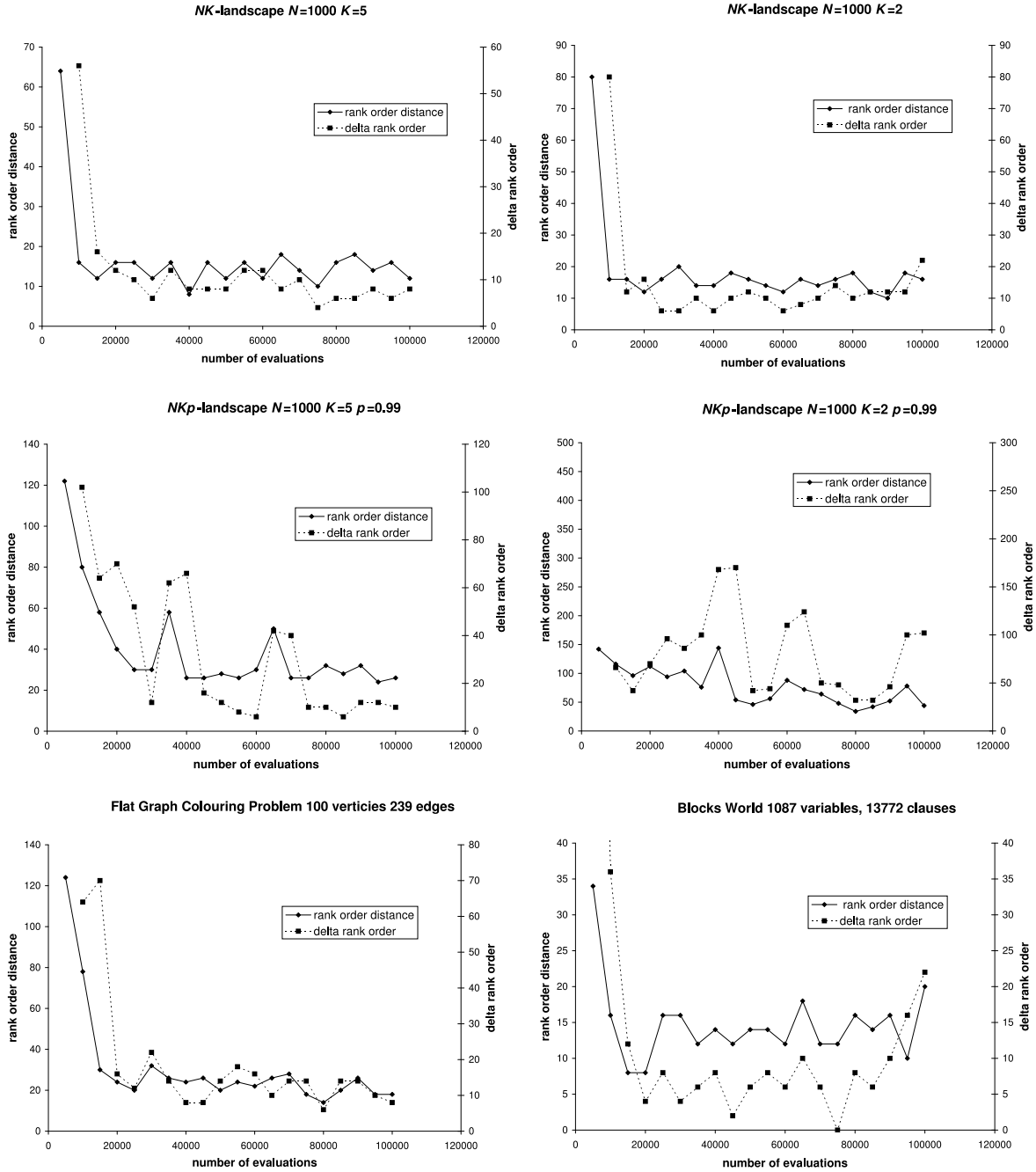


Fig. 3. Plots displaying the relation between; rank order distances between algorithms run on real landscape and LSMs constructed from these landscapes, the number of evaluations used in constructing the LSM and the total change in the transition matrix each time a new set of evaluations is added. LSMs constructed progressively using a sampling algorithm with 5000 evaluations.

C. Mutation rate tuning

The four best performing algorithms on the NK -landscape where $N=1000$ and $K=5$ described earlier (see Table I) were assessed using four separate genewise mutation rates of $1/L$, $2/L$, $4/L$ and $8/L$. An LSM based on a simulated annealing sampling algorithm run on this landscape was constructed using a mutation rate of $1/L$ and the performance of algorithms on this LSM was ranked. The procedure for running the algorithms on the LSM was then modified such

that multiple transitions were applied to solutions between fitness evaluations (as described in the methods section). The performance of the algorithms at these modified mutation rates was then ranked.

The overall rank order distance of the algorithms run on the real landscape at the different mutation rates and those run on the modified LSM is 40.0. This is despite the performance of many of the algorithms being similar. Overall, the mutation rate of $1/L$ works best on this problem,

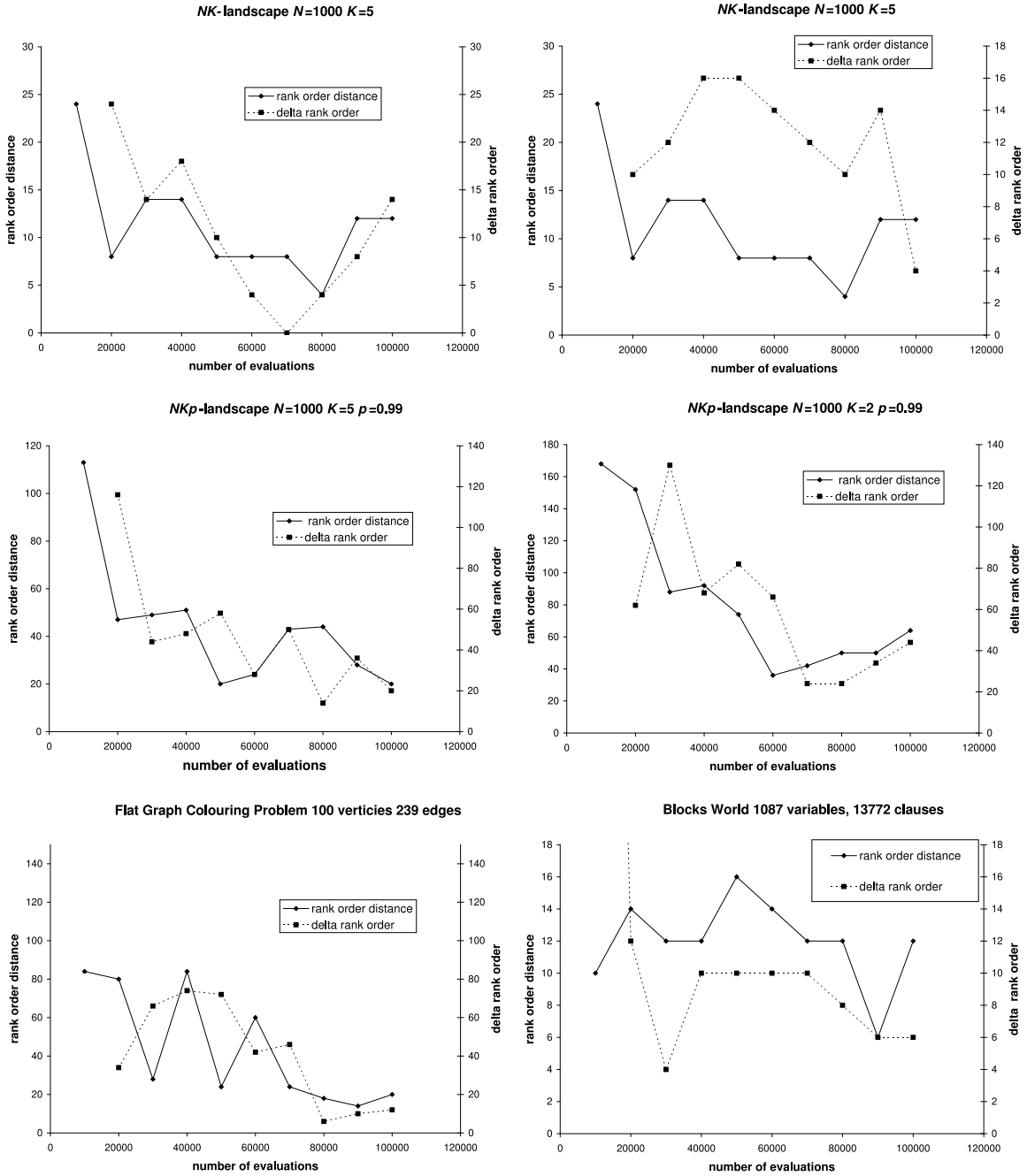


Fig. 4. Plots displaying the relation between; rank order distances between algorithms run on real landscape and LSMs constructed from these landscapes, the number of evaluations used in constructing the LSM and the total change in the transition matrix each time a new set of evaluations is added. LSMs constructed progressively using a sampling algorithm with 10000 evaluations.

with a reduction in performance of the algorithms with increasing mutation rate. This is reflected very well in the performance of the algorithms on the LSM.

IV. DISCUSSION

The results given above suggest that Landscape State Machines may prove useful as a tool for the prior tuning of stochastic algorithms for optimization problems where evaluation of solutions is (even moderately) expensive. It

has been demonstrated that, with an appropriate sampling algorithm, the rank order of performance of a range of algorithms, including simulated annealing and niching genetic algorithms can be predicted with a high level of accuracy. Initial studies suggest that mutation rates may also be tuned using the LSM methodology, allowing the optimum mutation rate to be determined for each algorithm. It should also be possible to assess algorithms with dynamic mutation rates.

Although the LSMs in this study, constructed using simu-

TABLE V

RANK ORDER OF PERFORMANCE OF ALGORITHMS ON LSM REPRESENTING NKp -LANDSCAPE, WHERE $N = 1000$, $K = 5$, $p = 0.75$, MODELING ORIGINAL MUTATION RATE OF $1/L$ AND MODIFIED MUTATION RATE OF UP TO $10/L$, CONSTRUCTED USING A SIMULATED ANNEALING ALGORITHM. RESULTS ARE BASED ON STATISTICAL DIFFERENCES IN BEST-FITNESS ACHIEVED OVER THIRTY INDEPENDENT TRIALS; BEST PERFORMING ALGORITHM IS LISTED FIRST. THE BOTTOM TWO ROWS SHOW THE COLUMN SUM OF RANK DIFFERENCES, \bar{R} AND THE P-VALUE FOR ACCEPTING THE NULL HYPOTHESIS THAT THE RANK ORDER DIFFERENCE \bar{R} IS NOT SIGNIFICANTLY DIFFERENT FROM THAT OBTAINED FOR A RANDOM RANKING (FISHER TEST).

NK -landscape	Algorithm run on LSM		Distance
	based on NK -landscape		
1.5	SA (1/L)	3 SA (2/L)	1.5
1.5	SA (2/L)	3 SA (1/L)	1.5
4	niche F (1/L)	3 SA (6/L)	4.5
4	SA (4/L)	3 SA (4/L)	1
4	niche J (1/L)	3 SA (8/L)	11
6	niche J (2/L)	6.5 niche F (1/L)	2.5
7.5	niche F (2/L)	6.5 niche J (1/L)	0.5
7.5	SA (6/L)	8 niche J (2/L)	2
10	F (2/L)	9 F (1/L)	1
10	F (1/L)	10 niche I (1/L)	0
10	niche I (1/L)	12 niche I (2/L)	0
12	niche I (2/L)	12 niche F (2/L)	4.5
13	niche J (4/L)	12 F (2/L)	2
14	SA (8/L)	14 niche J (4/L)	1
16	niche I (4/L)	15 niche I (4/L)	1
16	niche J (6/L)	17 niche F (4/L)	1
16	niche F (4/L)	17 niche J (6/L)	1
18	F (4/L)	17 F (4/L)	1
19	niche J (8/L)	19 niche I (6/L)	1
20	niche I (6/L)	20 niche J (8/L)	1
22	niche F (6/L)	22 F (6/L)	0
22	niche I (8/L)	22 niche F (6/L)	0
22	F (6/L)	22 niche I (8/L)	0
24	niche F (8/L)	24.5 niche F (8/L)	0.5
25	F (8/L)	24.5 F (8/L)	0.5
			40.0 ($\sim 10^{-9}$)

lated annealing, showed good behaviour at predicting search algorithm performance, it is unclear whether this is because the SA was itself one of the best search methods. In general, it may be safer to construct the LSM from an aggregate of different sampling algorithms; then if one of the sampling schemes is deficient for a particular problem, it may be compensated by other schemes that perform better. Racing algorithms [3] could be used to construct the LSM and do the statistical testing incrementally. In this approach, the algorithm used to update the LSM could be the one which is doing best so far. Algorithms doing poorly could be dropped as soon as there is sufficient statistical evidence against them.

A. Application to directed evolution experiments

The LSM approach may provide a very interesting means of selecting algorithms for *directed evolution* experiments. In directed evolution, novel biomolecules (for a range of applications) are evolved *in vitro* by repeated rounds of mutation and selection. This is an optimisation problem which is very ‘black-box’; indeed, it is not economically viable to even know what the protein sequences made along

the way actually are, since this would involve an expensive sequencing process. This makes LSMs a particularly suitable choice, since they do not use decision space information. Moreover, the tuning of potential algorithms is prohibited by the time and expense of generating and evaluating new variants *in vivo*, in this application, so great gains can be potentially made by tuning algorithms off-line.

V. CONCLUSION

We have furthered our knowledge regarding the efficacy of using LSMs constructed by sampling to predict search algorithm performance. In particular, we have increased our confidence that LSMs can predict the relative performance of heterogeneous algorithms and further shown the potential for mutation rate tuning using LSMs. There is much scope for future work but we are now focusing on the application of LSMs to directed evolution.

Acknowledgments

Joshua Knowles is supported by a David Phillips Research Fellowship from the Biotechnology and Biological Sciences Research Council (BB-SRC), UK.

REFERENCES

- [1] L. Altenberg. The schema theorem and Price’s theorem. In *Foundations of Genetic Algorithms 3*, pages 23–49. Morgan Kaufmann, 1995.
- [2] L. Barnett. Ruggedness and neutrality—the NKp family of fitness landscapes. In *Proceedings of the Sixth International Conference on Artificial life*, pages 18–27. MIT Press, 1998.
- [3] M. Birattari, T. Stützle, L. Paquete, and K. Varrentrapp. A racing algorithm for configuring metaheuristics. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2002)*, pages 11–18. Morgan Kaufmann, 2002.
- [4] R. Collins and D. Jefferson. Selection in massively parallel genetic algorithms. In *Proceedings of the 4th International Conference on Genetic Algorithms*, pages 249–256. Morgan Kaufmann, 1991.
- [5] D. Corne, M. Oates, and D. Kell. Landscape state machines: tools for evolutionary algorithm performance analyses and landscape/algorithm mapping. In *Applications of Evolutionary Computing*, volume 2611 of *LNC3*, pages 187–198. Springer, 2003.
- [6] Á. E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3:124–141, 1999.
- [7] J. Garnier, L. Kallel, and M. Schoenauer. Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7:173–203, 1999.
- [8] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57:97–109, 1970.
- [9] T. Hogg. Refining the phase transition in combinatorial search. *Artificial Intelligence*, 81:127–154, 1996.
- [10] S. Kauffman and S. Levin. Towards a general theory of adaptive walks on rugged landscapes. *Journal of Theoretical Biology*, 128:11–45, 1987.
- [11] H. Kautz and B. Selman. Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the Thirteenth National Conference on AI (AAAI-96)*, pages 1194–1201. AAAI, 1996.
- [12] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.
- [13] B. Naudts and L. Kallel. A comparison of predictive measures of problem difficulty in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 4(1):1–15, April 2000.
- [14] A. Prügel-Bennett and J. L. Shapiro. The dynamics of a genetic algorithm for simple Ising systems. *Physica D*, 104:75–114, 1997.
- [15] G. R. Raidl and B. A. Julstrom. Edge sets: An effective evolutionary coding of spanning trees. *IEEE Transactions on Evolutionary Computation*, 7(3):225–239, 2003.
- [16] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1:67–82, 1997.