Landscape State Machines: Tools for Evolutionary Algorithm Performance Analyses and Landscape/Algorithm Mapping

David Corne[1], Martin Oates[1,2], and Douglas Kell[3]

[1] School of Systems Engineering, University of Reading, Whiteknights, Reading RG6 6AY, United Kingdom
d.w.corne@reading.ac.uk
[2] Evosolve Ltd, Stowmarket, Suffolk, United Kingdom,
moates@btinternet.com
[3] Department of Chemistry, UMIST, Manchester, United Kingdom
dbk@umist.ac.uk

Abstract. Many evolutionary algorithm applications involve either fitness functions with high time complexity or large dimensionality (hence very many fitness evaluations will typically be needed) or both. In such circumstances, there is a dire need to tune various features of the algorithm well so that performance and time savings are optimized. However, these are precisely the circumstances in which prior tuning is very costly in time and resources. There is hence a need for methods which enable fast prior tuning in such cases. We describe a candidate technique for this purpose, in which we model a landscape as a finite state machine, inferred from preliminary sampling runs. In prior algorithm-tuning trials, we can replace the 'real' landscape with the model, enabling extremely fast tuning, saving far more time than was required to infer the model. Preliminary results indicate much promise, though much work needs to be done to establish various aspects of the conditions under which it can be most beneficially used. A main limitation of the method as described here is a restriction to mutation-only algorithms, but there are various ways to address this and other limitations.

1 Introduction

The study of fitness landscapes [14] in the context of evolutionary search strives to understand what properties of fitness landscapes seem correlated with the success of specific evolutionary algorithms (EAs). Much progress has been made, with a number of landscape metrics under investigation as well as sophisticated statistical techniques to estimate landscape properties [1,5,7,9,11,13]. Meanwhile, much effort has also gone into constructing landscapes with well understood properties in attempt to yield hypotheses and guidelines which may apply to 'real' landscapes [4,6,8,11,12]. For the most part, a striking aspect of such investigations has been the ability of EAs consistently to undermine predictions [1,6,10]. Correlation between proposed metrics or landscape features and evolutionary algorithm difficulty tends to be weak, or bothered by the presence of convincing counterexamples. Here we present an alternative approach to exploring landscapes and algorithm performance on them. We model a landscape as a finite state machine, whose states represent fitness levels and state transition probabilities characterize mutation. Such a model can be approximately inferred from sampling during an EA (or other algorithm) run, and then used for test-driving a range of algorithms under consideration to apply to the 'real' landscape.

This promises to be valuable in cases where good results are at a premium, but fitness evaluations on the 'real' landscape are prohibitively expensive, precluding intensive *a priori* algorithm comparison and/or parameter tuning; in contrast, a fitness evaluation on a landscape state machine (LSM) is computationally trivial. The success of this technique for algorithm comparison depends on the degree to which the LSM approximation captures those features of the real landscape which are salient in terms of algorithm comparison. Viability also depends on whether sufficiently useful LSMs can be inferred without incurring undue cost in prior search of the real landscape. We start to investigate these questions, and conclude that the technique is promising.

2 Landscape State Machines

What we call a landscape state machine (LSM) is simply a finite state machine (FSM) which models certain aspects of a search landscape. To be specific, given a search space *E*, an operator *M* (which takes a point $s \in E$, and returns another point from *E*) and an associated transition matrix *T* (such that t_{ij} gives the probability of yielding point $j \in E$ after applying *M* to $i \in E$). An LSM model of this landscape is a set of states and arcs (*S*, *A*), such that *S* corresponds to a partition of the set *E*, and *A* corresponds to an abstraction of *T*. In the extreme, the LSM can model a landscape precisely, with precisely one state for each point in *E*, and *A* corresponding precisely to *T*.

In the general case, one state in the LSM will map onto many points in *E*. We will normally expect the mapping between *S* and *E* to be such that each state *s* corresponds to a set of points in *E* with equivalent fitness. More generally, we may define an equivalence relation *R*, which partitions *E* into *c* of equivalence classes $E_1, ..., E_c$. The states in the LSM can then correspond precisely to the equivalence classes. In the case that the equivalence relation *R* forces equality in both fitness and genotype, the LSM model becomes the exact model described above. More generally, and as we later do in section 4, we might define *R* in such a way that states may be associated with a partition of the fitnesses into bands (e.g. all points with fitness between 0.2 and 0.3 might define an equivalence class).

2.1 Examples and Motivation

Before we discuss the uses of this simple LSM concept, an example will serve to clarify and motivate the issues involved. Consider the simple MAX-ONES problem, in which a candidate solution is a binary string of length *L*, and the fitness of a candidate (which is to be maximised) is the number of 1s it contains. Further, imagine we are interested in addressing this problem with an EA, and will employ the single-gene bit-flip mutation operator (but no other operator). That is, when we mutate a candidate solution, a single bit is chosen at random and its value is flipped.

The LSM in Figure 1 is model for this landscape when L = 5. There are six distinct fitnesses in this case, and we can identify each state s_i with the entire set of points whose fitness is i. It should be clear that the arc probabilities are simply derived in this case, and indeed we could easily calculate the corresponding LSMs for this landscape whatever the value of L. For example, if L was 1000 then the arc leading from s_{217} to s_{218} would have probability 1/783, which is the chance of the mutation operator flipping one of the '0' bits of a candidate represented by s_{217}.

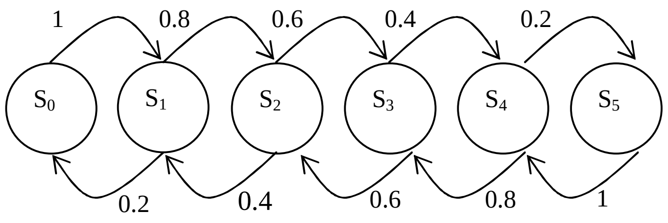

Fig. 1. A landscape state machine for the MAX-ONES function with L = 5, assuming single-gene flip mutation

Such a landscape model becomes practically useful when we consider the following scenario. Suppose that one wishes to compare the performance of each of m different EAs on MAX-ONES, using only single-gene bit-flip mutation. Also suppose each algorithm eschews any elements or operations (other than the fitness function) which require genotypic knowledge. E.g. phenotypic crowding may be employed, but not genotypic crowding. The space of algorithms which remain is certainly not overly restricted. E.g. we may wish to compare 1000 algorithms, each of which is a population-based evolution strategy, for all combinations of 10 different population sizes, 10 different selection schemes, and 10 different population-structure strategies (e.g. including islands-based schemes with a variety of migration strategies).

To properly compare these m EAs, we need t runs of each algorithm, and will typically run each trial of each algorithm for a baseline fixed number of fitness evaluations, e. The total number of fitness evaluations will be $m.t.e$, which can easily come to several billions or more on applications of interest. In the case of MAX-ONES the fitness calculation is inexpensive, but consider, for example, the requirement to measure these algorithms' performance when L=1,000,000. An intriguing fact is as follows. For algorithms of the type outlined, and landscapes which we can *precisely* model with an LSM, we can replace the landscape by the model, yet predict the resulting comparative algorithm performance with full statistical accuracy. As long as we use a suitable initial distribution of initial fitness labels for the initial population, we can expect the statistical properties of comparative algorithm performance on the described LSM model, of MAX-ONES to precisely (in statistical terms) match performance on the MAX-ONES landscape itself.

It is worth clarifying at this point what we mean by "replace the landscape with the model". To run an EA on an LSM, we simply replace the notion of a candidate solution with that of a state in the LSM. The initial population is simply a state number, and (assuming state numbering is mapped into a simple way onto fitnesses), and selection operates in the normal way. Evaluation is trivial, since the state number corresponds to fitness. Finally, when an individual (state) is mutated, the child state is chosen probabilistically according to the transitions emanating from the parent state.

By using LSM models for differential algorithm performance analysis, it could be possible to significantly reduce development times. The ability to replace a landscape with an LSM model allows, among other things, the possibility of the following method for use in developing EA-based solutions to an optimisation problem:

1. Derive an LSM model of the landscape in question;
2. Run many candidate algorithm designs on the LSM model;
3. Choose a suitable algorithm design given the results of step 2.
4. Use the chosen design for further development on the real landscape.

Steps 2–3 exploit the fact that an LSM model can be searched far more speedily than the real landscape. To the extent that the dynamics of an algorithm's search on the LSM accurately reflect its dynamics on the real landscape, the information gleaned in steps 2–3 will support an appropriate step 4 choice.

Where precise LSMs can be inferred, the problem is undoubtedly a toy one, but the LSM technique still may uses. E.g., with perfect statistical accuracy we can accurately investigate the relative performance of a wide range of EAs on MAX-ONES with L=1,000,000,000, without ever needing to store or evaluate a single 1Gb genome. The thought that this may be possible on more interesting and realistic landscapes is appealing, but the LSM model in such cases will invariably be an *approximate* model. It remains to be seen whether approximate LSMs of interesting landscapes retain the possibility to enable accelerated appropriate choice of algorithm design.

3 Approximate LSMs

Consider the LSM in Figure 2, which models the order 3 deceptive trap function with $L = 6$, again assuming single-gene bit-flip mutation. Note that the use of this particular mutation operator is not a restriction on the method, it just simplifies illustration. In contrast, the restriction to mutation only *is* a limitation which, though not insurmountable, is beyond the scope of a first description of the concept of LSMs and is the topic of ongoing work. Now, Figure 2 is an accurate LSM in the sense that any arc represents the correct proportion of mutations from the sending state which will arrive at the receiving state. State labels correspond precisely to fitnesses, e.g. state s_4 represents all genomes with fitness 4, and the arc to state s_5 indicates that a mutation of a point of fitness 4 is likely to yield a result of fitness 5 with probability 0.33. However, in an important way, it is less helpful than the earlier LSM in capturing the landscape features of importance in exploring search algorithm performance. This is simply because there is high ‘profile variance’ within some states, which we now explain.

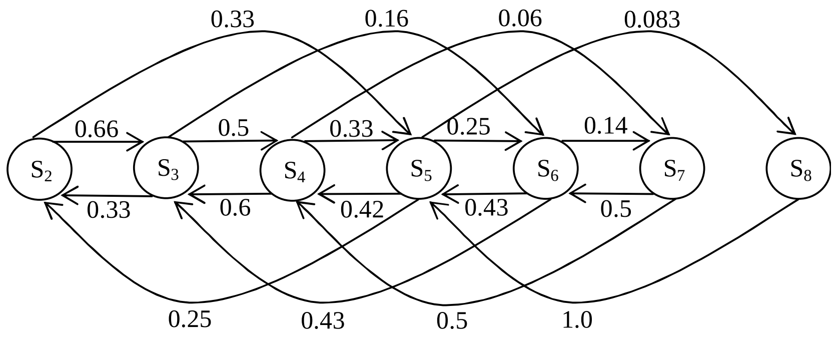

Fig. 2. An LSM for the order 3 deceptive problem with *L* = 6 and single-gene bit-flip mutation

The 'profile' of $s \in E$ is the vector of its transitions t_{sj} for all *j*. The profile elements correspond precisely to a_{sj} in the case when the LSM has one state for each point in *E*. In the previous MAX-ONES LSM, the vector of arcs from any state precisely matches the profile of every point ‘contained’ in that state. But in general the vector of arcs will be a weighted average of these profiles. Consider the state corresponding to a fitness of 6 in Figure 2. Two kinds of points in this landscape have fitness 6. One is exemplified by the point 000110, with one chunk of three genes fully unset, contributing 4, and a second chunk with 2 bits set, contributing 2. The six points of this type each have the profile (0.0, 0.5, 0.0, 0.33, 0.0, 0.16, 0.0) – that, is chance 0.0 of a mutant yielding a fitness of 2, chance 0.5 of a mutant yielding a fitness of 3, and so on. The other kind of point with fitness 6, of which there is just one, is 111111, whose profile is: (0.0, 0.0, 0.0, 1.0, 0.0, 0.0, 0.0). The profile of s_6 in Figure 2 represents these two profiles weighted and combined. A *precise* LSM for this problem would have two different states corresponding to a fitness of 6, one for each of the two corresponding genotypes. The LSM of figure 2, however, is an approximate LSM for this particular problem, since the dynamics of an algorithm running on it will not exactly reflect the dynamics of that algorithm on the real problem.

In general (and which we do in section 4), LSMs will be inferred from data obtained from preliminary sampling runs on the real landscape. Given the general case of no prior landscape knowledge, we will neither know the correct number of states for the LSM to use, nor the number of different profiles that may be shared by points with a given fitness. LSMs inferred by sampling will thus be highly approximated models of the precise LSM for that landscape. In the experiments which follow, we use the simplest kind of basic LSM framework in which (as in figures 1 and 2) each state represents a given fitness level, and there is only one state per fitness level.

4 Empirical Experience

Given that exact LSM models would clearly be very valuable, but approximations are inevitably all we are able to expect in reality, we need to investigate the degree to which an inferred LSM model for test-drive algorithm comparison purposes is at all useful, and get some insight into how this depends on the amount of prior sampling done, and on the underlying difficulty of the modeled landscape.

To underpin these analyses we use a simple metric for the utility of an LSM model. Suppose we wish to understand the relative performance of m different algorithms on a landscape P. We would naturally address this by performing several trials of each algorithm on P, and then obtain a rank-ordering of the algorithms. By performing algorithm comparisons trials on the LSM, with all other aspects of the experimental setup being the same, we yield another such rank ordering. We can then equate the quality of the LSM model with the distance of this rank ordering from that obtained on the real landscape. E.g. if algorithm A is in position 3 in one rank-order, and in position 5 in another, it contributes 2 to the total distance between the orderings. Clearly, if an LSM model scores 0, it precisely predicts the comparative performance of the tested algorithms on the real problem.

We now explain the design of our initial experiments in which we seek to explore LSM utility for approximate LSM models. We first set out, for easy reference, the algorithms and problems used, and later set out the experiments themselves.

4.1 Algorithms, Problems, and Preliminary Sampling

The algorithms tested are the following ten. In each case, the basic algorithm was a standard generational EA with elitism (a single best in the population is automatically entered into the next generation), using tournament selection (with replacement). The mutation operator in all experiments is gene-wise mutation with a probability of 0.001. That is, when a chromosome is mutated, each gene is mutated independently with that probability. This is unlike the single-gene bit-flip mutation operator used previously in illustrations, and leads to a more complicated LSM which is a greater challenge to approximate. The algorithms compared differ in tournament size and population size, and also number of generations (but the latter varied simply to ensure comparable fitness evaluations in each case).

- 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, 5 respectively.
- G–J. Population size 50. 91 generations, tournament sizes 3, 5, 7, 10 respectively.

This choice of algorithms is *ad hoc* and pragmatic, but reflects a suitable collection of potential parameterizations of an EA which might be tested, given the requirement for reasonably fast convergence, and given no prior empirical or theoretical knowledge of the problem in hand.

We use a different EA for preliminary sampling (in order to infer the LSMs used in the algorithm comparison tests). This is the same basic EA, but with a population size of 200 and random selection, run for either 50,000 or 100,000 evaluations. This is again an *ad hoc* choice; later we briefly discuss potentially better, or ways to choose, preliminary sampling techniques. Here we simply note that one salient aspect of this is the number of evaluations. If the 'real landscape' algorithm comparison would consume *e* evaluations, the number of evaluations consumed in preliminary sampling, *p*, should be suitably less than *e*. As we will see, *p* is 450,000 in the ensuing tests.

The optimization landscapes we explore are MAX-ONES problem with $L = 1000$, and *NK* landscapes with $N = 1000$, and with *K* set at 5 or 10. For each landscape the following is done: we run the preliminary sampling EA once on the real landscape, building two LSMs for each, one corresponding to the data gleaned up to 50,000 evaluations and another from the data gleaned up to 100,000 evaluations.

We infer the LSM as follows. Every (parent fitness, child fitness) pair corresponding to a mutation is recorded. With general real-world application in mind (in which there is no prior knowledge of the number of fitnesses), we chose instead to fix the number of states in the LSM at 200, and thus map the information gleaned, in every experiment, onto a 200-state LSM. The (parent) fitnesses sampled were simply scaled uniformly between 0 and 200; then, a state in the LSM was created to correspond to each interval $[x, x+1]$ for *x* from 0 to 199. Each state therefore corresponded to a number (in some cases 0) of fitnesses, and arc labels were calculated accordingly. A final point worth noting is that we also remembered the fitnesses of the initial population of the sampling run, and scaled them to yield a list of the corresponding states. The LSM trials then used this list of states to build its initial populations. E.g. for a trial run with a population of size 10, the initial population of the algorithm trial on the LSM would be generated by sampling uniformly 10 times from this list.

4.2 Experiments

The aim of these experiments was to gain some preliminary insight into how useful an inferred LSM might be in discriminating the performance of several algorithms. The experimental protocol for a given problem *P* was as follows:

1. Run the preliminary sampling EA on *P* in order to harvest data for the LSM.
2. Infer an approximate LSM for *P* from the data gathered in step 1.
3. Run an algorithm comparison study on *P*, with the aim of discriminating the relative performances of several chosen EAs.
4. Run the same algorithm comparison study on the LSM inferred for *P*, establishing the algorithms' relative performance on the LSM.
5. Compare the ordering of algorithms given by step 4 with that given by step 5.

In step 3, mirroring the typical shape of a real-world study which desires enough trial runs for a chance of statistical confidence, yet still needs to be parsimonious in time and resources, we run ten trials of each of the algorithms A–J described in section 4.1. We record the result (best fitness found) in each trial, and a rank ordering is then determined by running *t*-tests. The same is done in step 4, however this time relative performance is determined by speed of convergence. As we will further discuss later, for various understandable reasons the algorithms would typically converge to the optimum of the LSM (i.e. state 200) before their evaluation limits were reached, and so we could not discriminate in step 4 solely in terms of highest state attained.

A reminder of the object of all this is as follows: given conditions under which such experiments might be successful (i.e. the rank ordering found in step 4 is significantly close to that found in step 3), and supposing we have the task of choosing an algorithm to use in anger on a real problem, step 3 would not be necessary. We could skip to step 4, discover the algorithm which performs best on the LSM, and take this to be a suitably accurate prediction of which algorithm would perform best on the real landscape. Alternatively, we might use the results of step 4 to home in on a smaller selection of choices for tests on the real landscape. The conditions under which this scheme may be successful are easy to state: having an LSM which perfectly captures the landscape. What is to be determined is whether an approximate LSM can capture enough of the salient details of the landscape to produce accurate predictions in step 4.

4.3 Results

Table 1 shows the rank orderings identified in step 3 by running algorithms A–J on each of the three problems addressed here. Table 2 gives the rank ordering of algorithms identified by experiments on the appropriate inferred 200-state LSMs following 50,000 evaluations of the preliminary sampling EA. This table also provides measures of the distance between the LSM-inferred ordering and the real (Table 1) ordering. Associated with each algorithm is its distance *d* from its position in the corresponding Table 1 ordering, and the sum of these distances is also given. Table 3 provides the same information for the 100,000 evaluation LSMs. A brief summary of the main findings of these preliminary experiments is as follows.

Accuracy of the 50,000-evals LSM for MAX-ONES. The rank order of algorithms obtained by experiments on the 50,000-evals approximate LSM for MAX-ONES is nearly identical to that obtained from experiments on the real MAX-ONES landscape. This augurs very well for the method, since it seems that with 50,000 evaluations' worth of effort, we have predicted with near-perfect accuracy, and in a fraction of the time (since running the LSM experiments is extremely fast) a rank ordering which needed approximately 10 times that amount of effort on the real landscape. Even though MAX-ONES is a simple landscape, this initial result is promising since it shows that an approximate LSM can be used for this purpose, and predicting comparative algorithm performance remains very difficult, even on MAX-ONES, owing the complexity of EA dynamics.

Accuracy of the Approximate LSMs on *NK* with *K* = 5. The total distance between the 50,000-evals LSM-obtained ordering and the Table 1 ordering is statistically significant, indicating that we can say with confidence that the predictive ability of the 50,000-evals LSM experiments was far better than random. In the 100,000-evals case, the ordering is even better, and the LSM seems particularly good at distinguishing which algorithms are particularly unsuited to this landscape, and well able to identify a collection which should perform well. The improvement in distance as we go from 50,000 to 100,000 evaluations is expected, since more real data has been used in inferring the LSM.

Table 1. Rank-orderings of algorithms *A–J* on each of the three problems studied, identified via *t*-tests on ten trial runs of each algorithm. '1' is best and '10' is worst.

Problem	1	2	3	4	5	6	7	8	9	10
MAX-ONES	B	F	E	D	A	J	I	H	G	C
NK, *K* = 5	F	B	E	D	J	A	I	H	G	C
NK, *K* = 10	F	B	E	J	D	I	H	A	G	C

Table 2. Rank-orderings of algorithms *A–J* on each of the three LSMs inferred from 50,000 evaluation preliminary sampling runs. Rankings identified and annotated as in Table 1, but in this case, algorithms are compared in terms of speed of attaining the final state (state 200), using highest state attained where necessary. The column below each rank position gives the distance of the corresponding algorithm from is rank position on that problem in Table 1. The final column provides the total distance from the Table 1 ordering

Problem	1	2	3	4	5	6	7	8	9	10	Distance
MAX-ONES	B	F	E	D	J	A	I	H	G	C	
	0	0	0	0	1	1	0	0	0	0	2
NK. *K* = 5	B	E	D	F	A	J	H	I	G	C	
	1	1	1	3	1	1	1	1	0	0	10
NK. *K* = 10	F	J	I	H	G	D	C	B	A	E	
	0	2	3	3	4	1	3	6	1	7	30

Table 3. Rank-orderings of algorithms *A–J* on each of the three LSMs inferred from 100,000 evaluation preliminary sampling runs. Rankings identified and annotated as in Table 2

Problem	1	2	3	4	5	6	7	8	9	10	Distance
MAX-ONES	B	F	E	D	A	J	I	H	G	C	
	0	0	0	0	0	0	0	0	0	0	0
NK. *K* = 5	B	E	F	D	A	J	I	H	G	C	
	1	1	2	0	1	1	0	0	0	0	6
NK. *K* = 10	B	E	I	D	F	J	H	A	G	C	
	1	1	3	1	4	2	0	0	0	0	12

Accuracy of the Approximate LSMs on *NK* with *K* = 10. The total distance between the 50,000-evals LSM-obtained ordering and the Table 1 ordering is better than the mean of a random distribution of orderings (33), but is not statistically significant, indicating that we cannot say that the LSM in this case has been able to distinguish between the algorithms better than a random ordering would have done. However, the more accurate 100,000-evals LSM is able to obtain an ordering significantly close to the Table 1 ordering, and certainly provides usefully accurate predictions of the relative performance of the ten test algorithms on this highly epistatic problem.

Trends against Landscape Ruggedness and Sample Size. In partial summary of the above observations, it is worth noting that the accuracy of predicted relative performance decreases with ruggedness of the landscape and increases with the amount of data obtained from (i.e. the length of) the preliminary sampling run. Both of these trends were of course very much expected, however what we did not know *a priori* is whether significant accuracy could be obtained at all without an unreasonably large sample size, and on problems with nontrivial ruggedness. These preliminary experiments have shown that significant accuracy *can* be obtained with a reasonable sample size and on at least one highly rugged problem.

5 Discussion: Prospects and Research Issues

LSMs have a variety of potentially fruitful uses. For example, in the application of Directed Evolution [2,3], in which evolutionary algorithms are applied directly to the generation of novel proteins, one generation can take about 24 hours, and with considerable cost in molecular biology consumables. Similarly, structural design, especially regarding novel structures, often requires highly costly evaluation functions which may include fine-grained airflow or stress simulation. In such cases the need for fast and accurate methods to help design the algorithm is crystal clear.

We have started to explore the viability of inferring them from preliminary landscape data, with a view to obviating the need for algorithm choice and/or tuning runs. The results so far are encouraging, particularly given the following points. First, an arbitrary and simple LSM structure was used in each case, with no preliminary attempt to, for example, find the most suitable number of states for each problem. Second, the preliminary sampling EA was designed only as one of many possible ways to obtain parent/child fitness pairs for a good spread of fitnesses. Further investigation may reveal that alternative sampling approaches may be more effective, such as a Markov Chain Monte Carlo search, or an adaptive EA which explicitly attempts to sample evenly (or in some other suitably defined way) throughout the landscape. As it turned out, the prior sampling EA in each case performed rather less well than the better few of the test EAs in the real experiments; this means that the LSM contained no information about a considerable portion of the landscape traversed by these algorithms; it is therefore encouraging that the LSM was able to do a fair job in each case despite this fact. Third, there is no reason to expect that some useful accuracy of the LSM-based predictions would not be maintained if further (e.g. 20, or 50) different suitable algorithms were tested, and hence the ‘true’ potential savings in evaluations are potentially vary large. Fourth, LSMs can always be augmented by data from further runs on the real problem, If it turns out that a series of instances in the same problem class can be suitably modeled by the same approximate LSM (which seems intuitively reasonable), then each real algorithm run on the problem yields data which can augment the LSM and hence increase its accuracy for further algorithm choice/tuning experiments. Finally, an exciting prospect is the augmentation of inferred LSMs with prior knowledge or reasonable assumptions about the landscape.

There are many research issues here, ranging through the theory and practice of choosing a sampling method, how best to construct an LSM from given data, and establishing what confidence to bestow on the predictions of an applied LSM model.

A quite separate use of LSMs arisies from the fact that they can be artificially contrived (i.e. we can predefine a transition matrix), yielding a class of 'toy' landscapes which can be tuned in ways interestingly (and in some ways potentially usefully) different from other tunable models such as the NK family. We can construct landscape state machines, and hence models of real landscapes (although see below), at will. In so doing, it is not necessarily easy to directly and precisely control factors such as the degree of epistasis or the sizes of basins of attraction, however we *can* precisely control the number of fitness levels, the density of local optima, and how this density varies as we move around the landscape. The most attractive feature of LSMs in this sense is that we can describe an extraordinarily wide range of landscapes with them. A potential application of this is to search through spaces of landscape state machines to find, for example, landscapes which are particularly suited to one EA rather than another, or to a standard EA rather than hillclimbing, or to simulated annealing rather than hillclimbing, and so forth. This is an area we plan to explore in future work, however it is worth pointing out that one important research issue here is whether arbitrary LSMs correspond to any real landscapes. As it turns out, certain constraints are needed on the profiles of states in the LSM, but we have not yet fully resolved how to ensure an arbitrary LSM which meets these constraints is valid (i.e. corresponds to a realizable landscape over *k*-ary strings using a standard mutation operator).

7 Conclusions

In this preliminary article we have described a straightforward way to model a search landscape as a finite state machine, which we call a Landscape State Machine. An attractive aspect of such a model is that, for certain landscapes at least, the features relevant to the search dynamics of certain EAs can be fully captured in a very compact model, and this model can be 'run' in place of the 'real' landscape (perhaps saving immense time in fitness computations). Although this is no substitute for locating optima on the real landscape, it does substitute, with full confidence in the case of certain pairings of landscapes and algorithms, in the business of determining the relative performance of algorithms.

The real advantages of LSMs come, however, if the approximations necessary to model more interesting landscapes (in particular, modeling real landscapes via sampled data) are such that the salient aspects of algorithm performance remain captured in the models. In some cases LSMs may be accurate enough to enable correct (and very fast) differentiation between algorithm performances, thus correctly informing how resources should be applied in the case of the real landscape. Preliminary experiments on MAX-ONES and *NK* landscapes have demonstrated that the idea is promising. Further, by searching through spaces of LSMs, in future work we may be able to take a new look at the question of what kind of problem is easy/hard for an EA.

Acknowledgements

We are pleased to acknowledge the support of the BBSRC (UK Research Council for Biological Sciences), and Evosolve (UK registered charity no. 1086384). Thanks to its handy and efficient implementation of *NK* problems, Peter Ross' `pga` code was used for preliminary sampling and 'real landscape' EAs.

References

1. Altenberg, L. *Fitness distance correlation analysis: an instructive counterexample*. In Th. Bäck, editor, Proceedings of the 7th International Conference on Genetic Algorithms, pages 57--64. Morgan Kaufmann Publishers, 1997.
2. Arnold, F. (1998). Directed evolution, *Nature Biotechnology*, **16**: 617–618.
3. Arnold, F. (2001). Combinatorial and computational challenges for biocatalyst design. *Nature*, **409**: 253–257.
4. Barnett, L. *Ruggedness and neutrality: the NKp family of fitness landscapes*. In C. Adami, R. K. Belew, H. Kitano, and C. E. Taylor, editors, Alive VI: Sixth International Conference on Articial Life, pages 18-27, Cambridge MA, 1998. MIT Press.
5. Davidor, Y. (1991): "*Epistasis Variance: A Viewpoint on GA-Hardness*". In: Foundations of genetic algorithms, ed. G.J.E. Rawlins, Morgan Kaufmann Publishers, pp. 23–35.
6. Grefenstette, J.J. (1992). *Deception considered harmful*. Foundations of Genetic Algorithms, 2. Whitley, L. D., (ed.), Morgan Kaufmann, 75--91.
7. Jones, T. and S. Forrest. *Fitness Distance Correlation as a Measure of Problem Difficulty for Genetic Algorithms*. In L. J. Eshelman, editor, Proceedings of the 6th Int. Conference on Genetic Algorithms, pages 184--192, Kaufman, 1995.
8. Kauffman, S.A. and S. Levin. *Towards a General Theory of Adaptive Walks on Rugged Landscapes*. Journal of Theoretical Biology, 128:11--45, 1987.
9. Kallel, L., Naudts, B. & Reeves, C. (1998). Properties of fitness functions and search landscapes. In *Theoretical aspects of evolutionary computing* (ed. L. Kallel, B. Naudts and A. Rogers), pp. 175-206. Springer, Berlin.
10. Naudts, B. and L. Kallel (2000). A comparison of predictive measures of problem difficulty in evolutionary algorithms, *IEEE Transactions on Evolutionary Computation*, **4**(1):1–15.
11. Reeves, C. R. (1999). Landscapes, operators and heuristic search. *Annals of Operations Research* **86,** 473-490.
12. Stadler, P.F. Towards a Theory of Landscapes," in Complex Systems and Binary Networks, (R. Lopez-Pena et al, eds.), Berlin, New York, pp. 77–163, Springer Verlag, 1995.
13. Stadler, P. F. (1996). Landscapes and their correlation functions. *J. Math. Chem.* **20,** 1-45.
14. Wright, S. (1932). The roles of mutation, inbreeding, crossbreeding and selection in evolution. In *Proc. Sixth Int. Conf. Genetics*, vol. 1 (ed. D. F. Jones), pp. 356-366.