Barrier Trees For Search Analysis

Jonathan Hallam and Adam Prügel-Bennett

Southampton University, Southampton SO17 1BJ, UK
jbh02r@ecs.soton.ac.uk, apb@ecs.soton.ac.uk

The development of genetic algorithms has been hampered by the lack of theory describing their behaviour, particularly on complex fitness landscapes. Here we propose a method for visualising and analysing the progress of a search algorithm on some hard optimisation problems using barrier trees. A barrier tree is a representation of the search space as a tree where the leaves of the tree represent local optima and nodes of the tree show the fittest saddle points between subtrees. The depth of the leaves and nodes corresponds to the fitness of the local optima and saddle points. Each configuration in search space can be mapped to a position on the tree depending on which optima are accessible without ever decreasing the fitness below its starting value. Having computed a barrier tree we can easily visualise how any search algorithm explores the search space.

Barrier trees have been used in the study of many physical and chemical systems, such as disordered spin systems and bio-polymer folding. A generalisation of these trees, and a formal definition, is given by Flamm et al.[1]. Conventionally barrier trees have their leaves (fittest configurations) at the bottom of the tree. In building a barrier tree we need to specify the neighbourhood structure that we are interested in. Here, we have used a Hamming neighbourhood. The novelty of our approach is to consider every configuration in the problem space as a point on the barrier tree. This allows us to see how the search space is being explored. For a genetic algorithm each member of the population can be represented as a point on the barrier tree. At each generation these points move around (usually down) the tree.

Figure 1 shows two example barrier trees. The trees have been truncated above the root – all configurations above this fitness are mapped to a single point on the tree. The two barrier trees are for two classic NP-hard problems, the binary or Ising perceptron and MAX-3-SAT. As can be seen, there are clear differences visible between different types of problems. Various explanations for poor or good performance can be thought of after watching a genetic algorithm solving a problem. The majority of the fitness landscape has a worse fitness than the root of the tree, and so only a fairly small proportion is mapped to anywhere else on the tree. In the examples we have looked at, which admittedly are all fairly small problems, genetic algorithms very quickly (approximately ten iterations) descend into the tree, and then take a lot longer to reach the optimal solution.

The barrier tree allows us to see the influence of changing parameters of the search algorithm. For example, we can see what happens as we change the mu-

[1] Christoph Flamm, Ivo L. Hofacker, Petr F. Stadler, and Michael T. Wolfinger. Barrier trees of degenerate landscapes. *Z. Phys. Chem.*, 216:155-173, 2002.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2724, pp. 1586–1587, 2003.

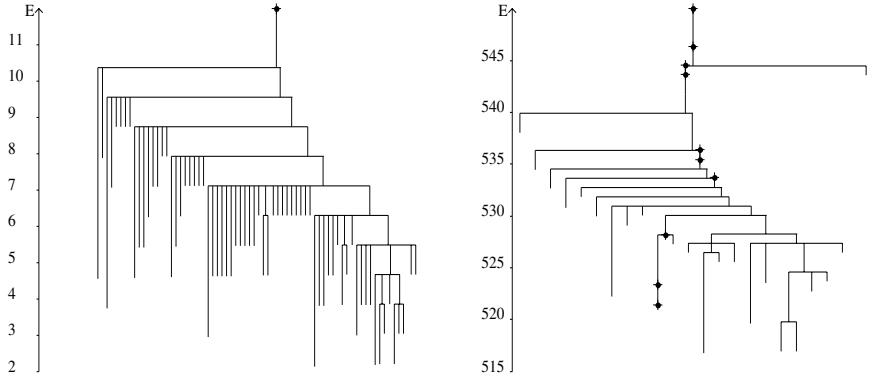

Fig. 1. Some example barrier trees, on minimisation problems. L.h.s a 17 variable binary perceptron training problem, with 1000 training patterns. The entire genetic algorithm population is plotted at the root of the tree, as all the members are less fit than the least fit saddle point, and can descend into any basin. R.h.s a seventeen variable MAX-3-SAT problem, with 4675 clauses. After seventy iterations, the genetic algorithm has descended into the tree, and has become trapped in a basin. It takes another seventy iterations to find a better solution

tation rate or use a different type of crossover. In addition, we can examine how the evolution changes if we use a deme GA (do the demes examine different local optima) or some other population structure. As well as visualising the landscape we can study the evolution systematically. For example, we can determine the probability of reaching the different local optima using a hill-climber or we can study the probability of jumping between branches of the tree if we crossover two parents from the same subtree.

One limitation of using barrier trees is that we need to enumerate all solutions in the search space. This limits the problems we can investigate to around twenty-five binary variables (around 35 million total configurations). However, for some problems it is possible to efficiently enumerate all the configuration above a certain fitness. We can then construct the barrier tree (or barrier forest) for these fit configurations. This should allow us to investigate much harder problems, although we have still to fully implement this enhancement.

Barrier trees provide a new insight into how search algorithms explore complex fitness landscapes. We hope that these investigations will encourage the development of a theoretical framework which is applicable to hard optimisation problems – that is, those problems of most interest for practitioners. Our preliminary investigations suggest that this approach yields many insights about how genetic algorithm explore complex landscapes.