Learning the Large-Scale Structure of the MAX-SAT Landscape Using Populations

Mohamed Qasem and Adam Prügel-Bennett
School of Electronics and Computer Science, University of Southampton, SO17 1BJ, UK.

***Abstract*—A new algorithm for solving MAX-SAT problems is introduced which clusters good solutions, and restarts the search from the closest feasible solution to the centroid of each cluster. This is shown to be highly efficient for finding good solutions of large MAX-SAT problems. We argue that this success is due to the population learning the large-scale structure of the fitness landscape. Systematic studies of the landscape are presented to support this hypothesis. In addition, a number of other strategies are tested to rule out other possible explanations of the success.**

Preliminary results are shown indicating that extensions of the proposed algorithm can give similar improvements on other hard optimisation problems.

***Index Terms*—Satisfiability, SAT, MAX-SAT, Hill Climbing, Clustering, K-Means.**

I. Introduction

CAN a population-based algorithm take advantage of global information about the fitness landscape provided by its members to help it solve an optimisation problem in a way that cannot be achieved by a local (solo) search algorithm? This is a common assumption of many users of evolutionary algorithms. However, there is rarely strong evidence that this is the case. Of course, there may be other important ways in which a population might be beneficial. For example, using a population may be advantageous because, by searching different parts of search space, it quickly finds promising regions where it can concentrate its search effort. Also, crossover may be beneficial as a macro-mutation which naturally anneals itself as the population converges. These other benefits may be very significant, but they are different to and arguably less exciting than the possibility that a population can learn about the large-scale structure of a problem, and then exploit this information to find superior solutions. Although there are a few artificially constructed problems that demonstrate that populations can in principle gain a significant advantage by learning properties of the landscape (e.g. [1], [2], [3], [4], [5], [6], [7]), there has been little unambiguous evidence that this is the case for any naturally occurring optimisation problem. In this paper, we present an algorithm which we will argue does precisely this for one of the classic combinatorial optimisation problems, MAX-SAT.

MAX-SAT is a generalisation of the well-known SAT decision problem. We are given a set of m clauses made up of n Boolean variables $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ where $X_i \in \{\text{true}, \text{false}\}$. The clauses consist of a disjunction of, usually a small number of, literals (a literal being either a variable, X_i, or its negation, $\neg X_i$). In this paper, we use the convention that each clause consists of a different set of literals. In SAT, the problem is to decide whether there exists an assignment of the variables which will satisfy every clause (that is, so that at least one literal in each clause is true). While in MAX-SAT the problem is to find an assignment which maximises the number of true (satisfied) clauses. If we restrict the number of variables in each clause to be K, then the problem class is known as K-SAT or MAX-K-SAT. Throughout this paper we will restrict our attention to MAX-3-SAT, which is the best studied class of MAX-SAT problems. If $K \geq 3$, SAT is known to be NP-complete (in fact, it is the archetypal problem of this class, being the first problem that was shown to be NP-complete). Since a solution to MAX-SAT would also provide an answer to the question of whether all the clauses were satisfiable, MAX-k-SAT is NP-hard (even for $k = 2$, although 2-SAT is actually in P). We study instances of the problem that are believed to be hard to solve.

The algorithm we present here is a hybrid algorithm. We find many good solutions using a local neighbourhood search algorithm. The solutions are clustered using a K-means clustering algorithm. The solution closest to the centroid of each cluster is then used as a starting position for applying a second round of the local neighbourhood search algorithm. This very simple algorithm finds remarkably good solutions—we describe our tests of the algorithm in section IV.

Our interpretation of why this algorithm performs so well is that the fitness landscape (which we take to be the landscape using the Hamming distance as a metric) consists of a few global maxima (assignments of the variables that maximise the number of satisfied clauses) positioned at different locations in the search space. The global maxima are correlated, but not strongly. Around each global maximum there is a 'galaxy' of local maxima. The closer the local maxima are to a global maximum the more likely they are to have high fitness values. Although these local maxima tend to be clustered around global maxima, they can still be quite far in Hamming distance from them. For example, they may differ in 30–40% of their variables, which would mean in a 1000 variable problem they would be at a Hamming distance of 300–400 from a global maximum. We postulate that our local search algorithm finds good solutions (close to, if not at, a local maximum). By clustering, we pick out good solutions centred around a global maximum (or, at least, around some very good local maximum). By taking the solution closest to the centroid we move closer to the centre of the galaxy where the high quality solutions are to be found. These centroid solutions usually are not very good, because the search space is extremely rugged. However, by performing a local neighbourhood search we end

up finding a solution which is almost always superior to the previous solutions we found. The major contribution of this paper is to present evidence to support this picture.

The rest of the paper is organised as follows. In the next section, we briefly discuss MAX-SAT, and describe how we generate the instances used in our tests. We briefly discuss the local neighbourhood search methods used to solve this problem. This is followed in section III by a presentation of some studies on the landscape properties of small problems. Section IV presents the results of a number of different algorithms on much larger problem instances. In section V, we present preliminary results on other NP-hard optimisation problems. We discuss the results and draw conclusions in section VI.

II. MAX-SAT

MAX-SAT is one of the best studied optimisation problems—in part because of its association with SAT, which, besides from its theoretical importance, has a huge number of practical applications. A large amount of research has gone into characterising the typical behaviour of random instances. In this paper, we also concentrate on random *fixed-length clause* instances [8]. These consist of a set of m clauses where the clauses consist of $K = 3$ literals (we take this to be a strict set rather than a multiset, so that no clause is repeated). The literals in any clause all involve different variables. Every allowable clause is chosen with equal probability. In practice, we generate the clauses by randomly choosing 3 variables from the n possible variables, excluding repeats. With a probability of one-half we either negate the variable or leave it as it is. The clause is added to our set of clauses provided it is not already a member of the set.

For this class of problems there is a transition between the case where most problems are satisfiable to the case where most problems are unsatisfiable, which occurs at a ratio of clauses to variables of $\alpha = m/n \approx 4.3$. This transition becomes increasingly sharp as the problem size increases, and is viewed as an example of a classic first order phase transition. At around the same ratio of clauses to variables, there is an observed change in the difficulty of problem instances. Below the phase transition most problems are easy to solve while above the phase transition most instances are hard to solve [8], [9]. That is, the empirical time complexity for most complete SAT solvers grows dramatically around this phase transition. For larger ratios of m/n, the 3-SAT decision problem typically becomes easy again, because it is straightforward to prove unsatisfiability. However, the MAX-SAT problem typically remains hard [10], [11] in this regime.

The phase transition has been investigated using statistical mechanics approaches [12], [13]. Although these are not rigorous, there is a region around the phase transition where the calculation is believed to be exact in the limit $n \to \infty$ (at least, it passes several stringent self-consistency tests, and it gives predictions in agreement with carefully conducted simulations). These calculations are also in agreement with rigorous bounds for the location of the phase transition [14]. For small values of $\alpha = m/n$, the problem has a simple landscape corresponding to one very large cluster of satisfied solutions which is easily reached by hill-climbing. Around the phase transition, the statistical mechanics calculation undergoes a, so called, one-step replica-symmetry breaking that is a signal for the existence of many local maxima weakly correlated with each other. Away from the phase transition, one-step replica-symmetry breaking no longer holds, and it is postulated that the system enters a state of full replica-symmetry breaking [15], [16]. Although there is no solution of the behaviour in this region, full replica-symmetry breaking is taken to be an indication of complex clustering of the local optima [17]. In this paper, we focus on random instances with $\alpha = m/n = 8$. This is deep in the hard phase for MAX-3-SAT where full replica-symmetry breaking is believed to hold (similar results are obtained at $\alpha = 4$, 6 and 10). To investigate the structure of the fitness landscape we have carried out extensive empirical studies.

Before discussing the landscape structure we briefly discuss the local neighbourhood search methods we use for the empirical investigations. Throughout most of the paper we use a basic hill-climber (BHC) where we randomly choose a variable, and flip it if doing so does not increase the number of unsatisfied variables. This algorithm produces a very rapid initial improvement in the cost. The most well known local-search algorithm for SAT or MAX-SAT problems is GSAT [18]. This exhaustively searches the entire neighbourhood before choosing the move that gives the greatest increase in fitness. As this will eventually get trapped in a maximum, a modification known as WALKSAT has been proposed which, with a probability p, picks an unsatisfied clause and flips a variable in it, otherwise it performs a GSAT step [19]. If these algorithms are efficiently coded they out-perform BHC when run sufficiently long. In section IV-D we compare our algorithm with state-of-the-art implementations of GSAT and WALKSAT.

III. LANDSCAPE OF MAX-3-SAT

In this section, we present some empirical observations on the fitness landscape of MAX-3-SAT for $\alpha = m/n = 8$. These were carried out as part of a broader investigation of the landscape of MAX-3-SAT, but here we only present results relevant to our thesis. We studied instances up to size $n = 100$ by finding many local maxima. To achieve this we used BHC starting from different, randomly-chosen, starting points. To ensure that we had found a local maximum we use the following algorithm. After running the hill-climber with no improvements in many attempts we switched to an exhaustive search method that checked all neighbours at the same cost as the current point. Then we checked their neighbours repeatedly, until either a fitter solution was found, or else all neighbours at the current cost had been searched, in which case we can be sure that that the hill climber has reached a local maximum. By performing multiple searches on the same instances, we were able to measure statistical properties of a local maxima. A common feature of all the instances that we investigated was that the higher the fitness of the local maximum the more likely we would find it. As

a rule-of-thumb, we observed that the likelihood of finding a local maximum roughly doubled each time we satisfied one more clause. This result is not so surprising as it is easy to imagine why better local maxima could typically have larger basins of attraction than less fit local maxima.

What makes MAX-3-SAT instances hard is that there are many more local maxima than global maxima. Thus, even though the basins of attraction appear to be largest for the global maxima, nevertheless, we are more likely to get trapped in a lower-fitness local maximum, because there are many more of them. The number of local maxima appears to increase exponentially with the size of the instances, which makes finding a global maximum increasingly less likely as the instance size becomes large. The exponent describing the exponential growth is, however, rather small so even for systems of size 100 finding a global maximum is not difficult (for $n = 100$ we found the probability of BHC reaching one of the quasi-global maxima to be approximately 0.25). At least, for problems up to this size we were able to find the fittest local maxima multiple times. We postulate that these are the global maxima, since if there was even a single maximum fitter than those we found, then we would expect to find it with high probability given the number of hill-climbs we made (unless it had an atypically small basin of attraction). We call our best maxima found in this way, *quasi-global maxima* as we believe them to be the true global maxima, although we have no proof of this. (For small problems, $n \leq 50$, we could find the true global maxima using a branch-and-bound algorithm. In every case, the best solution found by performing multiple BHC runs were true global maxima. We also tested problems with $n = 100$ from SATLIB, and in every case we were able to find the best solution for the problem using BHC). Note that if we were to look at much larger-sized problems, then we would find each best solution only once or a very few times, in which case we would have no grounds to argue that these are likely to be the global maxima. The fact that we believe we can find all global optima for relatively large instances makes this problem class very rich to study empirically.

We have studied the structure of the configurations making up the quasi-global maxima. These were found to form a small number of connected clusters typically separated from each other by a large Hamming distance (anywhere between 10–60 for instances of size 100). The separation between connected clusters varied considerably between problem instances. In figure 1, we show the frequencies of Hamming distances between all quasi-global maxima averaged over 300 problem instances. To find the set of quasi-global maxima we ran BHC followed by an exhaustive search 5000 times. The histogram has a large peak at a Hamming distance approximately equal to 5% of the total number of variables. This indicates a clustering of quasi-global maxima around each other. However, the histogram has a large tail with a second peak at a large Hamming distance away from the first. This is indicative of multiple clusters that are weakly correlated with each other (if there was no correlation then the clusters would be at a Hamming distance of $n/2$).

To demonstrate that the histogram is consistent with this picture, we generated clusters using the following procedure.

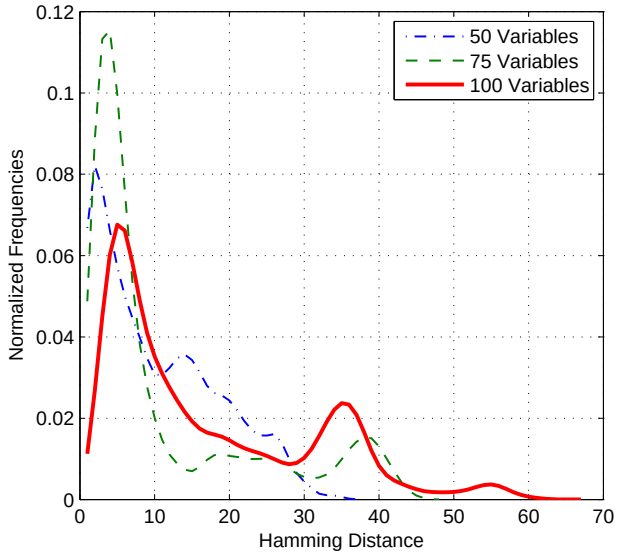

Fig. 1. Shows a histogram of the Hamming distance between quasi-global maxima for 50 instances of size $n = 50$, 75 and 100 variables and with a clause to variable ratio of $\alpha = 8$. There is a cluster of very close global optima below a Hamming distance of 10. Also, a significant number of global optima are found at Hamming distances equivalent to 30–40% of the variables.

We chose a centre $\boldsymbol{C} = (C_1, C_2, \ldots, C_n)$ where $C_i \in \{\text{true, false}\}$, and a second centre $\boldsymbol{C}' = (C'_1, C'_2, \ldots, C'_n)$ was generated from the first by randomly changing k variables, where k is a uniformly distributed integer between 0 and $3\,n/4$. Thus, on average, $\boldsymbol{C}$ and $\boldsymbol{C}'$, are separated by a Hamming distance $3n/8$. We then generated between 20 and 220 random strings centred around each of the two centres at an average Hamming distance of $n/10$. We then computed the correlation between all pairs of randomly chosen strings. This was then averaged over 100 samples. The histogram of correlations is shown in figure 2. We observe a very strong similarity in the structure of this figure and figure 1, which lends support to the hypothesis that the quasi-global solutions are themselves clustered around a few centres in the way described.

To illustrate how the lower-cost local maxima are clustered relative to the quasi-global solutions, we measured the Hamming distance between the local maxima and the nearest quasi-global solution. Histograms of this Hamming distance are shown in figure 3. In these figures, we consider only those local optima at a cost of 4 and 8 away from the quasi-global maximum cost. We note that the higher-cost solutions are closer on average to a quasi-global maxima than lower-cost solutions.

Figure 4 shows how the average Hamming distance between the local maxima and the nearest quasi-global maxima varies as a function of the difference in the cost between the local maxima and the quasi-global maxima. (By scaling both axes by $1/n$ these curves appear to collapse onto a universal curve—we have chosen, however, to present the raw data rather than scaled data so as not to over-complicate the story). It is easy to understand why higher cost solutions should be

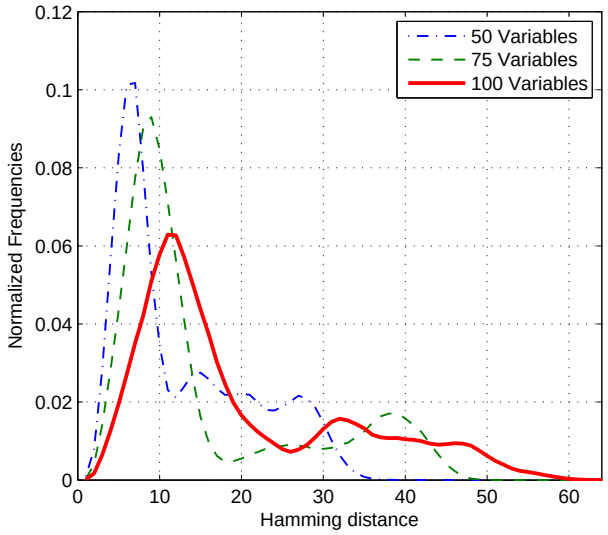

Fig. 2. Shows a histogram the Hamming distance between randomly chosen points forming two clusters. The distance between each cluster is taken to be a uniformly chosen random variable between 0 and $3\,n/2$. The graph is generated by averaging over 100 samples. We note the strong similarity between this and figure 1.

closely correlated on average with the quasi-global maxima since globally-optimum solutions represent good ways of maximising the number of satisfied clauses. Therefore, nearby solutions are also likely to satisfy many clauses. However, what is perhaps more surprising is that even the solutions whose costs differs by one from the quasi-global optima have a high average Hamming distance from any quasi-global optima. Even for relatively small problems with 100 variables this average Hamming distance is around 18, which is sufficiently large that the probability of a stochastic hill-climber reaching a global maximum from a local maximum is negligibly small. To be more explicit, if the Hamming distance between a local maxima and a better solution is k, then a local search algorithm would typically have to explore every assignment up to a Hamming distance k before finding a better solution. The number of solutions in a ball whose Hamming radius is strictly less than k is

$$\sum_{i=0}^{k-1} \binom{n}{i}$$

which for $n \gg k$ is $\Theta(n^{k-1}/(k-1)!)$ (even for small instances with $n = 100$ and $k = 18$ this is approximately 8×10^{18}). Within this "Hamming ball of radius k" there will be no solution better than the current solution (since by assumption the closest better solution is a Hamming distance k away). Thus, there is no heuristic information to exploit. There may be solutions of the same cost in this Hamming ball which are closer to the global solution, but there is no way of knowing whether it is closer to or further from a better solution than the current solution.

Although it is always dangerous to rely on low-dimensional pictures to understand what happens in high-dimensional space, nevertheless we offer the following caricature of our

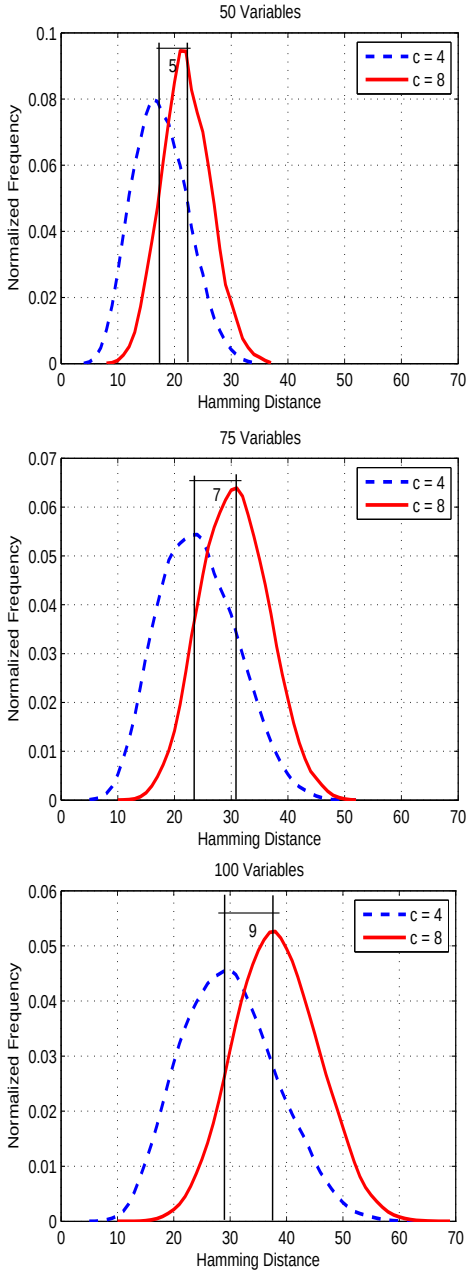

Fig. 3. Shows a histogram of the Hamming distance between the quasi-global maxima and local maxima with fitness 4 and 8 below the quasi-global maxima. As the number of variables increase, the Hamming distance to the quasi-global maxima also increase.

fitness landscape. We imagine the search space as being points on a 'world' where the heights of the points represent the fitness values. This is schematically illustrated in figure 5. The good solutions lie in mountain ranges. The mountain ranges have hugely more foothills than high mountains. There are only a few mountain ranges in this world, and they are slightly correlated (e.g. all the mountain ranges might lie in one hemisphere). The mountain ranges occupy only a very small proportion of the world. As with real mountain ranges, higher solutions tend to lie in the middle of the mountain ranges. Starting from a random position and hill-climbing we are likely to land up at a foothill, just because there

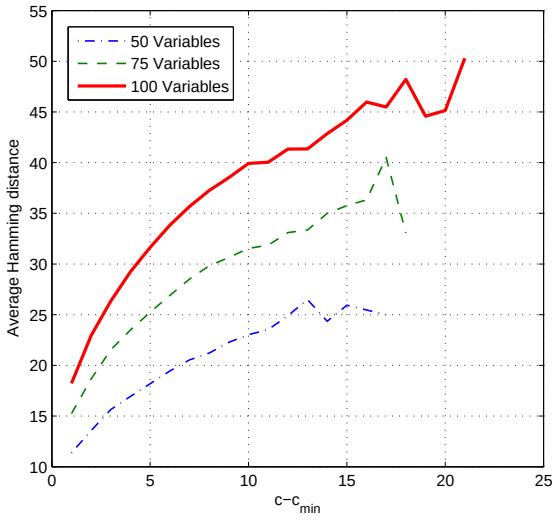

Fig. 4. The average Hamming distance between the quasi-global maxima and the local maxima. As the gap in fitness between the quasi-global maxima and local maxima decreases so does the average distance to the quasi-global minimum.

are so many of them. Finding a good solution through hill-climbing alone will be very difficult. An alternative strategy is to perform a large number of hill-climbs starting from different randomly-chosen positions. We could then take the average of the solutions we find. This will put us in the centre of the hilly hemisphere. Although, we are unlikely to be at a peak. If we then perform a hill-climb we are more likely to find a superior solution than if we started from a random position. However, we can do even better by clustering the solutions we find after performing hill-climbing. If we are lucky, a cluster will correspond to a mountain range. The centres of the clusters correspond to the regions with many high mountains. So if we restart hill-climbing from the centre of a cluster we have a very good chance of finding a high quality solution. Of course, this picture fails in many ways. The search space is not continuous, but discrete. Furthermore, using a Hamming neighbourhood, the topology of the search space is an n-dimensional hypercube. The high-dimensionality makes it harder for low-cost solutions to be local maxima, since they have a large number of neighbours. Also the set of costs is discrete so that there is no gradient information. Nevertheless, as we will see, an algorithm based on clustering seems to perform very well, which suggests that this simple picture might not be too misleading.

IV. Experimental Results

In this section we make a number of different comparisons of our algorithm. In subsections IV-A and IV-B, we define the algorithm we use, and compare its performance with modified versions of the algorithm. The purpose of this is to provide support for our hypothesis that the success of the algorithm is attributable to it 'learning' the large-scale structure of the landscape. In subsection IV-C, we perform a comparison between a hybrid genetic algorithm and our

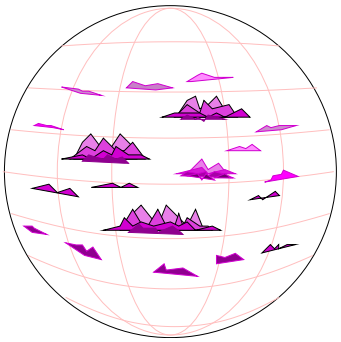

Fig. 5. Caricature of the Fitness Landscape showing the clustering of good solutions.

K-means algorithm as a function of CPU time. The aim is to show that K-means not only gives an initial short lived improvement, but that the advantage survives even after a long period of search. This is consistent with our hypothesis that clustering puts the searchers into a better part of the search space. Finally, in subsection IV-D, we compare our algorithms with state-of-the-art implementations of local search methods by using the UBCSAT program[1]. Although, the aim of our research is to present a new search strategy rather than produce the best algorithm for a particular problem, nevertheless, our algorithm out-performs all algorithms implemented by the UBCSAT program.

A. Experimental Setup

We generated random MAX-3-SAT instances using the method described in section II. We considered problem instances ranging in size from 6 000 to 18 000 variables in increments of 2000 variables, and with $\alpha = m/n = 8$. These are difficult problems since they are in the over-constrained region. We chose $\alpha = 8$ as this value appears to be the most commonly used value in the MAX-3-SAT literature, although we obtained qualitatively similar results at all other values of α that we tried ($\alpha = 4$, 6 and 10, results not shown). For each increment we generated 100 problem instances.

In all the tests we carried out we started by performing 1 000 hill-climbs starting from different random configurations. We used the basic hill-climbing strategy. The number of iterations used on problems with n variables was $T(n) = 5\,n/2 + 5000$. That gives 20 000 iterations for 6 000 variables and 50 000 for 18 000 variables. These numbers were chosen after experimentation as they gave good quality solutions. We increased the number of iterations with the size of the problem to give more opportunities for the larger problems to find good quality solutions, since it has been shown that the time to reach a local maxima grows with the problem size [20]. However it should be stressed that it was not the goal to necessarily reach a local maximum, but only to find a good solution. The best result for the 1 000 hill-climbs averaged over all 100 problem instances is shown in the second column of table I.

We then tested a number of different strategies to boost the performance obtained from these initial 1 000 points. The

[1] http://www.satlib.org/ubcsat/

TABLE I
COMPARISON OF DIFFERENT ALGORITHMS. COLUMN 1 SHOWS THE PROBLEM SIZE, WHILE COLUMNS 2–6 GIVE THE LOWEST NUMBER OF UNSATISFIED VARIABLES FOUND BY DIFFERENT ALGORITHMS. THESE ARE BHC, BHC+BHC (BASELINE), BHC+K-MEANS+BHC, BHC+AVERAGING+BHC, HYBRID-GA AND BHC+PERTURB+BHC. COLUMNS 7 AND 8 SHOW THE INCREASE IN PERFORMANCE OVER THE BASELINE ACHIEVED BY USING K-MEANS CLUSTERING AND AVERAGING RESPECTIVELY. THE TESTS WERE CARRIED OUT ON RANDOM MAX-3-SAT PROBLEMS WITH $\alpha = 8.0$. EACH TEST WAS PERFORMED ON 100 PROBLEM INSTANCES FOR EACH NUMBER OF VARIABLES.

1 **#Vars**	2 **First BHC**	3 **Second BHC (1)**	4 **K-Means/ BHC (2)**	5 **Average/ BHC (3)**	6 **hybrid-GA**	7 **Perturb/ BHC**	8 **(2)-(1)**	9 **(3)-(1)**
6000	1971.77	1448.35	1370.61	1385.82	2429.5	1447.92	77.74	62.53
8000	2944.03	2037.26	1913.26	1943.38	3691.22	2038.78	124	93.88
10000	3464.7	2614.65	2456.67	2507.56	4908.87	2617.19	157.98	107.09
12000	4235.8	3247.74	3051.09	3125.79	6218.57	3247.4	196.65	121.95
14000	4999.14	3892.06	3652.23	3761.51	7533.33	3895.38	239.77	130.55
16000	5711.81	4496.69	4226.15	4368.23	N/A	N/A	270.54	128.46
18000	6551.83	5256.28	4932.41	5129.12	N/A	N/A	323.87	127.16

testing procedure we carried out is shown schematically in figure 6. As a baseline we repeated the basic hill-climber for another $T(n)$ steps on all 1 000 search points. These results are shown in the third column of table I. This second round of hill-climbing shows that the solutions found in the first round were still some way away from being locally optimal.

B. K-Means and "Averaging"

We next performed clustering using the K-means clustering algorithm [21] on the 1 000 search points found by the initial hill-climbing. This algorithm starts by assigning a random string on the n-cube to each of K initial "centres" (note that, in this section, K is used to denote the numbers of centres in K-means clustering and should not be confused with the number of variables in each clause). Each of the 1 000 points is then assigned to the cluster with the nearest centre. The centres are then updated to be the configuration which best represents the points in the cluster, in the sense that it minimises the mean Hamming distances to the set of points in the cluster, $\mathcal{C}$, i.e.

$$\boldsymbol{X} = \operatorname*{argmin}_{\boldsymbol{X}} \frac{1}{|\mathcal{C}|} \sum_{\boldsymbol{Y} \in \mathcal{C}} H(\boldsymbol{X}, \boldsymbol{Y})$$

where $H(\boldsymbol{X}, \boldsymbol{Y})$ is the Hamming distance between configurations. The points are reassigned to the nearest centroid and the process is repeated until there are no changes. This usually happens after five to ten iterations. The computational cost of K-means clustering is small compared with the time required to do hill-climbing. Once the centroids have been computed, a new starting point is found by rounding each component of the centroid to obtain a feasible solution.

In the results we in table I, we used $K = 100$ clusters. This was decided after a small amount of experimentation. This is probably not optimal, but fits with our decision not to fine tune our algorithm. A second round of hill-climbing is carried out from the solutions obtained from the 100 centroids. The results obtained after this procedure are shown in the forth column of table I. In every case there is a considerable gain in performance compared to the baseline, even though the baseline involved considerably more work (because the second round of hill-climbing reported in column 3 of table I was carried out on all 1 000 points rather than 100 used in the K-means clustering algorithm). The gain in performance compared to the baseline is shown in column 8 of table I.

We have compared clustering with 'averaging', where we randomly selected 10 points and find the centroid of the group (for this problem the centroid can be found by taking the average assignment of each variable and rounding). This was repeated 100 times to give 100 centroids so as to give a fair comparison with the K-means clustering method. A second cycle of hill-climbing is then carried out. The results are shown in the fifth column of table I. This again produced a substantial gain in performance compared with the baseline (the gain is shown in the last column of table I), however, these gains are smaller than those obtained by K-means clustering, particularly for large number of variables. This provides further empirical support for the claim that the global maxima are clustered (although we have shown clustering for instances of size 100, these results are for much larger instances). It also shows that even the mean of all the good solutions provides a much better starting point than a random starting point.

We want to show that these results are *not* due to clustering or averaging acting as a macro-mutation which allows the search to escape out of local maxima. To do so we applied perturbations of 0.1%, 1%, 2%, 5% and 10% of the variables, and then repeating hill-climbing. We found that doing this gave us worse performance than the baseline algorithm. Even with 0.1% the perturbation appears slightly detrimental (see column 7 of table I). These results are not so surprising, since it is clear from comparing the results of the baseline algorithm with the results after the first hill-climb (columns 2 and 1 respectively) that we are far from being stuck in a local maxima.

As a final test, we compared our algorithm against a hybrid genetic algorithm. The hybrid genetic algorithm combined hill-climbing with selection and two-parent crossover. A population of 100 individuals was used. We used Boltzmann selection where we chose each member of the population with a probability proportional to $\exp(-\beta F_i/\sigma)$ where F_i is the fitness of individual, i; σ is the standard deviation of the fitness values in the population; and β controls the selection strength. Various values of β were tried, but this did not strongly affect the results. Uniform, single-point and multi-point crossovers

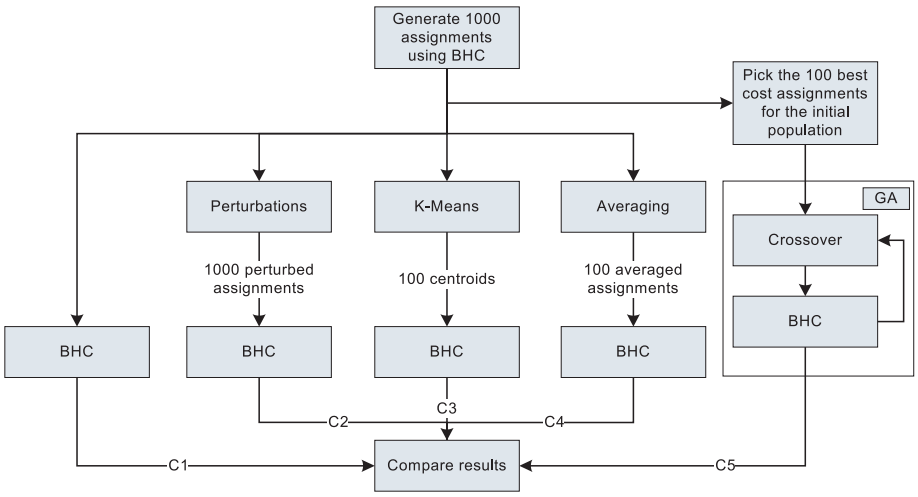

Fig. 6. Schematic diagram of the set of tests carried out and reported in table I.

were tried. The best results were obtained with single-point crossover. Column 6 of table I shows the best results we were able to obtain using a GA. Although we do not claim that all the parameters were optimally chosen, the results obtained by the hybrid-GA are disappointing compared to the other algorithms. The reason for this is, in part, due to the fact that the GA was not given sufficient time to converge. In the next subsection, we analyse the performance of the algorithms when they are run for longer times. Even then, we will see that the K-means clustering approach has a considerable advantage over a GA.

This may seem surprising as two-parent crossover might superficially appear to be doing something similar to averaging, however it is important to appreciate the difference. This is easily seen by considering a simple example. Consider a unitation problem consisting of a binary string $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ with $X_i \in \{0, 1\}$, where the fitness is a function of the number of 1's in the string. Defining the proportion of ones as $m = \sum_{i=1}^{n} X_i/n$, the fitness is given by

$$F(\boldsymbol{X}) = \begin{cases} m & m < m_1 \\ m_1 & m_1 \leq m < m_2 \\ m - m_2 + m_1 & m_2 \leq m. \end{cases}$$

This is shown in figure 7 for the case when $m_1 = 0.75$ and $m_2 = 0.95$.

We call this the Iceberg problem because the configurations with $m > m_2$ can be viewed as a small iceberg in a large ocean of solutions with cost m_1. For large n a hill-climber starting from a random string with fitness close to 0.5 will climb the slope until it reaches a state where 75% of the variables are 1's. When it reaches the plateau it has no heuristic information it can use. As the density of states falls off very fast, as a function of the number of 1's, the hill-climber will, with high probability, lie close to the edge of the plateau. That is, approximately 75% of the variables will be equal to 1. A population of hill-climbers will also lie very close to the edge of the plateau. If we were now to perform crossover on two individuals then again with high probability the child

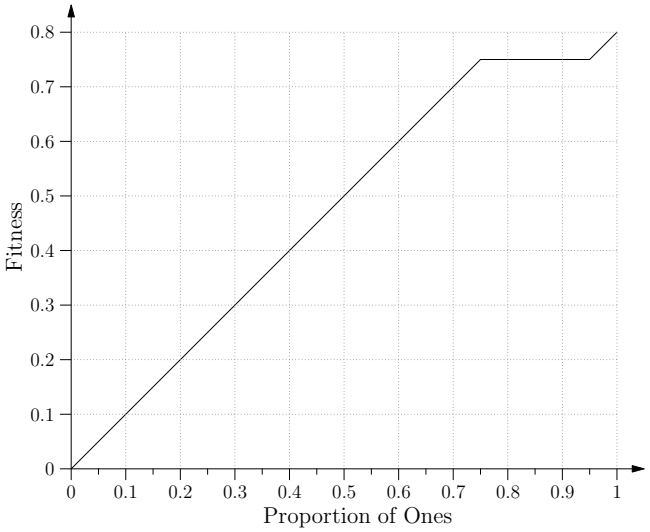

Fig. 7. Fitness function of the Iceberg problem. This problem is easily solved by averaging good solutions, but very hard for a hill-climber or genetic algorithm.

would have approximately 75% of its variables equal to 1. This would be a slightly more efficient way to explore the plateau than hill-climbing alone as most random mutation moves will, on average, move away from the all 1's string (since there are more 1's than 0's a random mutation is more likely to attempt to change a 1 to a 0 rather than a 0 to a 1). Crossover, by contrast, does not change the number of 1's on average. This concentration effect of crossover has been discussed in closely related models previously [1], [2], [3], [4]. Nevertheless, it still takes an exponential amount of time to find the global maximum using crossover. In contrast, if we average a population of say 100 individuals that have undergone hill-climbing, and round up to 0 or 1, then, for any reasonable size problem, the resulting solution will, with overwhelming probability, consist of the all 1's string.

Clearly, this is a contrived problem. Its purpose is to demonstrate that averaging is very different to crossover. This

is true even if we used multi-parent crossover [22] or a univariate estimation of distribution algorithm (EDA), where, despite averaging, the expected time to solve the problem shown in figure 7 would still grow exponentially. Clearly, the landscape of MAX-SAT is much more rugged on short length scales than the Iceberg problem. On very large length scales, the landscape of MAX-SAT differs because it possesses multiple global maxima some distance apart. However, on some intermediate scales this model appears to capture some important properties of the landscape of MAX-SAT—that is, very fit solutions lie at the centre of more easily found local optima.

C. Temporal Behaviour

In the section above, the behaviour of the hybrid genetic algorithm was particularly poor. This was due to the limited number of BHCs allowed for each algorithm. When given a longer time the hybrid-GA performs considerably better. In figure 8, we show the average performance of parallel-BHC (that is, we run several BHC in parallel, and report the best cost in the population), K-means clustering and the hybrid-GA. Each algorithm was run for 3 minutes and the results were averaged over 100 instances of randomly generated MAX-3-SAT. The instances consisted of 6000 variables at $\alpha = 8$. In parallel-BHC, we run 10 BHCs in parallel and show the best of these (10 runs were chosen as it appeared to give good performance in preliminary tests). K-means clustering was run starting with an initial population of 100 where we performed 27 000 hill-climbing steps before performing K-means clustering with $K = 10$ clusters, and then running BHC starting from the 10 centroids. No tuning was performed on the K-means clustering algorithm. Finally, we tested a hybrid-GA with a population of size 10 where we performed uniform crossover, Boltzmann selection with a selection strength of $\beta = 0.1$ and BHC. The parameters for the hybrid-GA were chosen after performing a large number of preliminary tests. As can be seen the GA outperforms BHC given enough time, but does not beat K-means on average, (although in some instances it does).

For larger problem instances the efficiency of K-means becomes more pronounced, so that for problems with 18 000 variables run for 5 minutes K-means gave better performance than a hybrid-GA on every one of 50 instances that were tested. These results demonstrate that the benefit of performing K-means clustering persists even after some time. We attribute this to the fact that K-means has moved the searcher to a part of the search space where there are more high quality solutions.

D. Comparison with Local Search Algorithms

Our algorithm performs well on large problem instances. This makes it difficult to compare with previous results reported in the literature, which tend to concentrate on small instances. The only work we are aware of which studied similar sized instances statistically (i.e. gave results of multiple runs on multiple problem instances) where those given by Zhang [23]. Our algorithm substantially out-performs the

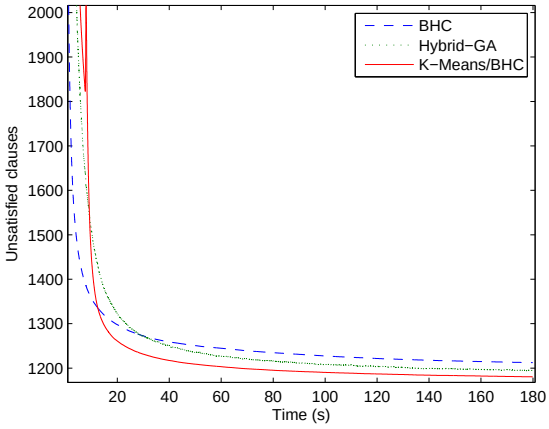

Fig. 8. Comparison of BHC, genetic algorithms and K-means clustering as a function of CPU time run on 100 instances of randomly generated MAX-3-SAT instances with 6000 variables at $\alpha = 8$. The large jump in fitness in the K-means algorithm after around 10 seconds marks the point where K-means clustering is carried out.

results obtained by Zhang. To provide a comparison with state-of-the-art local search algorithms we have compared our algorithms to those implemented in UBCSAT[2], which provides a fast implementation of a range of modern algorithms for MAX-SAT.

We tested all algorithms provided by UBCSAT using their default settings. Results are given on 5 randomly drawn MAX-SAT instances with 18 000 variable and $\alpha = 8$. Each algorithm is run five times. We report results for the five best algorithms from UBCSAT on these instances. These algorithm were; SAMD, IROTS, HWSAT, GWSAT, and GSAT. In each case, we run for 5 minutes. The performance is compared with BHC run for 3.5 minutes and two K-means runs. In both cases we run 100 BHCs for 200 000 iterations and then used K-means to find 5 centroids (again the choice of 5 centroids was chosen after some preliminary experimentation). This stage took no more that 15 seconds. We then run a hill-climber on each centroid for 30 seconds (thus the total amount of time spent by these algorithms was 2.75 minutes). The difference between the two tests was that in the first one we used our BHC algorithm after K-means while in the second test we used the GSAT from UBCSAT. The results are shown in table II. As can be seen, K-means substantially out-performs all other algorithms in UBCSAT despite giving them more time.

Most of the algorithms seem to plateau after 5 minutes. However, GWSAT (a fast implementation of WALKSAT) continues to find good solutions. We found that it gave similar solutions as BHC/Kmeans/gsat obtained in 2.75 minutes if it was run for around 1 hour. No doubt some of the algorithms we have tried may have run faster had we optimised their parameters. We have tried to compensate for this by allowing the other algorithms more time. Furthermore, we have not attempted to fine tune the parameters of our own algorithm. The fact that we have obtained such good performance, provides support for our contention that the clustering algorithm

[2] http://www.satlib.org/ubcsat/

Algorithm	Overall average	Time (minutes)
SAMD	3696.4	5
IROTS	3583.2	5
HWSAT	3678.2	5
GWSAT	3636.1	5
GSAT	3667.4	5
BHC	3667.1	3.5
BHC/K-Means/BHC	3572.6	2.75
BHC/K-Means/GSAT	3527.8	2.75

TABLE II
AVERAGE PERFORMANCE OF DIFFERENT SEARCH STRATEGIES ON 5 RANDOM INSTANCES OF 18 000 VARIABLE MAX-SAT PROBLEM WITH $\alpha = 8$.

explores the landscape in a fundamentally different way to existing algorithms.

As a final set of tests we have performed longer runs on larger problem instances, $n = 20\,000$ and $50\,000$, at $\alpha = 6$, 8 and 10. In these experiments, we used GSAT followed by K-means followed by WALKSAT. We run $1\,000\,000$ GSAT 200 times. We then performed K-means clustering with $K = 5$. This was followed by $40\,000\,000$ WALKSAT moves on each of the 5 centroids. We report the best result of the centroid. Although we take the same number of GSAT and WALKSAT moves, the majority of time is spent performing WALKSAT, which takes considerably more time to complete a move than GSAT. We used our own implementation of GSAT and WALKSAT. We compare this with UBCSAT's GWSAT run for $100\,000\,000$ moves and our own implementation of WALKSAT for $1\,000\,000\,000$ steps. Our WALKSAT appears to have the same performance as UBCSAT's GWSAT, but is considerably faster. We give timings for the algorithms run on an Intel Core 2 Quad Q6600 with 4 GB RAM running Windows Vista. We also compare with UBCSAT's IROTS run for $30\,000\,000$ steps. These results are shown in table III. We observe that our algorithm, despite being given considerably less time, out-performs 10^9 iterations of WALKSAT, which in turn outperforms the two top UBCSAT algorithms.

Code running our algorithm is publicly available in a package WINSAT[3]. We have also made available the random instances we used in the experiments reported above. These can be found from the link given.

V. OTHER PROBLEMS

We chose to study the MAX-3-SAT problem as this is one of the best understood NP-Hard problems. An important question is whether we can find similar behaviour on other NP-Hard problems? The essential features of the landscape which made our approach work was that good solutions are, at least weakly, correlated with the globally-optimal solutions. In those optimisation problems whose landscapes have been studied using statistical mechanics techniques there seems to be a general pattern that a phase transition from easy to hard to solve instances is related to the breaking of replica-symmetry, which indicates the appearance of many local optima widely separated in space. The form of this replica-symmetry breaking varies, so that, in some cases, a more complex clustering of solutions occurs. The algorithms proposed here would appear to have potential in finding good solutions for these hard instances. Clearly, though, the performance of these algorithms can only be determined empirically. This is work we are currently undertaking. We present some preliminary indications that these results might generalise by considering two other NP-Hard problems, graph colouring and the binary perceptron (these are solved in their native form, rather than being translated to a MAX-SAT problem).

[3]http://users.ecs.soton.ac.uk/mqq06r/winsat/

A. Graph 3-Colouring

Graph colouring is one of the best studied hard optimisation problems, in part because many practical problems can be expressed as graph colouring problems. The problem is defined on a graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of vertices, and $\mathcal{E}$ is the set of edges. The task is to assign a colour, $c(i)$, to each vertex $i \in \mathcal{V}$ such that the number of colour conflicts is minimised. A colour conflict is an edge $(i, j) \in \mathcal{E}$ such that $c(i) = c(j)$. This is a well known NP-hard problem. If we consider random graphs $G(n, p)$ where n is the number of vertices and p is the probability of an edge, then for a fixed number of colours there is a phase transition from satisfiable to unsatisfiable graphs as we increase p. In a similar manner to MAX-SAT, the decision problem of whether the graph is colourable becomes hard at the phase transition. Above the phase transition the problem of minimising the number of colour conflicts is believed to be hard for most instances.

Graph colouring poses a difficulty when we try to find the centroid of a set of solutions, because the permutation symmetry makes the notion of the centroid unclear. To overcome this problem we compute the centroid as follows. Given a population of solutions we associate a weight with every pair of nodes equal to the number of times the pair was coloured differently in each member of the population. We then find a colouring which minimises the sum of weights whose vertices have the same colour. In this way, the "centroid" will tend to colour nodes differently if they were differently coloured in most of the population. This approach may appear counter-intuitive, as to solve one graph-colouring problem, we are proposing that we solve a second one. However, the second (weighted) graph colouring problem is not necessarily a hard instance—by hill-climbing we quickly reach a state where we cannot improve the solution any more. In practice, the time to solve this single graph colouring problem is insignificant (less than one second) compared with the time we spend on updating the population. In figure 9, we show the lowest number of colour clashes in a population of size 100 versus the total number of function evaluations for 100 random graphs with 1000 nodes, where the probability of an edge is 0.01. We show the results for a hill-climber alone and for a hill-climber where we have performed "averaging" after attempting 3000 hill-climb moves on each member of the population. We note that, just as for MAX-3-SAT, averaging, although initially costly, gives an improvement in performance over hill-climbing alone. Again the obvious interpretation is that "averaging" moves the search into a better part of the search space. It is less obvious how we would cluster solutions for this problem, and we have not attempted to do so.

n	α	Algorithm	Total Number of Flips	Time	UNSAT
20 000	6	GSAT/K-Means/WALKSAT	2×10^8(GSAT) + 2×10^8(WALKSAT)	17.8 min	1 539
20 000	6	UBCSAT GWSAT	10^8	51.0 min	1 602
20 000	6	Our WALKSAT	10^9	1.08 hours	1 593
20 000	6	UBCSAT IROTS	3×10^7	52.3 min	1 631
20 000	8	GSAT/K-Means/WALKSAT	2×10^8(GSAT) + 2×10^8(WALKSAT)	24.1 min	3 916
20 000	8	UBCSAT GWSAT	10^8	51.9 min	3 953
20 000	8	Our WALKSAT	10^9	1.56 hours	3 944
20 000	8	UBCSAT IROTS	3×10^7	50.8 min	4 049
20 000	10	GSAT/K-Means/WALKSAT	2×10^8(GSAT) + 2×10^8(WALKSAT)	31.2 min	6 621
20 000	10	UBCSAT GWSAT	10^8	53.1 min	6 722
20 000	10	Our WALKSAT	10^9	1.94 hours	6 693
20 000	10	UBCSAT IROTS	3×10^7	50.3 min	6 699
50 000	6	GSAT/K-Means/WALKSAT	2×10^8(GSAT) + 2×10^8(WALKSAT)	30.0 min	8 684
50 000	6	UBCSAT GWSAT	10^8	1.98 hours	8 853
50 000	6	Our WALKSAT	10^9	1.82 hours	8 789
50 000	6	UBCSAT IROTS	3×10^7	2.15 hours	8 821
50 000	8	GSAT/K-Means/WALKSAT	2×10^8(GSAT) + 2×10^8(WALKSAT)	36.0 min	15 955
50 000	8	UBCSAT GWSAT	10^8	2.04 hours	19 194
50 000	8	Our WALKSAT	10^9	2.20 hours	15 992
50 000	8	UBCSAT IROTS	3×10^7	1.96 hours	16 321
50 000	10	GSAT/K-Means/WALKSAT	2×10^8(GSAT) + 2×10^8(WALKSAT)	47.0 min	23 838
50 000	10	UBCSAT GWSAT	10^8	2.11 hours	24 206
50 000	10	Our WALKSAT	10^9	2.83 hours	24 075
50 000	10	UBCSAT IROTS	3×10^7	2.23 hours	24 384

TABLE III
PERFORMANCE OF ALGORITHMS FOR $n = 20\,000$, $50\,000$ AND $\alpha = 6$, 8 AND 10. UNSAT IS THE NUMBER OF UNSATISFIED CLAUSES IN THE ASSIGNMENT FOUND BY THE ALGORITHMS.

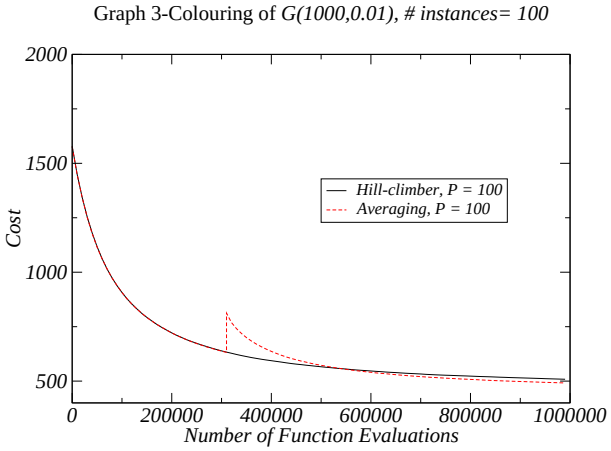

Fig. 9. Shows the mean cost (i.e. number of colour conflicts) versus the total number of function evaluations for 3-colouring 1000 random graphs. The random graphs have 1000 vertices with an edge probability of 0.01. The results show the best cost for a population of 100 individuals. In the solid curve we show results for a hill-climber alone, while the dashed curve shows a hill-climber where we have performed "averaging" after 3000 hill-climbs on each member of the population.

B. Binary Perceptron

We also consider a problem taken from machine learning known as the binary perceptron. Here we are given a set of "binary patterns", $\boldsymbol{\xi}_\mu$ together with class labels, c_μ,

$$\{(\boldsymbol{\xi}_\mu, c_\mu)|\mu = 1, 2, \ldots, P\}, \quad \boldsymbol{\xi}_\mu = (\xi_{1,\mu}, \xi_{2,\mu}, \ldots, \xi_{n,\mu})$$
$$\xi_{i,\mu} \in \{-1, 1\}, \qquad c_\mu \in \{-1, 1\}.$$

The problem is to find a vector $\boldsymbol{S} = (S_1, S_2, , \ldots, S_n)$ where $S_i \in \{-1, 1\}$ which correctly classifies as many patterns as possible. A pattern, $\boldsymbol{\xi}_\mu$, is correctly classified if

$$c_\mu \boldsymbol{S}^\mathsf{T} \boldsymbol{\xi}_\mu > 0.$$

In other words, we want to choose $\boldsymbol{S}$ so that it aligns with as many pattern-vectors in class 1 as possible, and anti-aligns with as many patterns in class 2 as possible. This problem has been very heavily analysed [24], [25], [26], [27]. It is known to be NP-Hard [28], but more significantly, in practice, it has proved to be extremely difficult to solve due to the enormous number of local minima it has [29], [11], [30]. Rather than perform hill-climbing using the number of misclassified patterns (which gives very poor performance) we instead minimise

$$\sum_{\mu=1}^{P} \left[\!\!\left[\frac{c_\mu \boldsymbol{S}^\mathsf{T} \boldsymbol{\xi}_\mu}{\sqrt{n}} > \kappa \right]\!\!\right] \left(\frac{c_\mu \boldsymbol{S}^\mathsf{T} \boldsymbol{\xi}_\mu}{\sqrt{n}} - \kappa \right)^2 \qquad (1)$$

where $[\![predicate]\!]$ is an indicator function (i.e. equal to one if *predicate* is true and zero otherwise). By setting $\kappa = 0.5$ this objective function was found to more effective than the naive choice—this is in accordance with the findings in reference [29]. In figure 10, we show the performance of a hill-climber with and without averaging. In both cases, we used a population of size 10, and run for $100\,000$ hill-climbs. We performed averaging after $50\,000$ hill-climbs. For this problem, we just took the centroid to be the solution with the minimum mean Hamming distance to all members of the population. After finding the centroid, we replicated the solution 10 times and performed a perturbation of 10% of the variables before applying another $50\,000$ hill-climbs. Averaging and perturbing gives a significant boost in performance over just hill-climbing. We also show the effect of just performing a 10% perturbation rather than averaging. In this case, the

perturbation also provides an increase in performance, but much less than that produced by averaging. We have not tried K-means clustering on this problem as the solutions found by hill-climbing are so weakly correlated that clustering (at least, for small populations) is hard to do meaningfully.

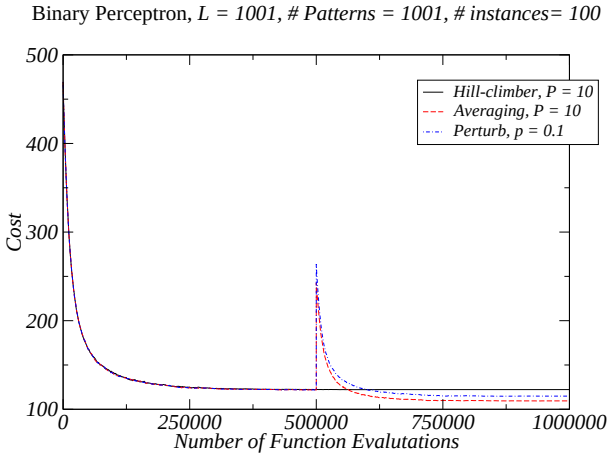

Fig. 10. Shows the mean cost (number of misclassified patterns) versus the total number of function evaluations for 1000 random instances of the binary perceptron problem with vectors of size $n = 1001$ and $P = 1001$ patterns. The three curves show the best member of a population of 10 individuals undergoing hill-climbing alone; hill-climbing with averaging performed after 4000 hill-climbs; and hill-climbing with a perturbation of 10% of the variables.

MAX-SAT, graph 3-colouring and the binary perceptron all show similar behaviours, although there are differences in the optimal time for performing averaging and in its effectiveness. We also tested the algorithm on Vertex Cover and the 3-dimensional spin-glass. Here we found a different behaviour. Averaging provided an immediate benefit (i.e. the centroid was considerably fitter than the best member of the population), however, over time the hill-climber without averaging caught up with the hill-climber with averaging. Although averaging was not detrimental, neither did it provide any long term benefit. On further investigations it appeared that the instances of the problems we examined do not suffer from the problem of a large number of local optima at a large distance from each other. Thus, although averaging appeared to reach a good part of the search space, the hill-climber could also reach this part of the search space given more time.

VI. Conclusions

The empirical evidence described in this paper provides strong support for the hypothesis that we are able to learn about the large-scale structure of the landscape of MAX-3-SAT problems. By doing so we have produced an algorithm which substantially out-performs all other algorithms we compared it with. It is not the intention of our research to tune our algorithm, and so we would expect that the performance of the K-Means algorithm may be improved upon. For example, it may be possible to fit a smooth model of the fitness landscape that filters out high-frequency fluctuations. This might allow the good regions of the solution space to be more accurately determined.

Interestingly a genetic algorithm using conventional crossover does not learn this large-scale structure nearly as well as clustering or averaging. Averaging, or using the centroid of a cluster, breaks the metaphor of natural selection. Averaging acts like blending inheritance in that it reduces the diversity in the population. Since the modern synthesis in population biology, it has been understood that an essential component of natural selection is the particulate nature of genes such that crossover exchanges genes rather than average them. As a consequence, the kind of averaging we have undertaken has been little explored, yet, as we argued at of section IV-B, this averaging allows the landscape to be explored in a very different way to conventional crossover.

We have also examined other problems, namely graph-colouring, binary perceptron, Vertex Cover and the 3-d spin-glass, although these results are very preliminary. We have not investigated the landscapes of these problems as we have for MAX-SAT, nor have we compared our algorithms against the state-of-the-art algorithms for these problems. Nevertheless, they clearly provide evidence that in these problems taking the "centroid" of a population of independent good quality solutions moves the search to a better part of the search space. For problems where moves around the search space are hampered by the presence many local optima, the ability to jump to a better part of the search space can provide a significant enhancement in performance. In MAX-SAT, we have gone further and shown that clustering solutions using K-means clustering finds even better solutions than averaging alone. We have yet to test K-means on other problem classes, however, for problems which exhibit many widely-separated global optima, the idea of clustering would appear to offer considerable potential benefits.

References

[1] J. L. Shapiro and A. Prügel-Bennett, "Genetic algorithms dynamics in two-well potentials with basins and barriers," in *Foundations of Genetic Algorithms 4*, R. K. Belew and M. D. Vose, Eds. San Francisco: Morgan Kaufmann, 1997, pp. 101–116.

[2] T. Jansen and I. Wegener, "On the analysis of evolutionary algorithms — a proof that crossover can really help," in *Proceedings of the 7th Annual European Symposium on Algorithms (ESA'99)*, J. Nešetřil, Ed. Berlin: Springer, 1999, pp. 184–193.

[3] A. Rogers and A. Prügel-Bennett, "The dynamics of a genetic algorithm on a model hard optimization problem," *Complex Systems*, vol. 11, no. 6, pp. 437–464, 2000.

[4] ——, "A solvable model of a hard optimization problem," in *Theoretical Aspects of Evolutionary Computing*, ser. Natural Computing, L. Kallel, B. Naudts, and A. Rogers, Eds. Berlin: Springer, 2001, pp. 207–221.

[5] R. A. Watson, "Analysis of recombinative algorithms on a non-separable building-block problem," in *Foundations of Genetic Algroithms (FOGA-6)*, W. N. Martin and W. M. Spears, Eds. San Francisco: Morgan Kaufmann, 2001, pp. 69–89.

[6] R. A. Watson and T. Jansen, "A building-block royal road where crossover is provably essential," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2007)*, 2007, pp. 1452–1459.

[7] A. Prügel-Bennett, "Finding critical backbone structures with genetic algorithms," in *Proceedings of the 9th annual conference on genetic and evolutionary computation (GECCO 2007)*. ACM Press, 2007, pp. 1343–1348.

[8] D. Mitchell, B. Selman, and H. Levesque, "Hard and easy distributions of SAT problems," in *Proceedings of the Tenth National Conference on Artificial Intelligence*. San Jose, CA, USA: Publ by AAAI, Menlo Park, CA, USA, 1992, pp. 459–465.

[9] J. M. Crawford and L. D. Auton, "Experimental results on the crossover point in random 3-SAT," *Artificial Intelligence*, vol. 81, no. 1-2, pp. 31–57, 1996.
[10] W. Zhang, "Phase transitions and backbones of 3-SAT and maximum 3-SAT," in *Principles and Practice of Constraint Programming - CP 2002. 7th International Conference, CP 2001. Proceedings (Lecture Notes in Computer Science Vol.2239)*. Paphos, Cyprus: Springer-Verlag, 2001, pp. 153–67.
[11] A. Prügel-Bennett, "Symmetry breaking in population-based optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 1, pp. 63–79, 2004.
[12] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky, "Determining computational complexity from characteristic 'phase transisition'," *Nature*, vol. 400, pp. 133–137, 1999.
[13] M. Mezard and R. Zecchina, "The random k-satisfiability problem: from an analytic solution to an efficient algorithm," *Physical Review E*, vol. 66, p. 056126, 2002. [Online]. Available: http://www.citebase.org/abstract?id=oai:arXiv.org:cond-mat/0207194
[14] D. Arclioptas, A. Naor, and Y. Peres, "Rigorous location of phase transitions in hard optimization problems," *Nature*, vol. 435, no. 9, pp. 759–764, 2005.
[15] A. Montanari, G. Parisi, and F. Ricci-Tersenghi, "Instability of one-step replica-symmetry-broken phase in satisfiability problems," *Journal of Physics A: Mathematical and General*, vol. 37, no. 6, pp. 2073–2091, 2004. [Online]. Available: http://stacks.iop.org/0305-4470/37/2073
[16] D. Battaglia, M. Kolár, and R. Zecchina, "Minimizing energy below the glass thresholds," *Phys. Rev. E*, vol. 70, no. 3, p. 036107, Sep 2004.
[17] M. Mézard, T. Mora, and R. Zecchina, "Clustering of solutions in the random satisfiability problem," *Phys. Rev. Letts.*, vol. 94, p. 197205, 2005.
[18] B. Selman, H. Levesque, and D. Mitchell, "A new method for solving hard satisfiability problems," in *Proceedings of the Tenth National Conference on Artificial Intelligence*. San Jose, CA, USA: Publ by AAAI, Menlo Park, CA, USA, 1992, pp. 440–446.
[19] B. Selman, H. A. Kautz, and B. Cohen, "Noise strategies for improving local search," ser. Proceeding of the Twelfth National Conference on Artificial Intelligence, vol. vol.1. Seattle, WA, USA: MIT Press, 1994, pp. 337–43.
[20] M. Qasem and A. Prügel-Bennett, "Complexity of max-sat using stochastic algorithms," in *GECCO*, 2008.
[21] J. A. Hartigan and M. A. Wong, "A k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–8, 1979, copyright 1979, IEE 1385369 0035-9254 clustering algorithm.
[22] G. Syswerda, "Simulated crossover in genetic algorithms," in *Foundations of Genetic Algorithms 2*, L. D. Whitley, Ed. San Mateo: Morgan Kaufmann, 1993.
[23] W. Zhang, "Configuration landscape analysis and backbone guided local search. part 1: satisfiability and maximum satisfiability," *Artificial Intelligence*, vol. 158, no. 1, pp. 1–26, 2004.
[24] E. Gardner and B. Derrida, "Optimal storage properties of neural network models," *J. Phys. A*, vol. 21, p. 271, 1988.
[25] W. Krauth and M. Mézard, "Storage capacity of memory networks with binary couplings," *Journal de Physique*, vol. 50, pp. 3057–3066, 1989.
[26] B. Derrida, R. B. Griffiths, and A. Prügel-Bennett, "Finite-size effects and bounds for perceptron models," *Journal of Physics: A*, vol. 24, pp. 4907–4936, 1991.
[27] H. Horner, "Dynamics of learning for the binary perceptron problem," *Zeitschrift für Physik: B*, vol. 86, pp. 291–308, 1992.
[28] L. Pitt and L. G. Valient, "Computational limitations on learning from examples," *Comput. Machinary*, vol. 35, p. 965, 1988.
[29] K. Patel, "Computational complexity, learning rules and storage capacities: Monte Carlo study for the binary perceptron," *Zeitschrift für Physik: B*, vol. 91, pp. 257–266, 1993.
[30] J. Hallam and A. Prügel-Bennett, "Large barrier trees for studying search," *IEEE Transaction on Evolutionary Computation*, vol. 9, no. 4, pp. 385–397, 2005.