Improving Performance in Combinatorial Optimisation Using Averaging and Clustering

Mohamed Qasem and Adam Prügel-Bennett

School of Electronics and Computer Science
University of Southampton, SO17 1BJ, UK.

Abstract. In a recent paper an algorithm for solving MAX-SAT was proposed which worked by clustering good solutions and restarting the search from the closest feasible solutions. This was shown to be an extremely effective search strategy, substantially out-performing traditional optimisation techniques. In this paper we extend those ideas to a second classic NP-Hard problem, namely Vertex Cover. Again the algorithm appears to provide an advantage over more established search algorithms, although it shows different characteristics to MAX-SAT. We argue this is due to the different large-scale landscape structure of the two problems.

1 Introduction

One of the potential benefit of using an evolutionary algorithm (EA) is the possibility to learn about the large-scale structure of the fitness landscape from the whole population. However, there are few examples of EAs on real world problems where the algorithm unambiguously exploits this global knowledge of the landscape. Recently we proposed an algorithm for solving large MAX-SAT problems based on clustering good solutions which we argued does precisely this [1]. We review that work here and extend the idea to a second classic NP-Hard problem, Vertex Cover. The algorithm behaves differently on Vertex Cover to MAX-SAT, although again the algorithm provides a performance improvement. We argue that the reason for the difference in behaviour of the algorithm on the two problems reflects difference in the large-scale structure of the fitness landscape.

The algorithm we use here is a hybrid algorithm. We find many good solutions using a local neighbourhood search algorithm (a basic hill-climber, BHC). The solutions are either averaged or clustered using a K-means cluster algorithm. The solution closest to the mean solution or 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 2.1.

Our interpretation for the good performance of this algorithm is that in many combinatorial optimisation problems good quality solutions tend to surround the global maxima. Thus by averaging good solutions, we can find a position in the vicinity of a global maximum, or, at least, a very high quality solution. As

the landscapes are often rugged, the average solution (or more accurately the nearest feasible solution to the average) may not itself be very good, although by continuing the search from the average solution we are likely to find an improved solution. It may be thought that crossover performs a similar operation to averaging, however, this is not the case, as, in expectation, a child is as far from the centroid of the parents as the parents themselves (this is even true when more than two parents are used in crossover). This observation is backed up empirically, where we find crossover does not perform at all like averaging.

For many combinatorial optimisation problems the landscape is more complex because there are often global optima a substantial distance apart. We can imagine the local optima being like stars clustered into galaxies. The high quality solutions typically being close to the centre of the galaxies. In this case, if we average solutions lying in different galaxies the mean solution may not lie close to the centre of any galaxy. However, by first clustering solutions and using the centroid of each cluster we can hope that at least some centroids might be close to the centre of a galaxy. For MAX-SAT we found clustering was superior to averaging all solutions (although, averaging all solutions was found to be superior to hill-climbing, suggesting that the galaxies themselves are correlated—this interpretation is supported by direct empirical studies). In Vertex Cover clustering provided no significant improvement over averaging suggesting that the global optima for Vertex Cover may not be so widely spread over the search space as MAX-SAT.

The rest of the paper is arranged as follows. In the next section, we recap on the behaviour of our new method for MAX-SAT. In section 3 we discuss the new results for Vertex Cover. Finally, in section 4 we draw conclusions.

2 Recap on MAX-SAT

MAX-SAT is a generalisation of the well-known SAT problem. A SAT problem consists of a set of m clauses $\{C_i | i = 1, 2, \ldots, m\}$, where each clause C_i is formed by the disjunction of Boolean variables, $\boldsymbol{X} = (X_1, X_2, \ldots, X_n)$ or their negation, where $X_i \in \{true, false\}$. The SAT decision problem is whether there exists a truth assignment $\boldsymbol{X}$ which satisfies all the clauses. If the number of literals in each clause is K then the problem is called K-SAT. If $K \geq 3$, then K-SAT is NP-Complete. In MAX-SAT we attempt to find an assignment of the variables which maximises the number of satisfied clauses. MAX-SAT is an NP-Hard problem.

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 [2]. These are created by generating m clauses from n variables by randomly picking K variables for each clause. Then with a probability of 0.5 a variables is negated. Duplicate clauses are discarded.

For this class of problem there is a transition between the case where most instances are satisfiable to the case where most instances 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 with most instances being easy to solve below the phase transition while above the phase transition most instances are hard to solve [2, 3]. 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 remains hard [4, 5].

2.1 Experimental Results

We briefly present empirical results of our algorithm on large instances of MAX-3-SAT for $\alpha = m/n = 8$. More details can be found in [1]. We were unable to compare our algorithm with most other algorithms that appear in the literature since the other studied were performed on much smaller instances (typically around 100 variables). For such small instances we found that running the basic hill-climber a few times would almost always find a solution we were unable to improve on and which we believe to be the global optimum. This made our clustering approach redundant. The only work we are aware of which studied similar sized instances to those used here is by Zhang [6]. Our algorithm substantially out-performs the results given in that paper. To provide some comparators to the clustering approach we ran a number of variants of the algorithms. The main purpose of these comparators was to rule out other possible explanations of why the approach we are taking is successful.

Experimental Setup We generated random MAX-3-SAT instances using the method described in section 2. We consider 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. For each increment we generated 100 problems instances.

In all tests we carried out we started by performing 1 000 hill-climbs using BHC starting from different random starting configurations. The number of iterations started from 20 000 for 6 000 variables and ended with 50 000 iterations for 18,000 variables in increments of 5 000. The number of iterations were increased with the number of variables so that BHC would be given more opportunities to find better quality solutions. With the growth of the number of variables it becomes more difficult for a local search algorithm to reach local maxima [7], although the goal was not necessarily to 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 1. We then tested a number of different strategies to boost the performance obtained from these

Table 1. Comparison of different algorithms. 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. The K-means algorithm performs best. The results are based on the number of unsatisfied clauses.

#Vars	First BHC	Second BHC (1)	K-Means/ BHC (2)	Average/ BHC (3)	hybrid-GA	Perturb/ BHC	(2) - (1)	(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

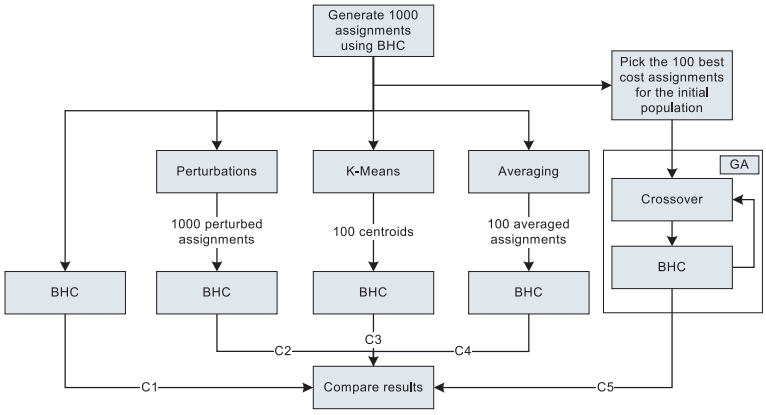

Fig. 1. Schematic diagram of the set of tests carried out and reported in table 1.

initial 1 000 points. The testing procedure we carried out is shown schematically in figure 1. As a baseline we repeated the basic hill-climber for same number of iterations on all 1 000 search points. These results are shown in the third column of table 1. This second round of hill-climbing shows that the solutions found in the first round were still some way away from being locally optimal.

***K*-Means and Averaging** We next performed clustering using the K-means clustering algorithm [8] 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). For a number of assignments $\boldsymbol{X}_k \in \{0,1\}^n$

with $k = 1, 2, \ldots, P$, the centre or average is defined to be

$$\boldsymbol{X} = \underset{\boldsymbol{X} \in \{0,1\}^n}{\operatorname{argmax}} \sum_k H(\boldsymbol{X}, \boldsymbol{X}_k)$$

where $H(\boldsymbol{X}, \boldsymbol{X}_k)$ is the Hamming distance between binary strings. Thus the average is a configuration which minimises the sum of distances to each assignment. Each of the 1 000 points is then assigned to the cluster with the nearest centre. The centres are then updated to be the centroid of the cluster. 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. Having computed the 100 centroids a second round of hill-climbing is then carried out. The results obtained after this procedure are shown in the forth column of table 1. In every case there is a considerable gain in performance compared to the baseline, although the K-means method used 100 points in contrast to the 1000 in the second round of hill-climbing (in consequence, the baseline method uses approximately 10 times as much CPU time in this second stage than the K-means clustering method). The gain in performance compared to the baseline in shown in column 8 of table 1.

We have compared clustering with 'averaging', where we randomly selected 10 points and averaged them to create a centroid. These were then rounded to give a valid assignment of the variables and a second cycle of hill-climbing carried out. This was repeated 100 times so as to give a fair comparison with the K-means clustering method. The results are shown in the fifth column of table 1. This again produced a substantial gain in performance compared with the baseline (the gain is shown in the last column of table 1), however, these gains are smaller than those obtained by K-means clustering. This seems to provide empirical support for the claim that the global maxima are clustered. It also shows that even the mean of all the good solutions provides a much better starting point than a random starting point.

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 we considered applying 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 1).

Comparison with GAs We also compared our algorithms against a hybrid genetic algorithm. This combined hill-climbing with selection and two-parent crossover. For selection we used scaled 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 were tried. The best results were obtained with single-point crossover. Column 6 of table 1 shows the best results we were able to obtain using

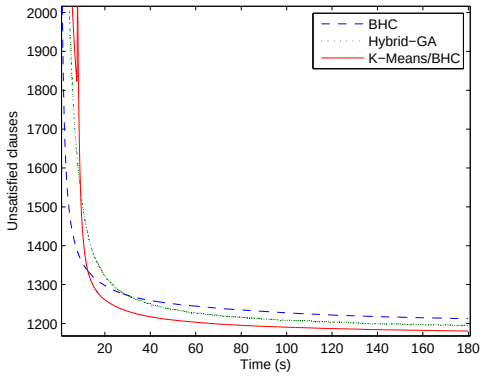

Fig. 2. Comparison of BHC, genetic algorithms and K-means clustering as a function of CPU time run on X instances of randomly generated MAX-3-SAT instances with 6000 variables at $\alpha = 8$. The large jump in fitness in the K-means algorithm after around 10 seconds marks the point where K-means clustering is carried out.

a GA. Although we do not claim that all the parameters were optimally chosen, the results obtained by the hybrid-GA are extremely disappointing compared to the other algorithms.

The behaviour of the hybrid genetic algorithm appears 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 2 we show the average performance of parallel-BHC, 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 instances with 6000 variables at $\alpha = 8$. In parallel-BHC, we run 10 BHCs in parallel and show the best of these. K-means clustering was run starting with an initial population of 100 where we performed 27 000 BHCs 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 speed 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 was tested.

2.2 Landscape of MAX-3-SAT

As described in the introduction we have attributed the performance of our algorithm to the clustering of the good quality solutions. Here we present some direct evidence in support of this picture obtained by extensive empirical observations on the fitness landscape of MAX-3-SAT for $\alpha = m/n = 8$. We studied instances up to size 100 by finding many local maxima. To achieve this we used the basic hill-climber. The algorithm was started from different, randomly-chosen, starting points. To ensure that we had found a local maximum, 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, and then 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 could be sure that we were at a local maximum.

We postulate that the best local maxima we found are the global maxima, since if there were 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 a very 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[1].

We investigated the distribution of quasi-global maxima by examining the frequencies of Hamming distances between all quasi-global maxima in an instance. In figure 2.2, we show these frequencies averaged over 300 problem instances. To find the set of quasi-global maxima we ran BHC followed by exhaustive search 5000 times on each problem instance. 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 $n/2$).

Figure 4 shows how the average Hamming distance between the local maximum and the nearest quasi-global maximum varies as a function of the difference in the cost between the local maxima and the quasi-global maxima. It is easy to understand why higher-cost solutions should be closely correlated on average with the quasi-global maxima as global-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 the solutions whose cost 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

[1] 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 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.

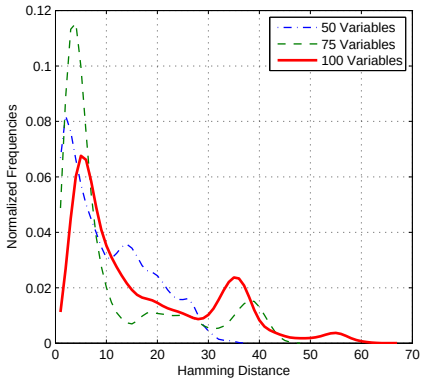

Fig. 3. Shows a histogram of the Hamming distance between quasi-global maxima for instances of size $n = 50$, 75, and 100 with $\alpha = 8$. There is a cluster of very close global maxima below a Hamming distance of 10. Also, a significant number of global maxima at a Hamming distances equivalent to 30–40% of the variables.

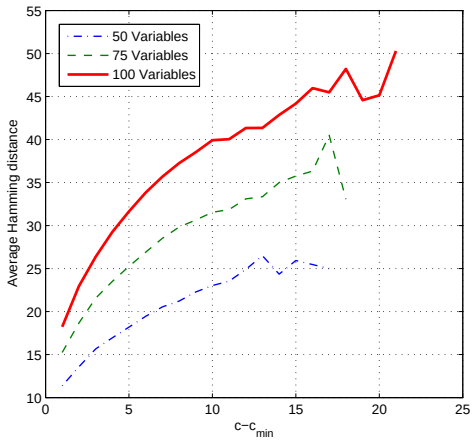

Fig. 4. The average Hamming distance between the quasi-global maxima and the local maxima. As the gap in fitness between the quasi-global maxima and local maxima decreases so does the average distance to the quasi-global minimum.

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.

Although it is always dangerous to rely on low-dimensional pictures to understand what happens in a high-dimensional space, nevertheless we offer the following caricature of our fitness landscape. We imagine the search space as

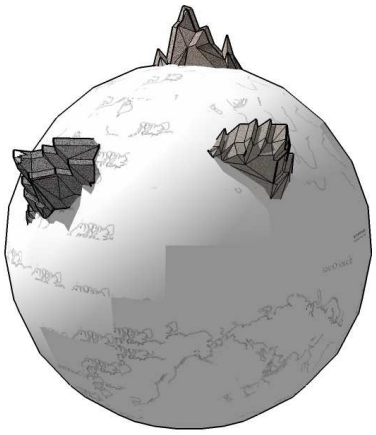

Fig. 5. Caricature of the Fitness Landscape showing the clustering of good solutions.

being points on a 'world' where the height of the points representing 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 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 cluster 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 corresponds 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 solutions. Of course, this picture fails in many ways. The search space is not continuous, but is 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 our algorithm based on clustering seems to perform very well which suggests that this simple picture might not be too misleading.

3 Vertex Covering

To further test our hypothesis that averaging or clustering is effective in other combinatorial problems we have tested the optimisation version of the Vertex Cover problem. A vertex cover for an undirected graph $G = (\mathcal{V}, \mathcal{E})$ (where $\mathcal{V}$ is the set of vertices, and $\mathcal{E}$ is the set of edges) is a subset of the vertices $\mathcal{S} \subseteq \mathcal{V}$ such that each edge has at least one end point in $\mathcal{S}$ [9]. The Vertex Cover decision problem is, given an integer R, does there exist a vertex cover. $\mathcal{S}$, such that $|\mathcal{S}| \leq R$? We consider an optimisation version of the Vertex Cover problem where we set $|\mathcal{S}| = R$ for a predetermine R and we seek a collection of elements of $\mathcal{S}$ which maximises the number of covered edges. Since the Vertex Cover problem is NP-complete, the optimisation version of the problem will be NP-hard.

To generate random problems we created 1 000 vertices, with an edge density of $p = 0.01$. That is an edge between two vertices is created with a probability of 0.01. For 1 000 vertices, the number of edges created was 4 995 on average. We chose R to be 500 vertices, and used a basic hill-climber to find the 500 most covering vertices amongst the 1 000. (We tested our algorithm on instances with many different parameter values, but found very similar results in each case).

We applied three different algorithms to solve this problem. We applied a basic hill-climber (BHC), Averaging and a Hybrid-GA. We represented a solution by two sets $\mathcal{S}$ and $\bar{\mathcal{S}} = \mathcal{V} \backslash \mathcal{S}$. Initially the elements of $\mathcal{S}$ were randomly chosen with the constraint that $|\mathcal{S}| = R$. In (BHC) we chose a random element from $\mathcal{S}$ and another random element from $\bar{\mathcal{S}}$. These were exchanged provided the number of edges covered by the vertices after the exchange was, at least, as large at the number of edges covered by the vertices before the exchange. The "average" was taken to be the set $\mathcal{S}$ with the largest intersection with all members of the population $\mathcal{S}_k$ for $k = 1, 2, \ldots, P$. As with MAX-SAT averaging is used just once, in this case after 500 basic hill-climbing iterations. We also tried performing K-mean clustering in a similar manner to that described for MAX-SAT, but this gave us no noticeable benefit compared with averaging.

In the Hybrid-GA we combined BHC with selection and crossover. After a set number of hill-climbs (500 in the case shown), $2 \times P$ members were selected using scaled Boltzmann selection with a selection strength of $\beta = 4$. The selected members were paired up and crossed to create P children. A child is produced from two parents, $\mathcal{S}_k$ and $\mathcal{S}_l$ by randomly choosing R elements from $\mathcal{S}_k \cup \mathcal{S}_l$. We experimented with different parameter values for the Hybrid-GA but were unable to find parameters where it out-performed BHC.

The results averaged over 1000 different instances of the problem are shown in Figure 6. The performance of the Hill-Climber and the Hybrid-GA are indistinguishable on the graph. Averaging produces a rapid improvement in performance. Although not very clear in the graph, the results after averaging are significantly better than either Hill-Climbing or the Hybrid-GA even at the end of the run. Interestingly, in Vertex Cover averaging produces an immediate improvement in fitness. This is in contrast to MAX-SAT where initially averaging decreases the fitness (although in the longer term it leads to a more rapid in-

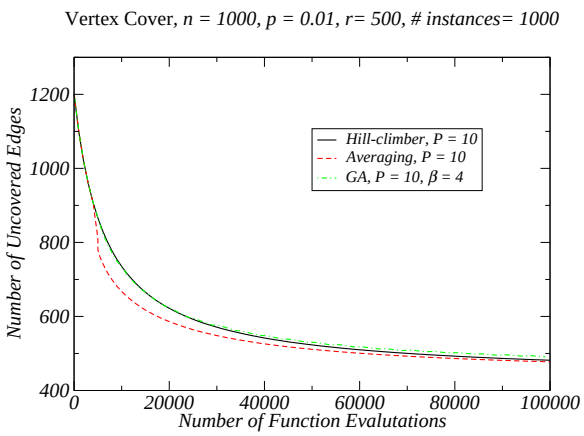

Fig. 6. Comparison of BHC, Hybrid-GA, and K-means clustering as a function of the number of evaluations. We see the the same jump in fitness we previously saw in the MAX-3-SAT problem.

crease in fitness), see Figure 2. This suggests that the landscape for Vertex Cover is much less rugged than MAX-SAT, at least, for the instances we chose.

An explanation of why clustering provides a performance advantage for Max-Sat, but not for Vertex Cover is that in Max-Sat many of the global optima are weakly correlated while in Vertex Cover the global optima lie in a much more confined part of the search space. To gain support for this view we have run independent hill-climbers on both problems and measured the correlation between solutions over time. Figure 7 shows the evolution of this correlation versus the number of hill-climbing steps. Initially, the hill-climbers are randomly chosen so on average they have zero correlation with each other. Over time the hill-climbers move towards fitter solutions. In both cases the solutions become correlated, however, this correlation is much more pronounced in Vertex Cover, suggesting that good solutions lie in a much smaller region of the search space than for MaxSat.

4 Conclusions

Clustering and averaging is shown to provide an important new search operator for solving hard optimisation problems. For MAX-3-SAT it provides a substantial improvement over other techniques. In Vertex Cover the improvement is less pronounced although it is still useful especially for finding good solutions quickly. From our studies of the landscape of both problems, the main difference would appear to be that Vertex Cover is not particularly hard—or more accurately, the instances we chose are not particularly hard.

However, the success of averaging and clustering on these two classic combinatorial optimisation problems is encouraging. The success of these operators

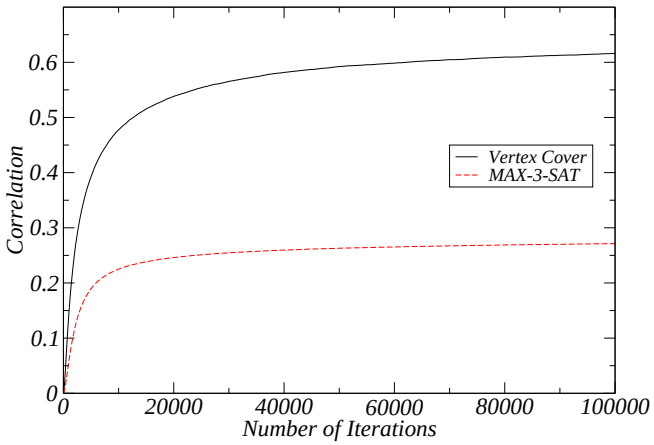

Fig. 7. Correlation between solutions found by hill-climbers versus number of iterations. The results are averaged over 100 instances of each problem. The Vertex Cover graphs are drawn from a distribution with $n = 1000$, $p = 0.01$ where $R = 500$. The MAX-3-SAT intances are for $n = 1000$ and $m = 8000$.

relies on two important features of a landscape. Firstly good solutions should be clustered around global optima (which is not difficult to imagine will often be the case). Secondly, these solutions should be isotropically clustered. If they were strongly biased in one direction then averaging may mislead the search. It is far from obvious that this isotropic clustering holds in real optimisation problems. The evidence from both MAX-3-SAT and Vertex Cover is that this is the case, which provides hope that this approach is applicable to more problems. We are currently extending this technique to new problems and particularly to problems which are known to by typically hard.

References

1. Qasem, M., Prügel-Bennett, A.: Learning the large-scale structure of the max-sat landscape using populations. (2008) Submitted to IEEE Transactions on Evolutionary Computation.
2. Mitchell, D., Selman, B., Levesque, H.: 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) 459–465
3. Crawford, J.M., Auton, L.D.: Experimental results on the crossover point in random 3-SAT. Artificial Intelligence **81**(1-2) (1996) 31–57
4. Zhang, W.: 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) 153–67
5. Prügel-Bennett, A.: Symmetry breaking in population-based optimization. IEEE Transactions on Evolutionary Computation **8**(1) (2004) 63–79

6. Zhang, W.: Configuration landscape analysis and backbone guided local search. part 1: satisfiability and maximum satisfiability. Artificial Intelligence **158**(1) (2004) 1–26
7. Qasem, M., Prügel-Bennett, A.: Complexity of max-sat using stochastic algorithms. In: GECCO. (2008)
8. Hartigan, J.A., Wong, M.A.: A k-means clustering algorithm. Applied Statistics **28**(1) (1979) 100–8 Copyright 1979, IEE 1385369 0035-9254 clustering algorithm.
9. Garey, M.R., Johnson, D.S.: Computers and intractability. A guide to the theory of NP-completeness. Freeman (1979)