Finding Critical Backbone Structures with Genetic Algorithms

Adam Prugel-Bennett
School of Electronics and Computer Science University of Southampton SO17 1BJ, UK
apb@ecs.soton.ac.uk

ABSTRACT

This paper introduces the concept of a critical backbone as a minimal set of variables or part of the solution necessary to be within the basin of attraction of the global optimum. The concept is illustrated with a new class of test problems BACKBONE in which the critical backbone structure is completely transparent. The performance of a number of standard heuristic search methods is measure for this problem. It is shown that a hybrid genetic algorithm that incorporates a descent algorithm solves this problem extremely efficiently. Although no rigorous analysis is given the problem is sufficiently transparent that this result is easy to understand. The paper concludes with a discussion of how the emergence of a critical backbone may be the salient feature in a phase transition from typically easy to typically hard problems.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algoruthms and Problem Complexity.

General Terms

Algorithms, Theory

Keywords

Combinatorial Optimisation, Backbone, Genetic Algorithm, Crossover

1. INTRODUCTION

A frequently observed phenomena of constrained optimisation problems is the existence of a 'backbone' structure. That is, some core part of the solution which is essential to get right to find high quality solutions. The concept of a backbone is a metaphor drawn from the percolation literature in statistical mechanics and first applied to combinatorial optimisation problems in [1]. In that paper, it was define as the set of variables that remain fixed in all global optimal solutions. The definition was modified in the GRAPH K-COLOURING problem to be the set of pairs of nodes each of which has the same colour in all optimal solutions. Although these definitions are easy to understand and relatively simple to calculate they miss the idea of *a small critical subset of variables necessary to be in the basin of attraction of the global solution*—a definition that we believe captures the true nature of the structure of the landscape underlying many hard optimisation problems. To make this distinction we define the *critical backbone* to be this subset of variables. The critical backbone may be a small subset of the traditional backbone. Although the concept of a critical backbone is less tangible than the formal definition, nevertheless, we believe it is more informative about the nature of constrained optimisation problems and how they are solved by heuristic algorithms. To make our notion of a critical backbone clearer we define a 'toy' problem class, BACKBONE, which has a set of variables that can be viewed as forming a critical backbone. In instances of BACKBONE it is easy to find local optima much superior to average solutions, but to find very high quality solutions requires having a good backbone structure. The purpose of introducing the class BACKBONE is to illustrate in a very simple model how a small set of variables can be critical to finding the optimal solution. Furthermore we will see that this problem can be tackled effectively using a hybrid genetic algorithm.

Since the backbone structure of a problem was given a precise definition it may appear counter productive to try to introduce a new notation of backbone which is much harder to define precisely. However, we argue that the term has taken on a folk meaning which is far richer and more pertinent than the formal definition. This is not just a semantic argument, but one of significant practical importance. The concept of a critical backbone structure that we are advocating has clear implications for heuristic search algorithms: *A successful search algorithm is one that can effectively search the space of critical backbones.* This is certainly true for BACKBONE where we show empirically that a hybrid genetic algorithm is considerably more efficient than many other standard search heuristics such as multiple descents or simulated annealing precisely because it can efficiently search the space of critical backbones. The structure of BACKBONE is sufficiently transparent to understand this result without detailed theoretical analysis. We will argue that the same mechanism of efficiently searching the space of backbones explains the success of some genetic algorithms on real world problems. An implication of this thesis is that a key to designing successful genetic algorithms (GAs) is to find a

GECCO'07, July 7–11, 2007, London, England, United Kingdom.

recombination operator that efficiently explores the space of backbones.

After submitting the first draft of this paper, the author has become aware of closely related work on *backdoors* [2, 3]. These are defined as sets of variables which when set allow a problem to be solved in polynomial time using some well defined algorithm (whether a set of variables is considered a backdoor will depend on the algorithm used). This idea has been examined almost exclusively in the area of SAT problems which, being a decision problem, has a slightly different character to many optimisation problems. These studies find that for small SAT problems it is necessary only to set a very few variables variables in order for a SAT solver to find a satisfying solution in polynomial time. Interestingly they find very little correlation between backdoors and backbones, although this result appears slightly at variance with other studies on backbones.

The rest of this paper is organised as follows. In the next section, we formally define BACKBONE and describe its properties. In section 3, we discuss how different search heuristics explore the space of critical backbones and present some representative empirical results. Finally in section 4, we return to our discussion about the definition of a backbone and argue why the notion of critical backbone is important in many classical constrained optimisation problems.

2. BACKBONE

We propose a new class of test problems, BACKBONE, which defines a cost function for an n-element binary string, $\boldsymbol{X} \in \{-1, 1\}^n$ (throughout we assume all binary strings consist of elements ± 1). The optimisation problem is to find a binary string to minimise the cost. We can formally define the problem class BACKBONE as a 6-tuple

$$\langle n_b, n_l, n_x, \boldsymbol{W}, \boldsymbol{f}(\cdot), \pi(\cdot) \rangle$$

where

- n_b is a positive integer defining the size of the critical backbone
- n_l is a non-negative integer defining the number of variables in each of n_b block
- n_x is a non-negative integer defining the number of variables in the cross-linking component of the cost function
- $\boldsymbol{W}$ is a weight vector consisting of $n = n_b \times (n_l+1) + n_x$ real numbers
- $\boldsymbol{f} : \{-1, 1\}^{n_b} \to \{-1, 1\}^{n_x}$ is a function that takes a binary string of length n_b (the projection of $\boldsymbol{X}$ onto the set of backbone variables[1]) and returns a binary string of length n_x
- $\pi : \mathbb{N} \to \mathbb{N}$ is a permutation of the numbers from 1 to $n = n_b \times (n_l + 1) + n_x$ defining the position of the variables in the string.

[1]We define $\hat{\mathrm{P}}_{BB} : \{-1, 1\}^n \to \{-1, 1\}^{n_b}$ to be the projection from the full binary string to a binary string containing only the backbone variables

The total number of variables is $n = (n_l + 1)n_b + n_x$. The cost is given by

$$\begin{aligned}
c(\boldsymbol{X}) &= c^{bb}(\boldsymbol{X}) + c^{bl}(\boldsymbol{X}) + c^{x}(\boldsymbol{X}) \\
c^{bb}(\boldsymbol{X}) &= \sum_{i=1}^{n_b} W_i X_{\pi(i)} \\
c^{bl}(\boldsymbol{X}) &= \sum_{i=1}^{n_b} c_i^{bl}(\boldsymbol{X}) \\
c_i^{bl}(\boldsymbol{X}) &= \sum_{j=1}^{n_l} W_{\pi(in_b+j)} X_{\pi(i)} X_{\pi(in_b+j)} \\
c^{x}(\boldsymbol{X}) &= \sum_{j=1}^{n_x} W_{\pi((n_l+1)n_b+j)} f_j(\hat{\mathrm{P}}_{BB}(\boldsymbol{X})) X_{\pi((n_l+1)n_b+j)}.
\end{aligned}$$

Note that permutation $\pi(i)$ defines where the i^{th} variable sits on the string. The first n_b variables (before making the permutation) are the critical backbone variables. The next $n_b \times n_l$ variables form n_b blocks of length n_l and the last n_x variable form a 'cross-linking block'.

The cost function is made up of three pieces. The first piece, c^{bb}, depends only on backbone variables and is minimised when these variables take the opposite sign to their associated weights, W_i. The second piece c^{bl} consist of a set of blocks; one block for each backbone variables. If the backbone variable remains fixed then the block variables can be chosen to minimise the cost by setting them to $-\operatorname{sign}(W_j X_{\pi(i)})$ where $X_{\pi(i)}$ is one of the critical backbone variable. Changing one of the backbone variables will completely change the cost of the associated block. Without the last term the cost function could be completely split into n_b independent pieces. The last term provides a cross-linking of the variables. For each backbone $\hat{\mathrm{P}}_{BB}(\boldsymbol{X})$ we define an n_x-dimensional binary 'mask' $\boldsymbol{f}(\hat{\mathrm{P}}_{BB}(\boldsymbol{X}))$. The cross-linking variables are minimised with respect to this binary mask. That is they minimise the cost when

$$X_{\pi((n_l+1)n_b+j)} = -\operatorname{sign}\left(W_{\pi((n_l+1)n_b+j)} f_j(\hat{\mathrm{P}}_{BB}(\boldsymbol{X}))\right).$$

In this paper, the weights, W_i, are assumed to be iid Gaussian variables (with mean zero and unit variance). However, the structure of the problem remains unchanged for many different choices of weight vector. For example, if we choose all the weights to be equal to one, we observe very similar behaviour. The binary masks, $\boldsymbol{f}(\hat{\mathrm{P}}_{BB}(\boldsymbol{X}))$, are taken to be random binary strings (implemented as a look-up table). We consider two types of permutations, π; a random permutation, π^r, and a high ordered permutation, π^o, where the block variables and their associated backbone variables are tightly linked. More specifically the variables are arranged in a set of n_b sections where each section consists of a single critical backbone variable its associated block variables and some number of cross-linking block variables. For example, if we had as many cross-linking variable as backbone variable ($n_x = n_b$) each section would be built up as shown below.

	X_i	$X_{n_b i+1} \cdots X_{n_b i+n_l}$	$X_{n_b(n_l+1)+i}$

The ordered version of BACKBONE (with no cross-linking between backbone variables, i.e. $n_x = 0$) is similar to a model recently proposed by Richard Watson which consisted

of similar blocks of variables. The motivation and interpretation of these two models is however very different. In the model of Watson, linkage is seen to be key to solving the problem. In BACKBONE, the key to solving the model is to discover the critical backbone. As we will see later, there seems to be no advantage in exploiting linkage even when it exists.

For a fixed set of backbone variables the block variables and cross linked variables are easily optimised by any simple descent algorithm—in this sense the critical backbone can been seen as playing the same role as backdoors in SAT problems [2, 3]. The difficulty in solving the problem is to learn the backbone variables since changing a backbone variable, $X_{\pi(i)}$, will not only change the backbone cost, c^{bb}, but will also change the cost of the associated block, c_i^{bl}, and the cross-linking cost, c^x. Provided the weights W_i in the backbone cost, c^{bb}, are not too large, there will be a local optimum associated with every possible assignment of the backbone variables. Thus the backbone variables in this problem precisely encode the information about which basin of attraction the solution is in. A feature of this problem, which is commonly observed in many classic optimisation problem, is that given a good solution (which may have taken many hours to discover) it is possible to perform a large perturbation (i.e. a macro-mutation) which would significantly increase the cost. However, if one then applies a descent algorithm one rapidly returns to the cost one started from. It is clear that this will happen in BACKBONE provided the backbone variables are not changed by the macro-mutation.

A feature of BACKBONE that has been observed, for example, in MAX-SAT is that lower cost local minima are typically closer in Hamming distance to the global optima than higher cost local minima [4]. This phenomena has important consequences for heuristic search as it provides an opportunity to learn the backbone structure. We discuss this in the next section.

3. HEURISTICS SEARCH ON BACKBONE

In this section, we discuss how a few well known heuristic search algorithms explore the space of BACKBONE. To provide a quantitative comparison between different algorithms we consider an instance of BACKBONE, $\langle 8, 6, 8, \boldsymbol{W}^g, \boldsymbol{f}^r, \pi \rangle$, where $\boldsymbol{W}^g$ is a vector of Gaussian variables, $\boldsymbol{f}^r$ are randomly drawn masks, and we consider two permutations; π^r where the variables are in random position and π^o where the variables are arranged in sections as described above. We measure run times in terms of the number of fitness evaluations, however, we distinguish between performing a fitness evaluation on a random string and making a fitness evaluation after one local move (i.e. changing a single variable). This reflects the situation in many real world search problems where computing the change in cost caused by a local move is typically quicker than calculating the cost on the whole string (this is also the case for BACKBONE).

We first consider a simple descent algorithm where a random position on the string is chosen and the corresponding variable has its sign changed if this reduces the cost. This algorithm will reach one of the local minima defined by the critical backbone variables. For instances when the weights of the backbone variable are of the same magnitude as the other weights there will be 2^{n_b} local minima. The basin of attraction of the local minima are the set of configurations where the cost of flipping a backbone variable is less than the penalty caused by disrupting the associated block cost and cross-linking cost. Once in a basin of attraction, the local minimum will be reached after every variable has been tried. Lower cost (fitter) local optima that are closer to the global optimum will typically have slightly larger basins of attraction than high cost local optima. However, because there are a large number (2^{n_b}) of local minima and the bias in the size of basins of attraction is not great, the likelihood of finding the global optima via descent is small.

We can increase this probability of finding the global optima by using multiple descents where we re-initialise the string to a random string after a given number of descents moves. The efficiency of this 'restart' algorithm is determined by the number of descent moves performed before reinitialising. For the instance we are using the optimal number of descent moves is around 250 (the precise optimum balance between descent moves and restart moves will depend on the complexity of recomputing the cost for a random string compared with the complexity of computing the change in cost following a local move). The average number of local descent moves need to find the optimal was $(4.3 \pm 0.1) \times 10^4$ with an average of 170 ± 5 restarts (requiring the full evaluation of the cost of a random string). These numbers are computed over 1 000 runs. Not all descents will have reached a local minima before the string is reinitialised. If we increase the number of descent moves to 1 000 then the average number of restarts need to find the optimum solution is 95 ± 3 which indicates that the probability of finishing in the global optimum is somewhat greater than that for the average local optima (in this instance there are 256 local optima so if they had identical basins of attraction the expected number of restart moves to reach one particular optimum would be 256). Although increasing the number of descent moves per restarts reduces the number of restarts it increases the number of descent moves overall.

Another commonly used heuristic search strategy for problems with many local minima is simulated annealing [5]. In this method a local move is made if it reduces the cost. If a local move increases the cost then a move is made with a probability $\exp(-\beta \Delta c)$ where Δc is the increase in cost and β is a control parameter known as the inverse temperature. The efficiency of simulated annealing depends critical on how β is chosen (it is usually changed at each time step; starting from a low value and slowly increased over time). For this problem, we used a self-adaptive mechanism to ensure that the probability of accepting any move is 0.2. More specifically we started from $\beta = 0.1$ and computed a running average for the acceptance rate, $r(t)$, according to $r(t+1) = 0.95\, r(t) + 0.5\, \chi(t)$ where $\chi(t)$ is one if the move was accepted and zero otherwise. After each move the value of β was increased by a factor of 1.1 if $r(t)$ was above 0.2 and decreased by a factor of 0.9 otherwise. The parameters were chosen after a small amount of testing. The average number of local descent moves needed to find the global optimum were $(8.8 \pm 0.3) \times 10^4$; thus taking around twice the time of the restart strategy. It is easy to see why simulated annealing finds this problem hard. The difference in the costs between local minima (which is determined by the weights in c^b) is small compared to the barrier between local minima which depends on the set of weights in the corresponding block variables and weights for the cross-linking variables. This ratio between the difference in cost of local minima and the barriers, is known to determine the hardness

of a problem for simulated annealing. The reason for this is that a low temperature is needed to distinguish between local minima, but this makes escaping from local minima difficult.

BACKBONE is also a difficult problem for a traditional generational genetic algorithm to solve. To maintain a reasonably low cost (fit) population the selection pressure must be very high. However, such a high selection pressure removes all diversity in the population. This lack of diversity prevents the population from searching the space of critical backbones effectively. We could find no set of parameters where a GA could solve the problem in a reasonable number of fitness evaluations.

The situation changes dramatically when we considered a hybrid GA where we performed descent combined with selection and crossover. We used a population of size 30. At each generation we performed 200 attempted descent moves followed by selection. For selection we used scaled Boltzmann selection with a selection strength $\beta = 0.1$ [6], combined with stochastic universal sampling [7]. After selection, we performed uniform crossover on all members of the population (i.e. we created a child string by randomly choosing a variable from either of two parents independently at each position on the string). These parameters were chosen after a limited amount of experimentation. The average number of local moves used to solve the problem was $(2.45 \pm 0.04) \times 10^4$ with a total of 122 ± 2 full fitness evaluations. This corresponds to solving the problem in an average of just over three generations. The hybrid algorithm takes just over half the time of the restart algorithm. One disadvantage of this approach is that it is possible to get fixation in one of the backbone variables that prevents the algorithm from finding the global optimum. The probability of fixation is reduced by increasing the population size. With the parameters used we were able to find the global optimum in all 1 000 runs for which we tested the algorithm.

When the string is randomly ordered (i.e. we use the permutation π^r) applying single-point crossover increases the number of local moves needed to solve the problem to $(3.56 \pm 0.06) \times 10^4$ with 177 ± 3 full fitness evaluations required. We would expect single-point crossover to be less effective at exploring the space of critical backbones as it typically mixes solutions much slower than uniform crossover [8]. Using the ordered string (π^o) the disruption caused by single-point crossover is much smaller. This allowed us to reduce the number of descent moves per generation to 100. However, we still paid the penalty of poor mixing so that the even with an ordered string we require an average of $(2.48 \pm 0.05) \times 10^4$ local moves and 248 ± 2 full fitness evaluations to solve the problem. The difference in the number of local moves used by single-point crossover and uniform crossover is not statistically different. In single-point crossover we used half the number of descents moves per generation, but it required twice as many generation to find the solution. In conclusion, it appeared that there was little or no advantage in exploiting any linkage structure in the string.

We can easily understand the success of the hybrid algorithm. Each round of descent leaves every member of the population close to or in a local minima. Selection, then selects for the lower cost local minima which are typically closer in Hamming distance to the global minimum than the high cost local minima. Crossover then searches the critical backbone space. Thus there are two very distinct levels of search. The descent method for finding local minima and crossover for exploring backbone space. The structure of the search space is easy at both levels. That is, finding a local minima through descent is relatively fast and finding the critical backbone through crossover is relatively fast. For a small problem the advantage of the hybrid search is not so significant, but as the size of the backbone increases, the advantage of the hybrid algorithm becomes more evident. Thus for $\langle 16, 6, 16, \boldsymbol{W}^g, \boldsymbol{f}^r, \pi^r \rangle$ the average number of local moves for the restart algorithm using 500 descents between restarts is $(1.8 \pm 0.6) \times 10^7$ with $(3.6 \pm 1.1) \times 10^4$ restarts (calculated over 10 runs). A hybrid GA with a population size of 50 and performing 400 attempted descent moves each generation solved this larger problem in $(3.0 \pm 0.1) \times 10^5$ local moves and 745 ± 30 full function evaluations (calculated over 1 000 runs). That is almost two orders of magnitude faster than the descent algorithm. We could not find a set of parameters that allowed simulated annealing to solve this large instance problem in a sensible time scale.

Although these comparisons are somewhat *ad hoc*, with only a limited amount of parameter optimisation for each algorithm, they are sufficiently clear cut to demonstrate the general trends. In particular, they show a very clear advantage of the hybrid genetic algorithm. It may appear that the problem has been contrived to favour hybrid algorithm. (Although, it is clear this is a contrived problem, it was originally designed to demonstrate the concept of a critical backbone.) The moot point is whether there exists any evidence for a similar structure to occur in real optimisation problems. As we have already pointed out, in MAX-SAT it has been shown that lower cost local minima tend to be closer in Hamming distance to a global minimum than higher cost local minima [4]—this is clearly a necessary requirement for a hybrid algorithm to be successful. However, the situation for MAX-SAT is more complicated than BACKBONE in that there can be many global minima so the structure of the space of critical backbones may be more complicated. Interestingly, there is some support for the success of hybrid genetic algorithms for another classic optimisation problem, namely GRAPH K-COLOURING.

GRAPH K-COLOURING is a very well studied problem. The problem is to colour a graph with K colours in such a way as to minimise the number of edges whose nodes have the same colour. Interestingly, the best known algorithm for this task is a hybrid genetic algorithm designed by Galinier and Hao [9]. They recognised that GRAPH K-COLOURING is not really a colouring problem, but rather a partitioning problem where the graph is partitioned into K sub-graphs such that there are as few edges in each sub-graph as possible. They designed a crossover operator which attempted to preserve sub-graphs. It combined two parents by considering each parent in turn for K steps. At each step it took the largest colour class (i.e. set of nodes with one colour) from the parent it was considering and copied it into the child. The actual colour of the colour class was ignored. After choosing the colour class, these nodes were removed from both parents. This is repeated until the child had K colour classes. Not all nodes of the child will be coloured at this stage. The remaining nodes are coloured at random. The algorithm was combined with a sophisticated Tabu search which acted as a very powerful local search method. Although the initial cost of crossover was very high (due to the random colouring of the final nodes), Tabu search very

quickly repaired the children. This algorithm substantially out-performed previous methods on a large range of test problems. This is a significant achievement as this is a competitive problem which has received considerable attention from across the optimisation community. Crossover was essential to the success of the hybrid algorithm. The Tabu search on its own, although it had a very respectable performance, was substantially out-performed by the hybrid algorithm. Nor was the the Tabu search strictly necessary. The same quality of results were obtained using a standard descent algorithm in combination with Galinier and Hao's crossover (although, it took considerably longer to obtain these results) [10].

An interpretation of Galinier and Hao's result is that the local search method (Tabu search) is finding good critical backbone structures while the crossover operator is combining solutions to efficiently search the space of good backbones. Just as in BACKBONE, the cost after crossover is typically relatively high, however, the local search method very quickly reduces the cost.

4. CONCLUSIONS

The traditional idea of a backbone arose in studying phase transitions in some of the classic constraint optimisation problems. As the number of constraints increase the optimisation problems are frequently observed to undergo a transition from simple to hard [1, 11] (for decision problems the problem often becomes easy again as the number of constraints increases further). At the phase transition, the backbone defined as the number of variables common to all global optimum solutions was observed to increase dramatically. The idea being that the problem became hard because it was necessary to fix a large proportion of the variables. We argue that this may be a misinterpretation. The backbone becoming large at the phase transition may only be indicating that the global optimum is becoming unique (or there may be a small number of neighbouring configuration making up the global optimum). Notice, that in BACKBONE all variables would be considered to be backbone variables as there is a unique global optimum. We observe also that the fact that every variable has to take on a particular value is not an indication that the problem is hard. The ONES-MAX problem has a backbone equal to the number of variables but is easy to solve.

A feature which seems to be much more indicative of the easy-hard phase transition is the "shattering" of the landscape into many valleys [12]. Search becomes difficult because if one starts in the wrong valley a neighbourhood search algorithm will only find a local optimum. The idea of the critical backbone is that it is a minimum set of variables (or more generally a part of the solution) needed to ensure that you are in the valley containing the global optima. The number of variable in the critical backbone may be a small fraction of the total number of variables or may even be sublinear (e.g. $\Theta(\sqrt{n})$). The size of the critical backbone may be a crude measure of the hardness of the problem. This is not true of the traditional backbone, which becomes extensive (i.e. $\Theta(n)$) at the easy-hard phase-transition. Nevertheless, for a problem like MAX-SAT, the hardness of the problem (as measured by algorithm run time) continues to increase as the number of constraints (clauses in the case of MAX-SAT) increases without a significant change in the size of the traditional backbone. We would speculate that the size of the critical backbone increases with the problem difficulty indicating that more information is required to be in the basin of attraction of a global optima.

There is however, an important difference between the traditional backbone and the critical backbone. The traditional backbone is defined independently of the neighbourhood structure. The critical backbone depends on the basins of attraction of the global minimum which depends on the neighbourhood structure imposed on the problem. Thus the critical backbone is a much more global concept, which makes it harder to compute and is the reason why it has a much more vague definition than the traditional backbone. Despite this it is our contention that it is a much more pertinent concept for understanding heuristic search. Understanding, what constitutes the critical backbone may be vital to designing efficient algorithms and in particular recombination in genetic algorithms. A good crossover operator should preserve the structure of the critical backbone. For BACKBONE this is rather easy to do, however for GRAPH K-COLOURING preserving the critical backbone is highly non-trivial. A naive crossover operator such as uniform crossover will not preserve the critical backbone for GRAPH K-COLOURING and gives a very poor algorithm. Thus a better understand of critical backbones may result in a significant improvement in the application of genetic algorithms.

5. REFERENCES

[1] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, and L. Troyansky. Determining computational complexity from characteristic 'phase transisition'. *Nature*, 400:133–137, 1999.

[2] R. Williams, C. Gomes, and B. Selman. Backdoors to typical case complexity.

[3] P. Kilby, J. Slaney, and T. Walsh S. Thiebaux. Backbones and backdoors in satisfiability. In 2005, editor, *Proceedings of the 20th National conference on artificial intelligence and the 17th innovative appllications of artificial intelligence conference*, pages 1368–1373, Menlo park, CA. AAAI/MIT Press.

[4] W. Zhang, A. Rangan, and M. Looks. Backbone guided local search for maximum satisfiability. In *Proc. of the 18th Intern. Joint Conference on Artifical Intelligence*, pages 1179–84, 2003.

[5] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[6] A. Prügel-Bennett and J. L. Shapiro. The dynamics of a genetic algorithm for simple random Ising systems. *Physica D*, 104:75–114, 1997.

[7] J. E. Baker. Reducing bias and inefficiency in the selection algorithm. In *Proceedings of the Second International Conference on Genetic Algorithms*. Lawrence Erlbaum Associates (Hillsdale), 1987.

[8] A. Prügel-Bennett. The mixing rate of different crossover operators. In W. N. Martin and W. M. Spears, editors, *Foundations of Genetic Algorithms 6*, pages 261–274. Morgan Kaufmann, San Francisco, 2001.

[9] P. Galinier and J. K. Hao. Hybrid evolutionary algorithms for graph coloring. *Journal of Combinatorial Optimization*, 3(4):379–397, 1999.

[10] C. A. Glass and A. Prügel-Bennett. Genetic algorithms for graph colouring: Exploration of Galinier and Hao's algorithm. *Journal of Combinatorial Optimization*, 7:229–236, 2003.

[11] O. C. Martin, R. Monasson, and R. Zecchina. Statistical mechanics methods and phase transitions in optimization problems. *Theoretical Computer Science*, 265(1-2):3–67, 2001.

[12] A. Prügel-Bennett. Symmetry breaking in population-based optimization. *IEEE Transactions on Evolutionary Computation*, 8(1):63–79, 2004.