Available online at www.sciencedirect.com

SCIENCE DIRECT®

Artificial Intelligence 158 (2004) 1–26

Artificial Intelligence

www.elsevier.com/locate/artint

Configuration landscape analysis and backbone guided local search. Part I: Satisfiability and maximum satisfiability

Weixiong Zhang

Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130, USA

Received 15 September 2003; accepted 4 April 2004

Abstract

Boolean satisfiability (SAT) and maximum satisfiability (Max-SAT) are difficult combinatorial problems that have many important real-world applications. In this paper, we first investigate the configuration landscapes of local minima reached by the WalkSAT local search algorithm, one of the most effective algorithms for SAT. A configuration landscape of a set of local minima is their distribution in terms of quality and structural differences relative to an optimal or a reference solution. Our experimental results show that local minima from WalkSAT form large clusters, and their configuration landscapes constitute big valleys, in that high quality local minima typically share large partial structures with optimal solutions. Inspired by this insight into WalkSAT and the previous research on phase transitions and backbones of combinatorial problems, we propose and develop a novel method that exploits the configuration landscapes of such local minima. The new method, termed as backbone-guided search, can be embedded in a local search algorithm, such as WalkSAT, to improve its performance. Our experimental results show that backbone-guided local search is effective on overconstrained random Max-SAT instances. Moreover, on large problem instances from a SAT library (SATLIB), the backbone guided WalkSAT algorithm finds satisfiable solutions more often than WalkSAT on SAT problem instances, and obtains better solutions than WalkSAT on Max-SAT problem instances, improving solution quality by 20% on average.

Keywords: Search; Local search; Backbone; Backbone guided search; Configuration landscape; Boolean satisfiability; Maximum Boolean satisfiability

E-mail address: zhang@cse.wustl.edu (W. Zhang).

0004-3702/$ – see front matter
doi:10.1016/j.artint.2004.04.001

1. Introduction and overview

Boolean satisfiability or *SAT* is an archetypical decision problem [6,11]. A SAT instance is typically represented by a formula consisting of a set of Boolean variables and a conjunction of a set of disjunctive clauses of literals, which are variables or their negations. A clause is satisfied if one of its literals is set to true, and a formula is satisfied if no clause is violated. A formula defines constraints on the possible combinations of variable assignments in order to satisfy the formula. A SAT problem is to decide if a variable assignment exists that satisfies all the clauses.

There are overconstrained problems in the real world in which not all constraints are satisfiable and the objective is to satisfy the maximal number of constraints. Such a problem is called *maximum Boolean satisfiability* or *Max-SAT*. Max-SAT is the optimization counterpart of SAT, and is more general and difficult to solve than SAT. The solution to a Max-SAT can be used to answer the question of its decision counterpart.

A SAT or Max-SAT with k literals per clause can be short handed as k-SAT or Max-k-SAT, respectively. It is known that k-SAT with k being at least three is NP-complete and Max-k-SAT with k at least two is NP-hard [11], meaning that there is no known polynomial algorithm for the problems. Many real-world problems can be formulated and solved as SAT or Max-SAT, including scheduling, multi-agent cooperation and coordination, pattern recognition, and inference in Bayesian networks [1,5,8,10].

Recent years have witnessed significant progresses on SAT in two directions. The first is the understanding of problem properties, such as phase transitions [4,12,16,23] and backbones in combinatorial problems [24,31]. It has been shown that there exist sharp transitions in the satisfiability of random SAT problem instances as the ratio of the number of clauses to the number of variables (clause/variable or C/V ratio) increases beyond a critical value [4,23], a phenomenon similar to phase transitions in disordered systems [12, 13,16]. It has also been observed that the fraction of backbone variables, the ones that have fixed values among all optimal solutions to Max-SAT, increases abruptly as the clause/variable ratio goes across a critical value, yet another phenomenon similar to phase transitions [31].

The second research direction is focused on developing efficient SAT solvers, especially local search algorithms [22,27,28]. The best known local search algorithms include WalkSAT [27] and its variations [22]. These algorithms are able to significantly outperform systematic search algorithms on most random problem instances and some problem classes of real applications, solving larger satisfiable problem instances in less time, albeit the former may fail to reach a solution even if such a solution exists. The great success of WalkSAT has subsequently led to the paradigm of formulating and solving difficult problems from other problem domains, such as planning, as satisfiability problems [19]. The SAT-based approach is now among the most competitive methods for planning.

A problem of fundamental interest and practical importance is how to utilize problem structural information, such as that of phase transitions and backbones, in a search algorithm to cope with the high computational cost of difficult problems, as well as to improve the performance of the algorithm. The published work on this topic is limited. Pemberton and Zhang [26] developed a transformation method that exploits phase transitions of tree search problems, and Dubois and Dequen [9] proposed a method to incorporate esti-

mated backbone information from SAT in a systematic search algorithm for random 3-SAT. Telelis and Stamatopoulos [29] proposed a heuristic sampling method for generating initial assignments for a local search for Max-SAT using a concept similar to backbone. Despite the success of these previous works, much research is needed to exploit problem structural information in order to demonstrate the viability of incorporating such information in search algorithms.

One of the challenges in utilizing structural information of a problem, such as phase transitions or backbones, in a search algorithm, is to make the algorithm not only work on random problem instances, but also perform well on *individual problem instances*, especially those from real-world applications. To our knowledge, no result on individual real problem instances has been reported in the previous work using structural information. A phase-transition property is a characteristic of a collection of problem instances drawn from a common distribution. Therefore, phase-transition information may only help when solving a set of related problem instances from which structural information can be extracted. For individual problem instances, however, information of an ensemble may be irrelevant. Therefore, new structural information must be acquired and new mechanism of applying such information must be devised.

In this paper we are mainly interested in Max-SAT, thanks to its generality and broad applicability in practice, and local search algorithms, the WalkSAT algorithm in particular, which was originally developed for SAT problem instances. We first investigate the configuration landscapes of local minima reached by WalkSAT. Briefly, we define the configuration landscape of a set of local minima as their distribution with respect to their cost and structure differences in reference to all optimal solutions or a particular reference local minimum. Similar global structures have been analyzed on other optimization problems, such as the Traveling Salesman problem (TSP) and graph bisection problem, and local search algorithms, such as 2-opt for the TSP [3,20]. Our experimental results on SAT and Max-SAT show that local minima from WalkSAT reside close to one another, forming clusters in configuration landscapes. The results also indicate that WalkSAT is effective for Max-SAT as well, finding many high quality local minima that are very close to optimal solutions in terms of cost and structure differences. Although we carry out this analysis in part for the purpose of developing a new method, the work itself and the results are of interest of their own. Note that the WalkSAT algorithm, as well as other local search algorithms, is a procedure for optimization. However, previous researches on WalkSAT have focused on its application to SAT, a decision problem. To our knowledge, this paper appears to provide the first systematic performance analysis on WalkSAT for Max-SAT, an optimization problem that includes SAT as a special case. Previous research also showed that WalkSAT is an effective method for solving overconstrained Steiner Tree problems [18].

The second main contribution of this paper is an innovative and general heuristic method that exploits backbone information to improve the performance of a local search algorithm. The new method is driven by the results on the configuration space analysis of local minima uncovered in the first part of this paper. It is also inspired by the previous research on phase transitions [4,12,16,23] and backbones of combinatorial problems [24,31]. This method is built upon the following working hypothesis: On a problem whose optimal and near optimal solutions form a cluster, if a local search algorithm can reach close vicinities of such solutions, the algorithm is effective in finding some information of the

solution structures, backbone in particular. This implies that the local minima reached by the algorithm must share many parts of the solution structures with the optimal solutions. If we extract some of the structure information from the local minima, we can then use it to adjust the local search in such a way that it attempts to fix the parts of the current state which are not compatible with optimal solutions, so as to guide the search toward the regions of the search space containing high quality solutions. Using Max-SAT and the WalkSAT local search algorithm, we demonstrate how this new method can improve WalkSAT's effectiveness. We empirically show that the new method is effective on random problem instances and *real problem instances* from a SAT library (SATLIB [15]).

We proceed as follows in the rest of the paper. In Section 2, we first describe SAT and Max-SAT, as well as the WalkSAT local search algorithm for both SAT and Max-SAT. In this section, we also describe an extension to WalkSAT that allows a dynamic parameter tuning [14] at runtime, which we utilize to free WalkSAT from reliance on a manually set noise parameter. We investigate the configuration landscapes of local minima from WalkSAT in Section 3. We then develop the backbone guided local search algorithm in Section 4. We discuss the main idea of this method, consider how it can be incorporated in WalkSAT to make biased moves, and describe ways of capturing backbone information. We then present in Section 5 experimental results of backbone guided local search on random problem instances and instances from SATLIB [15]. We conclude in Section 6 with discussions on future work.

An early version of the paper appeared in [33]. The software developed and used in this research is freely available at http://www.cse.wustl.edu/∼zhang/projects/bgwalksat/index.html.

2. WalkSAT local search

We provide in this section some background information of the WalkSAT local search algorithm [22,27]. We also study an existing refinement to WalkSAT for SAT [14], which eliminates WalkSAT's dependence on a manually set noise parameter, and demonstrate its efficacy for Max-SAT.

2.1. The WalkSAT local search algorithm

Even though the WalkSAT local search algorithm [27] can be applied to both SAT and Max-SAT, the existing study of WalkSAT and its variations has mainly concentrated on *satisfiable* SAT instances. As Max-SAT, which includes *satisfiable* as well as *unsatisfiable* instances, is our main focus in this research, we are interested in the effectiveness of WalkSAT in finding optimal solutions to both satisfiable and unsatisfiable instances.

WalkSAT is a randomized algorithm. The algorithm and its variations all follow the same overall procedure that starts with an initial random variable assignment and makes moves by flipping one variable at a time from True to False or vice versa, until it finds a satisfying assignment or reaches a predefined maximal number of flips. Each such attempt is called a *try* or *restart*. The procedure repeats until a maximal number of tries has been attempted.

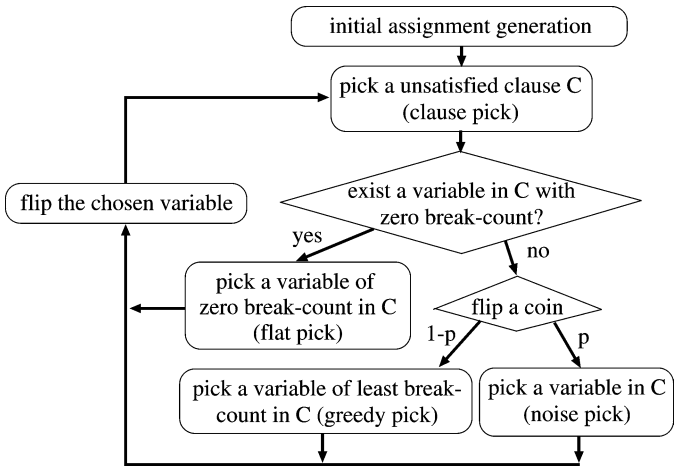

Fig. 1. Main operations in a try of WalkSAT.

To select which variable to flip in each step, the effect of flipping a variable is assessed. Flipping a variable may make some unsatisfied clauses satisfied, and some satisfied clauses unsatisfied. The numbers of clauses that will be made unsatisfied by flipping a variable is called the *break-count* of the variable at the current assignment. WalkSAT attempts to flip a variable with zero break-count, trying to make the next assignment no worse than the current one. To find a variable with zero break-count, WalkSAT first selects an unsatisfied clause C, *uniformly randomly*, from all unsatisfied clauses. This is called **clause pick**. If C has a variable of zero break-count, WalkSAT then picks such a variable, *uniformly randomly*, from the ones that qualify (called **flat pick**). If no zero break-count variable exists in C, WalkSAT then makes a random choice. With probability p it chooses, *uniformly randomly*, a variable from all the variables involved in C (called **noise pick**); or with probability $1 - p$ it selects a variable with the least break-count, breaking a tie *arbitrarily* if multiple choices exist (called **greedy pick**). The overall procedure for one try of the algorithm is shown in Fig. 1. The algorithm takes three parameters to run: the number of tries, the maximal number of flips in each try, and a probability for noise pick, which is commonly referred to as the *noise ratio* of the algorithm.

2.2. WalkSAT with dynamic noise strategy

One limitation of the WalkSAT family of algorithms is their dependence on a manually set noise ratio. To be effective, the noise ratio needs to be tuned for each individual problem, especially for those that do not share common features. So far, two methods have been proposed to resolve this issue for SAT. Auto-WalkSAT [25] uses a probing phase to estimate the optimal parameter for the noise ratio. The estimated noise ratio is then adopted throughout the search phase of the algorithm. Similar to the original WalkSAT, Auto-WalkSAT uses a static noise ratio.

Deviating from the static strategy, WalkSAT with dynamic noise [14] adopts the strategy of automatically adjusting noise ratio as the search progresses. In other words, the dynamic

strategy uses different noise ratios at different stages of the search. This strategy seems to be more reasonable than the static strategy. It is relatively easier to make great progress at an early stage of a local search than at a later stage, therefore the noise ratio should be adjusted accordingly, depending on where the current search is in the overall search space.

The idea of dynamic noise strategy is simple: start a local search with the noise ratio equal to zero, and examine the number of violations in the current state every θm flips, where m is the number of clauses of a given problem, and θ a constant. If the number of violations has not decreased since the last time we checked (θm flips ago), the search is assumed to have stagnated, and the noise ratio is increased to $wp + (1 - wp)\phi$, where wp is the current noise ratio and ϕ is another constant. Otherwise, the noise ratio is decreased to $wp(1 - 2\phi)$. The discrepancy between the formulas for increasing and decreasing the noise ratio is based on some empirical observations of how WalkSAT behaves when the noise ratio is too high, compared with how it behaves when the parameter is too low [14]. The dynamic strategy was designed and tested with WalkSAT's cutoff parameter set to infinity; i.e., no random restarts. This is the setting we use for WalkSAT with dynamic noise for all of our experiments for SAT and Max-SAT. For convenience, we refer to this strategy as Dyna-WalkSAT in the remaining of the paper.

Note that Dyna-WalkSAT uses two parameters, θ and ϕ. The difference of using these two new parameters and using the noise ratio in the original algorithm is that these two parameters do not have to be tuned for every single problem instance; the performance of Dyna-WalkSAT with the same values for θ and ϕ is relatively consistent across different problem instances. Following [14], we have set $\theta = 1/6$ and $\phi = 1/5$ in our experiments for SAT and Max-SAT.

Dyna-WalkSAT was originally designed for and tested on SAT. We complete the study in [14] by showing that Dyna-WalkSAT is effective on Max-SAT as well. In our experiments, we used problem instances of 2000 variables and C/V ratios of 4.3, 6.0 and 8.0 to capture problem instances from different constrainedness regions. We generated 1000 problem instances at each of these C/V ratios. The problem instances were random in that a clause was generated by uniformly picking three literals, without replacement, and by discarding duplicate clauses. For WalkSAT, the noise ratios were set from 0.0 to 0.9, with an increment of 0.1, the number of tries per problem instance at 100, and the number of flips per try at 10 000. We also ran Dyna-WalkSAT on each of these problem instances. To make a fair comparison, we let Dyna-WalkSAT execute one million flips total. Dyna-WalkSAT also used the same parameters as used by WalkSAT, except the noise ratio. To reiterate, following [14] we set $\theta = 1/6$ and $\phi = 1/5$ in Dyna-WalkSAT, which have been found to be effective over a wide range of SAT and Max-SAT instances.

The experimental results are shown in Fig. 2. The horizontal axes are noise ratios for WalkSAT and the vertical axes record the average solution quality. The error bars in the figures measure the 95% confidence intervals of the results. For all three C/V ratios tested, the performance of Dyna-WalkSAT is very close to the performance of WalkSAT with the optimal noise ratio, indicating that the dynamic noise strategy is effective for 3-SAT and Max-3-SAT.

Due to its simplicity and reasonable performance, in the rest of the paper we will use Dyna-WalkSAT with $\theta = 1/6$ and $\phi = 1/5$ as default parameters to replace WalkSAT in our experimental analysis.

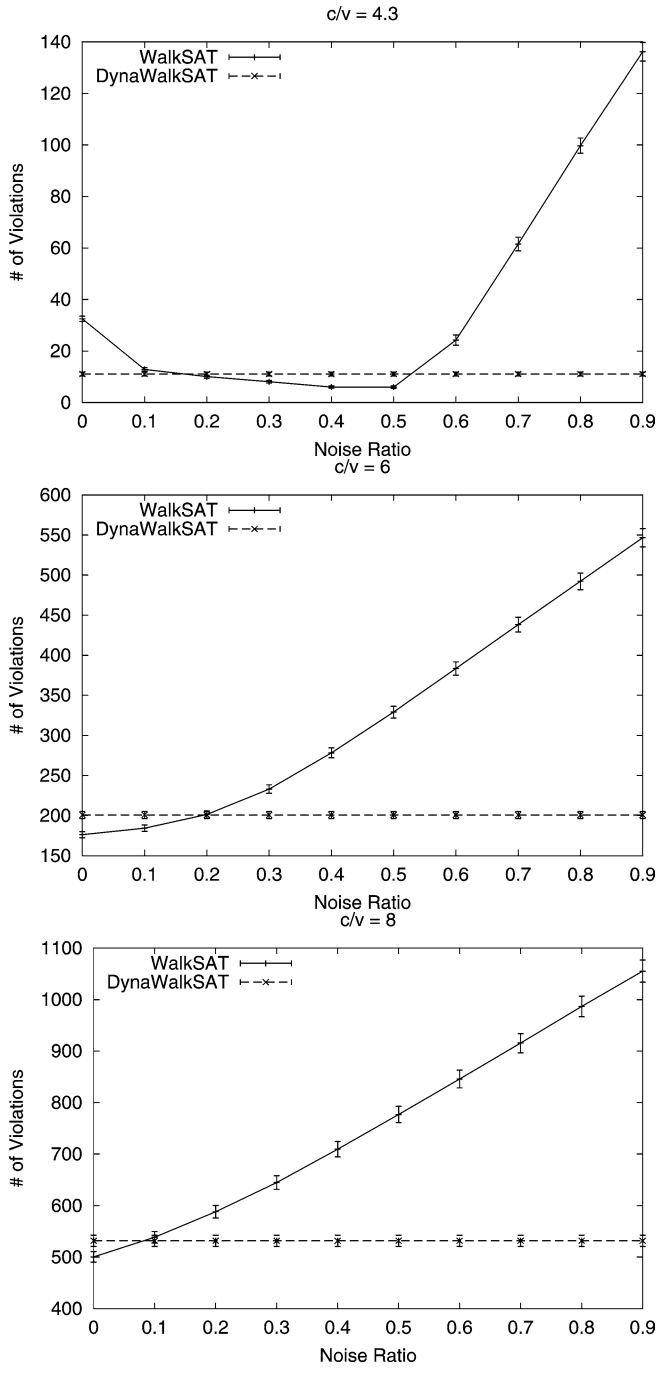

Fig. 2. Experimental validation of Dyna-WalkSAT on random Max-3-SAT, for 2000-variable problem instances.

3. Configuration landscapes

Given two variable assignments to a given SAT or Max-SAT problem instance, we can measure their differences in two ways. The first is the difference of their costs or the numbers of violated clauses. This difference can be normalized (divided) by the total number of clauses, giving the difference of violations per clause. The second quantity measures structural difference in the form of the Hamming distance between the two assignments. Since a solution to a SAT problem is simply a string of 0 and 1, Hamming distance here is simply the conventional Hamming distance for binary strings. The Hamming distance can also be normalized by the total number of variables, resulting in the normalized Hamming distance per variable. We adopt normalized cost difference and normalized Hamming distance to make the results from problems of different sizes directly comparable.

With the relative solution quality and structure difference of two assignments specified, we define the configuration landscape of a set of assignments or solutions as the distribution of the solutions in terms of their qualities and structure differences relative to a reference solution, which can be an optimal solution or the best solution in a given set. The set of solutions can be all the optimal solutions, all suboptimal solutions up to a fixed bound, as well as local minima from a local search.

We can use landscape configuration to capture the effectiveness of the WalkSAT and Dyna-WalkSAT algorithms on 3-SAT and MAX-3-SAT. We carried out two sets of experiments. In the first set of experiments, we aimed to directly measure the effectiveness of WalkSAT in terms of finding optimal and near optimal solutions. To this end, we used all optimal solutions to measure the quality of a set of local minima from WalkSAT. Since finding all optimal solutions is computationally expensive, we restricted ourselves to relatively small random problem instances with 100 variables and C/V ratios of 2.0, 4.3, 6.0 and 8.0 to capture problems in different constrainedness regions. We generated 1000 problems for each C/V ratio. The problems were randomly generated by uniformly picking three literals without replacement for a clause, with duplicate clauses discarded.

To find all optimal assignments to a Max-SAT problem, we extended the well known Davis–Putnam–Logemann–Loveland (DPLL) algorithm for SAT [7] to Max-SAT. We ran our extended DPLL algorithm for Max-SAT [30] and WalkSAT on the same set of 100-variable Max-3-SAT problem instances. For WalkSAT, we set the number of tries per problem at 100, the number of flips per try at 10 000, and the noise ratio at 0.5. We then examined the configuration landscapes of the local minima reached by WalkSAT against the optimal solutions in terms of the cost difference between a local minimum and an optimal solution as well as the Hamming distance of the local minimum to its nearest optimal solution. Note that the Hamming distance of a local minimum in fact measures the minimal number of flips required to turn the local minimum into an optimal solution.

The configuration landscapes of local minima from WalkSAT are summarized in Fig. 3. Since WalkSAT is very efficient on underconstrained SAT instances, finding satisfiable solutions of all problem instances when the C/V ratio is 2.0, we do not include the results for C/V ratio of 2.0 here. The X–Y planes in the figures show the correlation between the normalized Hamming distance and the normalized cost difference. Each point on the X–Y plane represents a set of possible local minima with the same cost difference and Hamming distance that may be visited by WalkSAT. The origins of the figures correspond to

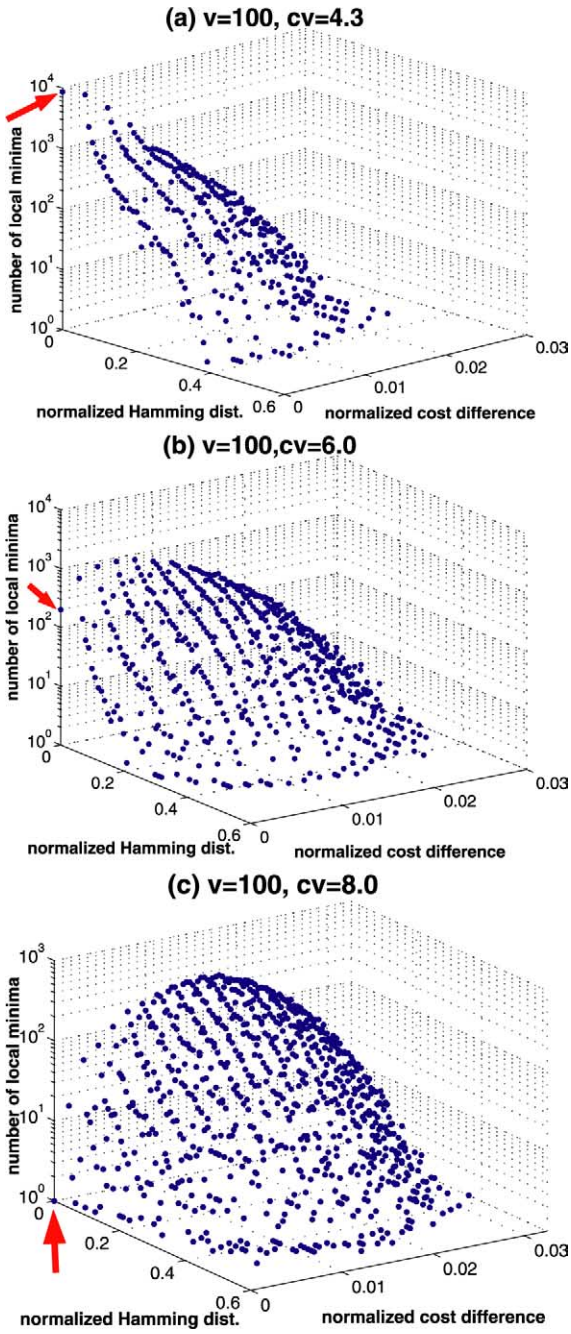

Fig. 3. Configuration landscapes of local minima from WalkSAT on 100 variable random 3-SAT and Max-3-SAT, relative to optimal solutions.

global optima. The vertical Z axes measure the total numbers of local minima reached by WalkSAT.

As shown in Fig. 3(a), WalkSAT performs well on underconstrained and critically constrained problems, in that it can find global minima very often. This is shown by the point on the Z axis indicated by the arrow in the figure. Therefore, WalkSAT is effective in finding optimal solutions on underconstrained and critically constrained problems. However, the number of local minima that are also global optima decreases from 66 677 to 8616 as the C/V ratio increases from 2.0 to 4.3, indicating that the effectiveness of WalkSAT decreases. This number decreases further from 201 to 0 on overconstrained problems with C/V ratios of 6.0 and 8.0, respectively (Figs. 3(b) and (c)). This result indicates that WalkSAT becomes less effective in finding optimal solutions as problem constrainedness increases. Fig. 4 shows the contours of the configuration landscapes in Figs. 3(b) and (c) on the X–Y planes, showing a nearly linear correlation between the cost difference and the Hamming distance, i.e., a local minimum that has a high cost tends to have a large Hamming distance to an optimal assignment.

In the second set of experiments, we examined the configuration landscapes of local minima from Dyna-WalkSAT. We used 2000-variable random 3-SAT and Max-3-SAT with C/V ratios of 4.3, 6.0 and 8.0. As before, we generated 1000 problem instances for each C/V ratio. Because the problems were too large to be solved optimally, we built a configuration landscape of a set of local minima with respect to the best local minimum among them. The results are shown in Fig. 5, which are qualitatively similar to the configuration landscapes in Fig. 3.

An interesting result from these experiments is that the configuration landscapes of local minima reached by WalkSAT and Dyna-WalkSAT exhibit *bell surfaces* on overconstrained problems with large C/V ratios. More importantly, the summit of such a bell surface shifts away from optimal solutions, the $(0, 0)$ point on the X–Y plane, as the C/V ratio increases. This observation is examplified by the contours in Fig. 4. Nevertheless, despite the increased difficult of Max-SAT as the C/V ratio grows, WalkSAT and Dyna-WalkSAT are still fairly effective in that they are able to reach local minima that are close to global optima. For 100-variable instances with C/V ratio of 8.0 (Figs. 3(c) and 4(b)), the majority of local minima reached by WalkSAT, the ones located at the peak point of the bell surface of the figure, have a normalized cost difference to optimal solutions of 0.014 for 100 variable problems, which is equivalent to about eleven more constraints violated than an optimal solution on such overly constrained problem instances. Since WalkSAT is typically executed with multiple trials, the best local minimum it can landed on will be much better than such most likely local minima.

Another interesting and important observation of the results in Fig. 4 is that there is a near linear correlation between the cost difference between a local minimum and its nearest optimal solution and their Hamming distance, the two quality measurements we adopt. This is evident that the contours in Fig. 4 have ellipse shapes. An implication of this observation is that a local minimum with a small cost is more likely to share a larger common solution structure with an optimal solution. More importantly, majority local minima have most parts of their variable assignments consistent with their nearest global optima. For instance, the majority local minima on 100-variable problems with C/V ratio of 8.0 have normalized Hamming distances around 0.13. This means that out of 100 variables, 87 of them are

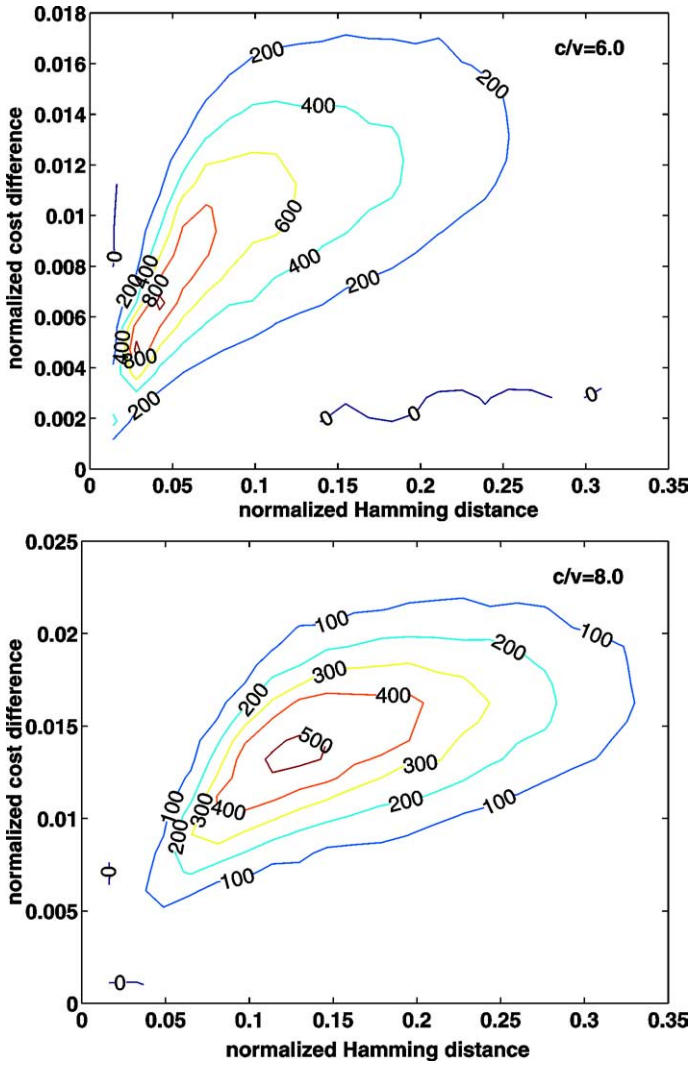

Fig. 4. Contours of the configuration landscapes of local minima from Walksat on 100 variable Max-SAT with C/V ratios of 6.0 and 8.0.

correctly set. This implies that these local minima must share large portions of the variable assignments with the optimal solutions. We will exploit this phenomenon in our new search algorithm in the next section.

In concluding this section, we need to point out that the results of this section extended the previous studies on global structures of optimization cost surfaces [2,3,20]. It has been shown in these studies (and other works cited in [2,3,20]) that there exists a correlation between the cost differences and distances among local minima of such optimization problems as the symmetric Traveling Salesman problem and graph bisection problems. Boese

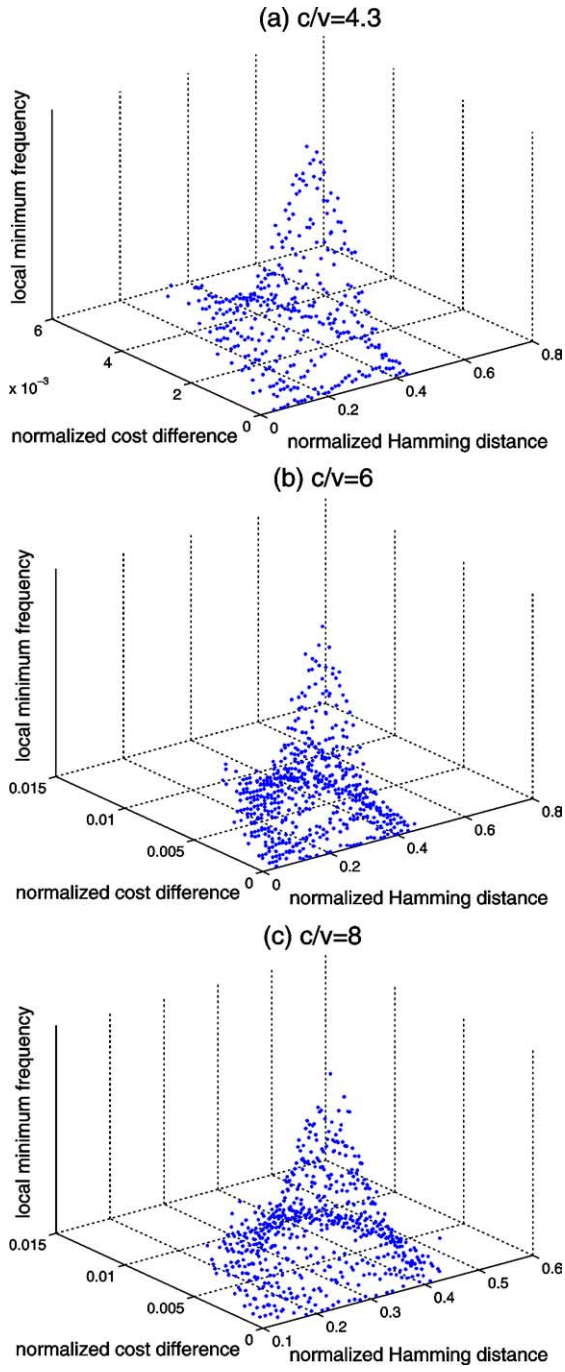

Fig. 5. Local minima from WalkSAT on 2000-variable Max-3-SAT with C/V ratios of 4.3, 6 and 8.

et al. [2,3] particularly showed a near linear relation between the quality of local minima and distances among them, which is similar to the contour results in Fig. 4. Our results made two noticeable extensions to the study of global structures of optimization problems. First, we considered configuration landscape defined as distribution of local minima with respect to the cost differences and distances between the local minima and optimal solutions or best local minima. This allows us to directly examine globally convex property or "big valley" structure of the cost surfaces of optimization problems [3]. Secondly, our results showed that there indeed exist big valley structures in the configuration landscapes of 3-SAT and Max-3-SAT, supporting the "big valley" conjecture on optimization problems [2].

Furthermore, we would also like to point out that the "big valley" results in [2,3] led to a new multi-start strategy that utilize a "big valley" structure of local minima. In this paper, we present a different, novel way to exploit such global structures, which is the topic of the next two sections.

4. Backbone guided local search

In this section, we discuss in detail the backbone guided local search algorithm. We first present the main idea and then discuss how it can be applied to WalkSAT, forming the backbone guided WalkSAT algorithm. We also consider two different ways to estimate backbone frequencies using local minima.

4.1. Main ideas

The backbone variables of a problem are the ones that are critically constrained; they must be set to particular values to make an optimal solution feasible. By the same token, if a pair of a variable and one of its values appears more often in the set of all optimal solutions, the variable is more constrained. If, somehow, we knew the frequency of a variable-value pair in all optimal solutions, we could construct a "smart" search algorithm by using the backbone frequency information as an oracle to guide each step of the algorithm. Take WalkSAT as an example. At each step of the algorithm, we can use the backbone frequencies to change the way in which a variable is chosen to flip, i.e., we prefer flipping a variable that is unsynchronized with its backbone frequency more than another variable under the current assignment. In other words, we should focus on fixing the critically constrained variables that are not currently set correctly.

Unfortunately, exact backbone frequencies of a problem are even more difficult to come by than actual problem solutions. To address this problem, the second key idea of backbone guided local search is to estimate backbone frequencies using local minima of a local search. We simply treat local minima as if they were optimal solutions and compute *pseudo backbone frequencies*, which are an estimation of real backbone frequencies. More precisely, we define the pseudo backbone frequency of a literal (a variable-value pair) as the frequency with which the literal appears in all local minima, which we denote as $p(l)$ where l is a literal. Note that $p(l) = 1 - p(\neg l)$, where $\neg l$ is the negation of l.

The quality of pseudo backbone frequencies depends on the effectiveness and efficiency of the local search algorithm used. As discussed in Section 3, high-quality local minima

can be obtained by efficient local search algorithms, such as WalkSAT. Even though WalkSAT may land on suboptimal solution with fairly high probabilities, particularly on overconstrained problem instances, most of the local minima from WalkSAT indeed have large portions of variables set to the correct values, so that they contain parts of optimal solutions or partial backbone. In this research, we adopt WalkSAT to collect local minima and then in return apply the backbone guided search method to WalkSAT to improve its performance.

4.2. Biased moves and selections

Pseudo backbone frequencies can be incorporated in a local search algorithm to make "biased" moves or flips. Consider a simple example of two variables, x_1 and x_2, that appear in a violated clause and have the same effect under the current assignment, i.e., flipping one of them makes the violated clause satisfied, and both variables have the same break-count or will cause the same number of satisfied clauses unsatisfied if flipped. Let B be the set of backbone variables along with their fixed values, $\mathcal{T}$ be the set of local minima from which pseudo backbone frequencies were computed, and v_1 and v_2 are the current values of x_1 and x_2. We will prefer to flip x_1 over x_2 if under the current assignment, $P\{(x_1 = v_1) \in B \mid \mathcal{T}\} < P\{(x_2 = v_2) \in B \mid \mathcal{T}\}$, which means that under the current assignment, x_1 is less likely to be part of backbone than x_2, given the set of local minima $\mathcal{T}$. Note that $P\{(x = v) \in B \mid \mathcal{T}\}$ can be considered as an estimate of the pseudo backbone frequency of literal $x = v$ under the evidence of a set of local minima $\mathcal{T}$.

How can the pseudo backbone frequencies be used to alter the way that WalkSAT chooses variables? As discussed in Section 2.1, WalkSAT makes *uniformly* random choices in selecting a variable to flip when multiple choices exist. For example, when there are more than one unsatisfied clause under the current assignment, WalkSAT arbitrarily (uniformly) chooses an unsatisfied clause. Similarly, when there are multiple variables with zero break-count, WalkSAT chooses one arbitrarily.

Based on the maximum entropy principle [17], it is optimal on average to make an unbiased choice if there is no information to distinguish one choice over another. Therefore, with no additional information on optimal assignments, the WalkSAT algorithm is optimal on average in terms of selecting a variable to flip. In backbone guided search, we apply pseudo backbone information to force WalkSAT to make random but *biased* choices. If a backbone variable and a nonbackbone variable can make a clause satisfied, the backbone variable should be chosen. In other words, we modify WalkSAT's random strategies in such a way that a backbone or overconstrained variable is preferred over a nonbackbone variable. To this end, we use pseudo backbone frequencies to help make random biased selections.

4.3. Backbone guided WalkSAT

Backbone guided WalkSAT has two phases. The first is a *estimation* phase that collects local minima by running WalkSAT, with a fixed number of tries. The local minima thus collected are compiled to compute the pseudo backbone frequencies of all literals.

The second phase carries out the actual *backbone guided* local search, which uses pseudo backbone frequencies to modify the way that WalkSAT chooses variables to flip. The second phase also runs many tries, each of which produces a local minimum, very often a new one. The newly discovered local minima can be subsequently added to the pool of all local minima found so far and be used to update the pseudo backbone frequencies.

We now consider methods for making biased moves in WalkSAT. The first random choice in WalkSAT is clause pick, where an unsatisfied clause is selected when multiple ones exist. We want to pick, with high probabilities, those variables that are either part of the backbone or highly constrained in all optimal solutions. Therefore, we should choose a clause containing the maximal number of critically constrained variables. To this end, we use the total pseudo backbone frequency of all the literals in an unsatisfied clause, normalized among all unsatisfied clauses, to measure the likelihood that the clause contains backbone and highly constrained variables. We then select an unsatisfied clause among all unsatisfied based on their likelihoods of containing backbone variables. Specifically, let $\mathcal{C}$ be the set of unsatisfied clauses, and q_C be the sum of pseudo backbone frequencies of all the literals in a clause $C \in \mathcal{C}$. We then let $p_C = q_C/Q$ be the probability to select clause C among all unsatisfied clauses in $\mathcal{C}$, where $Q = \sum_{C\in\mathcal{C}} q_C$ is a normalization factor.

WalkSAT uses three other random-pick rules to arbitrarily select a variable after an unsatisfied clause is chosen (see Section 2.1 and Fig. 1). To reiterate, the flat-pick rule chooses a variable from a set of zero break-count variables, if any; the noise-pick rule selects one from all variables involved in the chosen clause; and the greedy-pick rule takes a variable among the ones of least break-count. In essence, these rules use the same operation; picking a variable equally likely from a set of variables. Therefore, we can modify all these rules in the same way by using pseudo backbone frequencies. Let $\{x_1, x_2, \ldots, x_w\}$ be a set of w variables from which one must be chosen, $\{v_1, v_2, \ldots, v_w\}$ their current assignments, and $\{p_1, p_2, \ldots, p_w\}$ the pseudo backbone frequencies of literals $\{(x_1 = v_1), (x_2 = v_2), \ldots, (x_w = v_w)\}$. Then we choose variable x_i with probability $(1 - p_i)/\sum_{j=1}^{w}(1 - p_j)$. Here we use probability $1 - p_i$ because it is the probability of literal $(x_i = \neg v_i)$ being in the pseudo backbone, which is the value x_i is going to change to.

Furthermore, the idea of pseudo backbone frequencies can also be applied to generate an initial assignment for a local search. Specifically, a variable is assigned a particular value with a probability proportional to the pseudo backbone frequency of the variable-value pair. This was called heuristic backbone sampling in [29].

To summarize, we list the biased moves in WalkSAT below in comparison to the description of WalkSAT in Section 2.1 and Fig. 1.

- *BG-Initialization*: Generate an initial assignment based on the pseudo backbone frequencies of the variables.
- *BG-ClausePick*: Probabilistically select a clause among all unsatisfied ones based on their relative constrainedness under the current pseudo backbone frequencies (see text).
- *BG-FlatPick*: Probabilistically choose a variable among all zero-count variables in a selected clause based on their pseudo backbone frequencies.

- *BG-NoisePick*: Probabilistically choose a variable among all variables in a selected clause based on their pseudo backbone frequencies.
- *BG-GreedyPick*: Probabilistically choose a variable among all variables, which have the smallest break-count, from a selected clause based on their pseudo backbone frequencies.

4.4. Backbone guided WalkSAT with dynamic noise

To make the backbone guided WalkSAT algorithm more general and robust, we need to let it use dynamic noise strategy. The dynamic noise strategy discussed in Section 2.2 runs a long sequence of variable selections and flips with no restarts. This, unfortunately, is incompatible with backbone guided local search, which requires random restarts in order to collect local minima to construct pseudo backbone frequencies.

To overcome this problem, we have devised a "compromise", which allows a reasonable combination of using dynamic noise method and applying backbone information. Specifically, we run WalkSAT with dynamic noise strategy for a number of short runs to construct pseudo backbone frequencies, followed by several long runs of backbone guided local search. In our particular implementation of backbone guided Dyna-WalkSAT, we let it run thirty short runs for computing pseudo backbone frequencies, followed by seven long runs, each of which has ten times more flips than a short run.

4.5. Computing pseudo backbone frequencies

The performance of backbone guided local search depends greatly upon the quality of pseudo backbone information used. The more faithful the pseudo backbone information is, the more effective the new local search will be. In order to retrieve as much backbone information as possible, an unbiased sample of local minima should be used, in which local minima need to be derived from independently generated starting assignments. Therefore, random initial assignments should be preferred.

Given a set of local minima, pseudo backbone frequencies needs to be computed with care. We propose two different ways to compute pseudo backbone frequencies. The first and most straightforward method is to treat all the given local minima as if they were of equal quality, and take the frequency of a literal l that appears in local minima as its pseudo backbone frequency $p(l)$. Specifically, we have

$$p(l) = \frac{\sum_{\forall s_i \in S, l \in s_i} 1}{|S|}, \tag{1}$$

where S is the set of local minima. We call this method *averaging counting* or AC for short.

It is imperative to note that not all local minima are of equal quality. In general, a lower quality local minimum tends to contain less backbone information than a higher quality local minimum, as discussed in Section 3. Therefore, the backbone information carried in a lower quality local minimum is less reliable. This means that a literal appearing in a lower quality local minimum should contribute less to the pseudo backbone frequencies than a literal appearing in a higher quality local minimum. As a result, we introduce a discount factor to adjust the contribution of a literal based on the quality of the local minimum where

it came from. If a local minimum s_i has cost c_i, which is the number of violated clauses in the local minimum, then we can compute the pseudo backbone frequency $p(l)$ of a literal $l = (x_i = v_i)$ as follows.

$$p(l) = \frac{\sum_{\forall s_i \in S, l \in s_i} (\frac{1}{c_i})}{\sum_{\forall s_i \in S} (\frac{1}{c_i})}. \tag{2}$$

In other words, the contribution of a local minimum toward a pseudo backbone probability of a literal is reciprocally weighted by the cost of the local minimum. We thus call this method *cost reciprocal averaging counting* (CRAC).

5. Experimental evaluation

We now experimentally analyze the performance of the backbone guided WalkSAT (BG-WalkSAT) algorithm on SAT and Max-SAT. We use the dynamic noise strategy in both WalkSAT and BG-WalkSAT, i.e., we compare Dyna-WalkSAT and BG-Dyna-WalkSAT. Our benchmark problems are randomly generated problems and those from the SATLIB [15]. We use the number of flips of an algorithm as a time measure because the overhead of backbone-guided search on the actual CPU time is negligible.

5.1. Random ensembles

In this set of experiments, we generated random MAX-3-SAT instances with 2000 variables and three different C/V ratios of 4.3, 6.0, and 8.0, to sample instances from regions of differing constrainedness. We ignored C/V ratio of 2.0 since WalkSAT can easily find satisfiable assignments to most of underconstrained problems. At each of the ratios considered, we generated 1000 problem instances, and compared Dyna-WalkSAT and BG-Dyna-WalkSAT. Dyna-WalkSAT ran one long try with a maximum of one million flips. BG-Dyna-WalkSAT ran 30 short tries, each with a maximum of 10 000 flips, as in WalkSAT, to collect local minima. It then executed seven long tries, each with 100 000 flips, which was ten times longer than a short try. Thus BG-Dyna-WalkSAT executed one million flips also.

We found that on random problem instances the average counting (AC) method for computing pseudo backbone frequencies is less effective than the cost reciprocal averaging counting (CRAC) method under all different C/V ratios we tested. Therefore, we will present the results from CRAC here.

Using random problem instances, we first examined the effects of applying different biased moves to Dyna-WalkSAT. The results are included in Table 1. In the table, we list the results of average constraint violations for Dyna-WalkSAT and BG-Dyna-WalkSAT first, followed by the applications of biased moves to different random picks in WalkSAT. For instance, BG-ClausePick stands for Dyna-WalkSAT with biased clause picks. As shown, biased noise pick and biased initialization can improve Dyna-WalkSAT under all C/V ratios; biased clause pick has negative effects on performance; and biased greedy pick is only effective on highly overconstrained problems. Their combination also has mixed

Table 1
Comparison of backbone guided Dyna-WalkSAT variations over Dyna-WalkSAT on 2000 variable random Max-3-SAT. Performance is measured by the average number of constraint violations. The errors represent 95% confidence intervals

Configuration	C/V ratio		
	4.3	6.0	8.0
Dyna-WalkSAT	11.06 ± 0.718	200.87 ± 4.411	531.79 ± 10.990
BG-Dyna-Walksat	23.28 ± 1.060	190.91 ± 4.958	504.21 ± 10.643
BG-GreedyPick	20.58 ± 1.046	200.01 ± 4.499	506.68 ± 10.638
BG-NoisePick	10.47 ± 0.725	185.21 ± 4.203	518.48 ± 10.827
BG-Initialization	7.33 ± 0.627	190.29 ± 4.190	519.96 ± 10.771
BG-ClausePick	13.75 ± 0.693	202.74 ± 4.595	537.02 ± 11.156

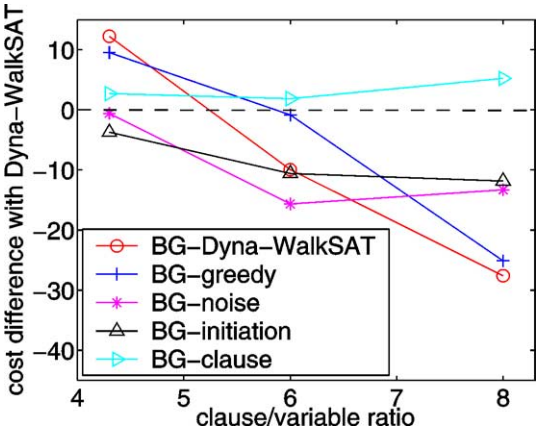

Fig. 6. Comparison on average solution quality between different strategies and the Dyna-WalkSAT algorithm on random instances.

effects: the combined biased moves improve upon Dyna-WalkSAT on overconstrained Max-3-SAT. Note that we do not include the results for biased flat pick because it has no effect on almost all problem instances we tested. The reason is that in most cases, there is only one variable with zero break-count for 3-SAT, so that biased flat pick was not used most of the time.

To examine more closely the results in Table 1, we show the difference between the new methods and the Dyna-WalkSAT algorithm. In Fig. 6, we show the average cost difference between BG-Dyna-WalkSAT and Dyna-WalkSAT as well as the cost differences between the biased moves and Dyna-WalkSAT. As shown in the figure, the performance of BG-Dyna-WalkSAT and greedy biased moves improves as the problem constrainedness increases.

The results in Table 1 also show that BG-Dyna-WalkSAT can only improve upon Dyna-WalkSAT on overconstrained instances with the C/V ratios equal to 6.0 and 8.0; while it fails to do so on critically constrained instances with the C/V ratio of 4.3. One possible

Table 2
Comparison of BG-Dyna-WalkSAT and Dyna-WalkSAT on random Max-3-SAT with C/V ratio of 8.0, averaged over 1000 instances. *Diff* is the improvement on solution quality of BG-Dyna-WalkSAT over Dyna-WalkSAT

#Var	*Dyna-WalkSAT*	*BG-Dyna-WalkSAT*	*Diff*
2000	528.32	497.61	30.71
4000	1097.60	1041.92	55.68
6000	1675.80	1597.36	78.44
8000	2248.19	2159.61	88.58
10000	2831.32	2724.14	107.18

reason is that Dyna-WalkSAT is very effective and efficient, finding optimal solutions very often on satisfiable instances.

Additional insight can be gained from an inspection of anytime behavior of Dyna-WalkSAT and BG-Dyna-WalkSAT, which is shown in Fig. 7. As discussed earlier, BG-Dyna-WalkSAT outperforms Dyna-WalkSAT only when the C/V ratios are 6.0 and 8.0. A key observation on these figures is that there was a big jump on the quality of the best local minimum found so far right after pseudo backbone information was applied to the search algorithm. This indicates that pseudo backbone information can indeed improve the search performance.

We also investigated the performance of BG-Dyna-WalkSAT with all biased moves as the problem size increases. We considered random Max-3-SAT with C/V ratio fixed at 8.0, and the number of variables from 2000 to 10 000, with an increment of 2000. We used 1000 problem instances for each different size of problems. The results comparing to Dyna-WalkSAT are shown in Table 2.

As the results showed, BG-Dyna-WalkSAT is able to improve upon Dyna-WalkSAT on random Max-3-SAT, especially on overconstrained problem instances. However, the results also show that BG-Dyna-WalkSAT is not effective comparing to Dyna-WalkSAT on random 3-SAT instances with clause/variable ratios less than 5. This is particularly true when the clause/variable ratio is less than 4.3 and problem instances are satisfiable (see the first panel in Fig. 7), partially due to the fact that Dyna-WalkSAT can find a satisfiable solution quickly. Therefore, we can conclude that we should not apply the backbone guided search to random underconstrained (satisfiable) problem instance. However, backbone guided search is effective on satisfiable problems with some structures, as we will see next.

5.2. Problem instances from SATLIB

We compared BG-Dyna-WalkSAT against Dyna-WalkSAT on problem instances from SATLIB [15]. The test problems include SAT-encoded instances from a variety of application domains, including blocks world planning, bounded model checking, all interval series problems, and hard graph coloring problems. We only chose problems with more than 350 variables, and discarded those that can be easily solved by WalkSAT and BG-WalkSAT. The chosen problem instances are difficult to solve in general, and their details can be found on the SATLIB website.

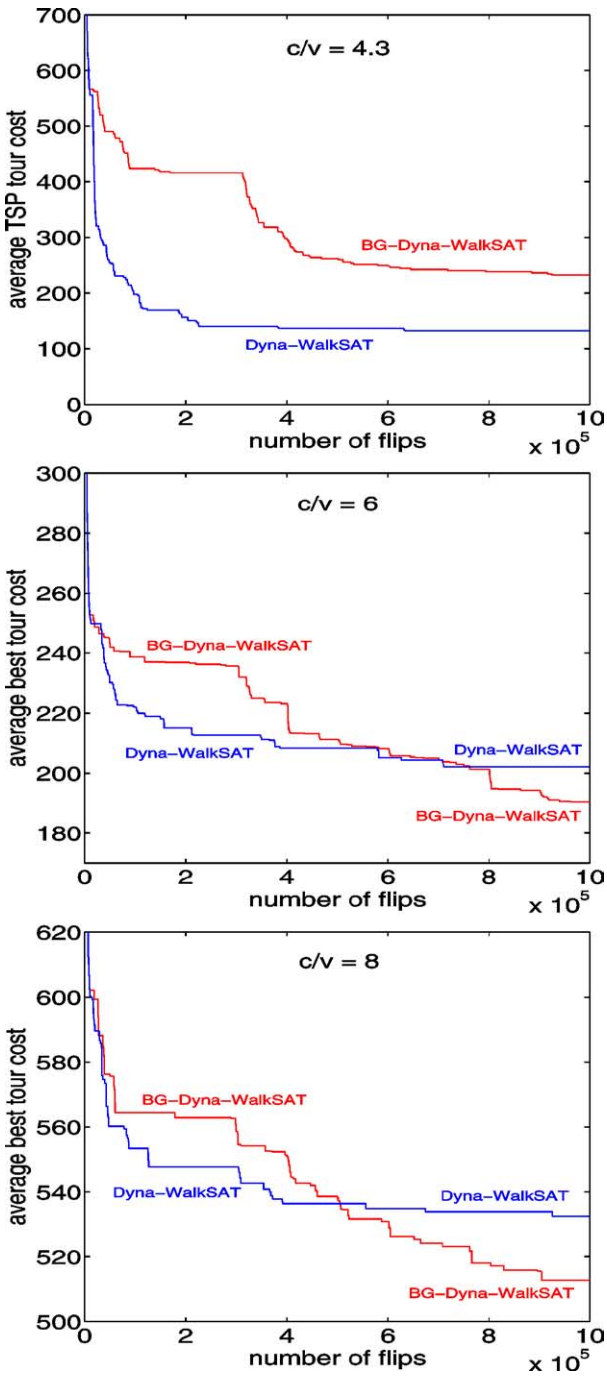

Fig. 7. Anytime performance of Dyna-WalkSAT and BG-Dyna-WalkSAT on random Max-3-SAT with 2000 variables.

Table 3
BG-Dyna-WalkSAT versus Dyna-WalkSAT on relatively easy satisfiable problems. *Dyna-WalkSAT* and *BG-Dyna-WalkSAT* are the numbers of runs resulting in satisfying solutions (out of 20) by these algorithms. The better results from the two algorithms are underlined and in bold

Problem	*#Var*	*#Clause*	*Dyna-WalkSAT*	*BG-Dyna-WalkSAT*
bw_large.c	3016	50457	1	**2**
bw_large.d	6325	131973	**1**	0
par8-1	350	1149	6	**19**
par8-2	350	1157	6	**19**
par8-3	350	1171	7	**17**
par8-4	350	1155	0	**16**
par8-5	350	1171	1	**15**
qg1-08	512	148957	8	**12**
qg2-08	512	148957	1	**4**
qg3-08	512	10469	11	**20**
qg6-09	729	21844	0	**5**
qg7-09	729	22060	4	**5**
g125.17	2125	66272	**5**	0
g250.29	7250	454622	**4**	2

We considered satisfiable and unsatisfiable problems. We ran both Dyna-WalkSAT and BG-Dyna-WalkSAT with a total of ten million flips (compared with one million for our results for random instances) as most of these problem instances are larger than the random Max-3-SAT instances considered in the previous experiments. Interestingly, on most of these problem instances the average counting (AC) method for computing pseudo backbone frequencies is slightly better than the cost reciprocal averaging counting (ARAC) method. Moreover, biased noise pick and biased clause pick provided substantial improvements to Dyna-WaslkSAT; their combination exhibited superior performance, over a wide range of real instances. In the rest of this section, we present the results of BG-Dyna-WalkSAT using these two biased moves. In our experiments, we ran each of BG-Dyna-WalkSAT and Dyna-WalkSAT twenty times, with each run executing a maximum of ten million flips.

In viewing the results, we found it useful to divide the satisfiable instances into two categories, the easier instances, which were solved at least once (Table 3), and the harder ones, which were not solved by either method, in any of their runs (Table 4). Results for unsatisfiable instances are presented in Table 5.

As the results show, BG-Dyna-WalkSAT significantly outperforms Dyna-WalkSAT in most cases. On easier satisfiable instances (Table 3), BG-Dyna-WalkSAT finds more satisfying solutions than Dyna-WalkSAT for all parity (*par*) and quasigroup (*qg*) classes, and produces results similar to those from Dyna-WalkSAT on blocksworld (*bg*) instances. On harder satisfiable instances (Table 4), BG-Dyna-WalkSAT outperforms Dyna-WalkSAT in all but two of the 34 instances, where these two results are less than half a percent worse. In contrast, the overall average gain is about 30%, and the gain is over 50% in 11 of them. On unsatisfiable instances (Table 5), BG-Dyna-WalkSAT produces impressive gains on longmult instances and on ssa6288-047, with an overall average gain of 20%. On unsatisfiable quasigroup instances (not shown), BG-Dyna-WalkSAT's performance was

Table 4
Dyna-WalkSAT vs. BG-Dyna-WalkSAT on harder satisfiable problems. Dyna-WalkSAT and BG-Dyna-WalkSAT are the average numbers of violations in the best solutions found by the algorithms for a given problem, averaged over 20 runs. Gain is the percentage improvement of BG-Dyna-WalkSAT over Dyna-WalkSAT. The better results are underlined and in bold

Problem	*#Var*	*#Clause*	*Dyna-WalkSAT*	*BG-Dyna-WalkSAT*	*gain* (%)
bmc-ibm-1	9685	55870	25.3	**4.15**	83.60
bmc-ibm-2	3628	14468	5.4	**1.2**	77.78
bmc-ibm-3	14930	72106	115.25	**19.7**	82.91
bmc-ibm-4	28161	139716	118.15	**38.9**	67.08
bmc-ibm-5	9396	41207	12.95	**1.25**	90.35
bmc-ibm-6	51654	368367	358.25	**103.6**	71.08
bmc-ibm-7	8710	39774	17.4	**6.4**	63.22
bmc-galileo-8	58074	294821	65.65	**15.5**	76.39
bmc-galileo-9	63624	326999	95.95	**17.3**	81.97
bmc-ibm-10	61088	334861	406.15	**162.45**	60.00
bmc-ibm-11	32109	150027	439.8	**358.45**	18.50
bmc-ibm-12	39598	19477	554.65	**445.25**	19.72
bmc-ibm-13	13215	6572	88.05	**2.7**	96.93
f2000	2000	8500	2.2	**2.05**	6.82
par16-1-c	317	1264	5.45	**5.35**	1.83
par16-1	1015	3310	10.45	**9.45**	9.57
par16-2-c	349	1392	6.2	**5.9**	4.84
par16-2	1015	3374	10.6	**10.4**	1.89
par16-3-c	334	1332	6	**5.65**	5.83
par16-3	1015	3344	10.45	**9.75**	6.70
par16-4-c	324	1292	6.15	**5.5**	10.57
par16-4	1015	3324	10.4	**9.55**	8.17
par16-5-c	341	1360	6.25	**6.05**	3.20
par16-5	1015	3358	10.45	**9.85**	5.74
par32-1-c	1315	5254	21.7	**20.85**	3.92
par32-1	3176	10277	30.95	**30.25**	2.26
par32-2-c	1303	5206	**21.15**	21.2	−0.24
par32-2	3176	10253	32.1	**28.35**	11.68
par32-3-c	1325	5294	22.05	**21.3**	3.40
par32-3	3176	10297	32.95	**28.55**	13.35
par32-4-c	1333	5326	**21.3**	21.4	−0.47
par32-4	3176	10313	33.65	**29.4**	12.63
par32-5-c	1339	5350	23.15	**22.05**	4.75
par32-5	3176	10325	32.9	**30.3**	7.90
Average					**29.82**

similar to that of Dyna-WalkSAT. The performance of BG-Dyna-WalkSAT is never more than 10% worse than Dyna-WalkSAT on any of the unsatisfiable instances we tested.

The most glaring failure of BG-Dyna-WalkSAT is on the satisfiable instances $g125.17$ and $g250.29$, shown in Table 3. These instances are SAT-encoded graph coloring problems, and serve to illustrate an important point. As described in Section 3, we believe that our method is effective because it exploits the "big valley" structure of the solution space. However, graph coloring problems exhibit a particular type of symmetry in their solution

Table 5
Dyna-WalkSAT vs. BG-Dyna-WalkSAT on unsatisfiable problems. The legend is the same as that in Table 4

Problem	*#Var*	*#Clause*	*Dyna-WalkSAT*	*BG-Dyna-WalkSAT*	*gain* (%)
longmult06	2848	8853	**1.5**	1.65	−10.00
longmult07	3319	10335	**2.05**	2.2	−7.32
longmult08	3810	11877	3.65	**2.65**	27.40
longmult09	4321	13479	6.75	**2.9**	57.04
longmult10	4852	15141	10.25	**5.6**	45.37
longmult11	5403	16863	15.05	**9.2**	38.87
longmult12	5974	18645	17.8	**16.2**	8.99
longmult13	6565	20487	23.25	**21.4**	7.96
longmult14	7176	22389	32.6	**24.6**	24.54
longmult15	7807	24351	41.5	**30**	27.71
ssa6288-047	10410	34238	100.25	**89.7**	10.52
Average					**20.01**

structures which is opaque to local search methods such as WalkSAT. For example, given a solution to a graph coloring problem, swapping red with green results in another solution, which is symmetrical to the original. Thus, there is no single "big valley" but several in the configuration landscape of the problem, which can bury the true backbone information and thus lead to degraded performance. Presumably, BG-Dyna-WalkSAT's performance will suffer on all instances with this type of symmetry.

6. Conclusions and discussions

In this paper, we first carried out a systematic investigation of configuration landscapes of local minima reached by the WalkSAT local search algorithm on random 3-SAT and Max-3-SAT problems. We introduced configuration landscapes to capture the distributions of local minima in terms of their cost and structural differences. Our analysis revealed that the configuration landscape of a set of local minima from WalkSAT exhibit a single bell-shaped surface, showing that the local minima form a single large cluster. Our results also showed that WalkSAT, being a procedure for optimization problems, is effective on Max-3-SAT, finding high quality local minima that have large portions of variable assignments consistent with optimal solutions.

Based on the configuration landscape analysis, we developed a novel method to exploit backbone information to improve the performance of a local search algorithm, the WalkSAT algorithm in particular. The main ideas of the method are to extract backbone information from local minima and use it directly to fix possible discrepancies between the current assignment and optimal solutions, so as to guide a local search algorithm towards the regions of search space containing high quality as well as optimal solutions. Our experimental results showed that the new method can significantly improve the performance of the WalkSAT local search algorithm on most problem instances from SATLIB, including SAT-encoded optimization problem instances from various applications. On these problem instances, the backbone guided WalkSAT algorithm has

a higher probability of reaching satisfiable solutions than the original WalkSAT algorithm, and is able to improve its solution quality on Max-SAT problem instances by 20%.

In retrospect, the most important contributions of this paper are the idea of using backbone information to improve the performance of a local search algorithm and a simple way of capturing backbone information by using local minima from a local search algorithm. These ideas are general and applicable to other combinatorial problems and search methods. For example, we have successfully applied the ideas of backbone guided local search to the Traveling Salesman Problem and the Lin–Kernighan local search algorithm [32], which is one of the oldest and most efficient algorithm for the problem [21]. By using structural information such as backbones, the new method drives a search algorithm towards the areas of the search space where most optimal or near optimal solutions are located. In comparison, most existing search techniques focus on the costs of the states in a search space. Therefore, the new algorithm is focused more on where the problems are in the current state, and tries to fix them directly.

Note that the ideas of applying backbone information to improving performance of search algorithms in [9] and this paper follow the same principle. However, in addition to the fact that we applied the ideas to a local search while [9] focused on a systematic search, there is another fundamental difference between the two. In [9], backbone is estimated for a remaining subproblem during a systematic search so as to determine which variable to instantiate next. In other words, backbone in [9] is local to a small subproblem. In our method, backbone is global in that it is collected from a set of approximate solutions.

One possible drawback of our method is that it requires a good estimation of backbone information. If this estimation deviates substantially from the real backbone information, the new method will not be effective. Nevertheless, the cost reciprocal method for estimate backbone frequencies provides a simple mechanism to ease this problem to some extent by discounting the contribution of a poor local minimum to the pseudo backbone frequencies. Furthermore, most local search methods are randomized algorithms, so better solutions may occasionally be discovered and added to the pool of local minima. Such better local minima will subsequently improve the quality of the estimation of backbone information.

The new backbone guided local search method seems to be not very effective on Max-SAT problems with little structure. The method is particularly hindered by symmetry embedded in a problem. An example of such a problem is graph coloring, where swapping two colors in a solution leads to another solution. In short, backbone guided local search seems to be effective on problems from which significant structural information can be extracted. How to extend the ideas and algorithm presented in this paper to address symmetries is an interesting future research topic.

Acknowledgements

This research was supported in part by NSF grants IIS-0196057 and EIA-0113618 under the ITR program, and in part by DARPA Cooperative Agreements F30602-00-2-0531 and F33615-01-C-1897. The views and conclusions herein are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the U.S. Government. Thanks to Henry Kautz for making the

source code of the WalkSAT algorithm available, and to Zhongsheng Guo and Xing Zhao for an implementation of the David–Putnam–Logemann–Loveland algorithms for finding one and all optimal solutions to maximum Boolean satisfiability. Special thanks to Ananda Rangan for an implementation of backbone-guided WalkSAT and some experiments at an early stage of the research, and to Moshe Looks for an implementation of dynamic noise strategy in WalkSAT and BG-WalkSAT, some experiments and assistance with preparating the paper. Thanks also to the anonymous reviewers to this paper and [33] for comments and suggestions that improved the quality of the research and presentation.

References

[1] J.C. Beck, M.S. Fox, A generic framework for constraint-directed search and scheduling, AI Magazine 19 (4) (1998) 101–130.
[2] K.D. Boese, Models for iterative global optimization, PhD Thesis, UCLA/Computer Science Department, Los Angeles, CA, 1996.
[3] K.D. Boese, A.B. Kahng, S. Muddu, New adaptive multistart techniques for combinatorial global optimizations, Oper. Res. Lett. 16 (1994).
[4] P. Cheeseman, B. Kanefsky, W.M. Taylor, Where the really hard problems are, in: Proc. IJCAI-91, Sydney, Australia, 1991, pp. 331–337.
[5] P. Codognet, F. Rossi, Notes for the ECAI2000 tutorial on solving and programming with soft constraints: Theory and practice, Available at http://www.math.unipd.it/~frossi/papers.html.
[6] S.A. Cook, The complexity of theorem-proving procedures, in: Proc. 3rd IEEE Symposium on the Foundations of Computer Science, FOCS-71, East Lansing, MI, 1971, pp. 151–158.
[7] M. Davis, G. Logemann, D. Loveland, A machine program for theorem proving, Comm. ACM 5 (1962) 394–397.
[8] R. Dechter, Constraint Processing, Morgan Kaufmann, San Francisco, CA, 2003.
[9] O. Dubois, G. Dequen, A backbone-search heuristic for efficient solving of hard 3-SAT formula, in: Proc. IJCAI-01, Seattle, WA, 2001, pp. 248–253.
[10] E.C. Freuder, R.J. Wallace, Partial constraint satisfaction, Artificial Intelligence 58 (1992) 21–70.
[11] M.R. Garey, D.S. Johnson, Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, New York, 1979.
[12] C.P. Gomes, T. Hogg, T. Walsh, W. Zhang, IJCAI-2001 tutorial: Phase transitions and structure in combinatorial problems, http://www.cs.wustl.edu/~zhang/links/ijcai-phase-transitions.html, 2001.
[13] T. Hogg, B.A. Huberman, C. Williams, Phase transitions and the search problem, Artificial Intelligence 81 (1996) 1–15.
[14] H.H. Hoos, An adaptive noise mechanism for WalkSAT, in: Proc. AAAI-02, Edmonton, AB, 2002, pp. 655–660.
[15] H.H. Hoos, T. Stuzle, SATLIB—The satisfiability library, http://www.informatik.tu-darmstadt.de/AI/SATLIB, 1999.
[16] B.A. Huberman, T. Hogg, Phase transitions in artificial intelligence systems, Artificial Intelligence 33 (1987) 155–171.
[17] E.T. Jaynes, The rationale of maximum entropy methods, Proc. IEEE 70 (1982) 939–952.
[18] Y. Jiang, H. Kautz, B. Selman, Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT, in: Proc. 1st Workshop on AI and OR, Timberine, OR, 1995.
[19] H. Kautz, B. Selman, Pushing the envelope: Planning, propositional logic, and stochastic search, in: Proc. AAAI-96, Portland, OR, 1996, pp. 1194–1201.
[20] S. Kirkpatrick, G. Toulouse, Configuration space analysis of traveling salesman problems, J. Phys. (France) 46 (1985) 1277–1292.
[21] S. Lin, B.W. Kernighan, An effective heuristic algorithm for the traveling salesman problem, Oper. Res. 21 (1973) 498–516.

[22] D. McAllester, B. Selman, H. Kautz, Evidence for invariants in local search, in: Proc. AAAI-97, Providence, RI, 1997, pp. 321–326.
[23] D. Mitchell, B. Selman, H. Levesque, Hard and easy distributions of SAT problems, in: Proc. AAAI-92, San Jose, CA, 1992, pp. 459–465.
[24] R. Monasson, R. Zecchina, S. Kirkpatrick, B. Selman, L. Troyansky, Determining computational complexity from characteristic 'phase transitions', Nature 400 (1999) 133–137.
[25] D.J. Patterson, H. Kautz, Auto-Walksat: A self-tuning implementation of Walksat, in: Electronic Notes in Discrete Mathematics (ENDM), vol. 9, Elsevier, Amsterdam, 2001, Presented at the LICS 2001 Workshop on Theory and Applications of Satisfiability Testing, June 14–15, Boston University, MA, 2001.
[26] J.C. Pemberton, W. Zhang, Epsilon-transformation: Exploiting phase transitions to solve combinatorial optimization problems, Artificial Intelligence 81 (1996) 297–325.
[27] B. Selman, H. Kautz, B. Cohen, Noise strategies for local search, in: Proc. AAAI-94, Seattle, WA, 1994, pp. 337–343.
[28] B. Selman, H. Levesque, D. Mitchell, A new method for solving hard satisfiability problems, in: Proc. AAAI-92, San Jose, CA, 1992, pp. 440–446.
[29] O. Telelis, P. Stamatopoulos, Heuristic backbone sampling for maximum satisfiability, in: Proc. 2nd Hellenic Conf. on Artificial Intelligence, 2002, pp. 129–139.
[30] Z. Xing, W. Zhang, A constraint sensitive algorithm for maximum satisfiability, 2003, in preparation.
[31] W. Zhang, Phase transitions and backbones of 3-SAT and maximum 3-SAT, in: Proc. Internat. Conf. on Principles and Practice of Constraint Programming (CP-01), Paphos, Cyprus, 2001, pp. 153–167.
[32] W. Zhang, M. Looks, Backbone guided local search for the traveling salesman, in: 5th Metaheuristic International Conference, Kyoto, Japan, 2003.
[33] W. Zhang, A. Rangan, M. Looks, Backbone guided local search for maximum satisfiability, in: Proc. IJCAI-03, Acapulco, Mexico, 2003, pp. 1179–1184.