SAT-Encodings, Search Space Structure, and Local Search Performance

Holger H. Hoos
University of British Columbia
Department of Computer Science
2366 Main Mall, Vancouver, B.C., Canada V6T 1Z4
hoos@cs.ubc.ca

Abstract

Stochastic local search (SLS) algorithms for propositional satisfiability testing (SAT) have become popular and powerful tools for solving suitably encoded hard combinatorial from different domains like, e.g., planning. Consequently, there is a considerable interest in finding SAT-encodings which facilitate the efficient application of SLS algorithms. In this work, we study how two encodings schemes for combinatorial problems, like the well-known Constraint Satisfaction or Hamilton Circuit Problem, affect SLS performance on the SAT-encoded instances. To explain the observed performance differences, we identify features of the induces search spaces which affect SLS performance. We furthermore present initial results of a comparitive analysis of the performance of the SAT-encoding and -solving approach versus that of native SLS algorithms directly applied to the unencoded problem instances.

1 Introduction

In the past few years, the development of extremely fast stochastic algorithms for the propositional satisfiability problem (SAT) have stirred considerable interest in the AI community. Modern stochastic local search algorithms, like WalkSAT and its more recent variants [McAllester *et al.*, 1997], can solve hard instances with several thousand variables and ten thousands of clauses within minutes of computing time. While only a very few "real-world" problems come as instances of SAT, many of these are combinatorial problems with quite natural CSP-like formulations, which can be easily encoded into SAT. In practice, however, for solving problems by encoding them into SAT and applying state-of-the-art SAT solvers, one has to find encodings which cannot only be efficiently generated, but which are also efficiently solvable by the respective SAT-solvers. While recent successes in solving SAT-encoded planning problems suggest that such efficient encodings can be found, our understanding of how encodings affect the performance of modern SAT algorithms and how good encodings can be found is very limited. "Characterizing the computational properties of different encodings of a real-world problem domain, and/or giving general principles that hold over a range of domains" has therefore been proposed as a challenge for propositional reasoning at IJCAI'97 [Selman *et al.*, 1997].

We approach this problem by studying the impact of encoding strategies on search space structure and the performance of stochastic local search (SLS) algorithms. In particular, we investigate the following questions:

1. Does it pay off to minimise the number of propositional variables, *i.e.*, are compact encodings which achieve small search spaces preferable to the bigger, but conceptually simpler sparse encodings?
2. Is the generic problem solving approach using SAT-encodings and modern SLS algorithms for SAT competitive with applying SLS algorithms to the un-encoded problem instances?

Our investigation is based on two case-studies, covering sets of hard instances of the NP-hard Constraint Satisfaction Problem (CSP) and Hamilton Circuit Problem (HCP). Building on earlier work on search space structure and SLS performance [Clark *et al.*, 1996; Yokoo, 1997; Frank *et al.*, 1997] and SAT encodings [Ernst *et al.*, 1997], we empirically analyse the impact of different encoding strategies on SLS performance and provide explanations for our observations by identifying search space features correlated with the observed SLS performance. Furthermore, we compare the performance of SLS algorithms on the best SAT-encoding of hard random binary CSP encodings with the performance of the best known native CSP algorithm based on stochastic local search.

As a preview of the results presented in the next sections, we briefly summarise our answers to the questions from above — however, given the general nature of these questions and the limited scope of the underlying empirical evidence these answers are tentative and should be understood as testable hypotheses.

1. According to our results for CSP and HCP, it seems to be much more advisable to use sparse rather than compact encodings, as the search spaces produced by the compact encodings are smaller but have characteristic features which impede local search.
2. When comparing the performance of SLS algorithms for CSP and SLS-based SAT algorithms applied to SAT-encoded CSPs, we observe a surprisingly small advantage for the direct CSP solving approach, which is outweighed by other advantages of the generic SAT-

encoding and solving approach such as the availability of very efficient implementations.

The remainder of this paper is structured in the following way. Section 2 gives some background on SLS algorithms for SAT and introduces the problems classes and test-sets used for our empirical analysis. Section 3 reviews previous work on search space structure and its impact on SLS performance and presents our approach to search space structure analysis. Sections 4–6 present the empirical investigations of the three questions from above, and Section 7 contains some conclusions and points out directions for future research.

2 Background

Stochastic local search approaches for SAT became prominent in 1992, when independently Selman, Levesque, and Mitchell [Selman *et al.*, 1992] as well as Gu [Gu, 1992] introduced algorithms based on stochastic local hill-climbing which could be shown to outperform state-of-the-art systematic SAT algorithms on a variety of hard subclasses of SAT. Since then, numerous other SLS schemes for SAT have been proposed. To date, state-of-the-art SLS algorithms can solve hard SAT problems up to several thousand variables, including SAT-encoded problems from other domains. In the recent past, especially the successful applications of SLS-based SAT algorithms for solving SAT-encoded planning problems, have stirred considerable interest in the AI community [Kautz and Selman, 1996; Ernst *et al.*, 1997; Kautz and Selman, 1998].

The algorithms considered here are model finding algorithms for CNF formulae. The underlying state space is always defined as the set of all assignments for the variables appearing in the given formula. Local search steps modify at most the value assigned to one of the propositional variables appearing in the formula; such a move is called a *variable flip*. The objective function is generally defined as the number of clauses which are unsatisfied under a given variable assignment; thus, the models of the given formula correspond to the global minima of this function. The general idea for finding these is to perform stochastic hill-climbing on the objective function, starting from a randomly generated initial assignment.

The main difference between the individual algorithms lies in the strategy used to select the variable to be flipped next. In this paper, we focus on the well-known WalkSAT family of algorithms [McAllester *et al.*, 1997], which provided a substantial driving force for the development of SLS algorithms for SAT and have been extremely successful when applied to a broad range of problems from different domains. WalkSAT algorithms start from a randomly chosen variable assignment and repeatedly select one of the clauses which are violated by the current assignment. Then, according to some heuristic a variable occurring in this clause is flipped using a greedy bias to increase the total number of satisfied clauses. For the original WalkSAT algorithm, in the following referred to simply as WalkSAT, the following heuristic is applied. If in the selected clause variables can be flipped without violating other clauses, one of these is randomly chosen. Otherwise, with a fixed probability p a variable is randomly chosen from the clause and with probability $1-p$ a variable is picked which minimises the number of clauses which are currently satisfied but would become violated by the variable's flip (number of breaks). The walk probability p (also called the *noise parameter*) has an important influence on the algorithms' overall performance. In this paper, we always use approximately optimal noise parameter settings for evaluating SLS performance; these settings are experimentally determined such that they minimise the expected number of flips for solving the given problem instance.

For the empirical study presented here, we use two well-known classes of combinatorial problems: Random binary Constrained Satisfaction Problems (CSPs) and Random Hamilton Circuit Problems (HCPs). Both binary CSPs and HCPs form NP-complete problem classes. Random binary CSPs have been studied extensively by various researchers, especially in the context of phase transition phenomena and their impact on the performance of CSP algorithms [Smith and Dyer, 1996; Prosser, 1996]. To obtain a test-set of hard problem instances, CSP instances with 20 variables and domain size 10 were sampled from the phase transition region characterised by a constraint graph density of $\alpha = 0.5$ and a constraint tightness of $\beta = 0.38$. Filtering out the insoluble instances with a complete CSP algorithm, test-set `csp20-10`, containing 100 soluble instances from this problem distribution, was generated.

For empirically investigating SLS performance for different SAT-encodings of the HCP, we focussed on the Hamilton Circuit Problem in directed random graphs. For a given graph, the HCP is to find a cyclic tour (Hamilton Circuit) using the edges of the graph which visits every vertex exactly once. In earlier work, when investigating the dependence of the existence of Hamilton Circuits on the average connectivity (edges per vertex), a phase transition phenomenon was observed [Cheeseman *et al.*, 1991]. The associated peak in hardness for backtracking algorithms was located between $\kappa = e/(n \log n) \approx 0.9$ and 1.2, where n is the number of vertices and e the number of edges [Frank and Martel, 1995]. Based on these results, we created a test-set by randomly sampling soluble HCP instances from distributions of directed random graphs with $n = 10$ and $\kappa = 1$. The test-set was generated using a HCP generator and solver developed and provided by Joe Culberson; insoluble instances were filtered out using the integrated systematic HCP solver, such that the final test-set `hcp10` contains 100 soluble instances.

3 Search Space Structure and SLS Performance

Obviously, the behaviour of SLS algorithms is determined by the topology of the search space, *i.e.*, the objective function induced by a specific problem instance. Recently, a growing number of researchers have been investigating the nature of this dependency for SLS-based SAT algorithms [Clark *et al.*, 1996; Yokoo, 1997; Frank *et al.*, 1997]. However, all of the studies we are aware of are restricted to Random-3-SAT, a widely used class of randomly generated SAT problems, while SAT-encoded problems from other domains have not been addressed. In [Clark *et al.*, 1996], it has been shown that for a given problem instance, its number of solutions is strongly negatively correlated with the performance of some SLS algorithms for SAT and CSP based on hill-climbing. This confirms the intuition that instances with a high solution density, *i.e.*, a large number of solutions, tend to be much easier to solve for SLS algorithms than instances with very few

solutions. [Yokoo, 1997] explains the peak in local search cost observed at the phase-transition region of Random-3-SAT in terms of the number and size of local minima regions in the search space. His analysis, however, relies on an exhaustive analyses of the search space and is therefore restricted to very small problem instances. [Frank *et al.*, 1997] analyse the topology of search spaces induced by Random-3-SAT formulae, but do not investigate the concrete impact of these features (such as the size of local minima) on SLS performance.

The approach taken here is to compare both the impact of encoding strategies on SLS performance as well as to provide explanations of the observed correlation in terms of search space features induced by the different encodings. To be not restricted to very small problem instances which possibly do not reflect the typical situation as encountered for realistic problems (*i.e.*, instances which can be solved with state-of-the art SLS algorithms), we restrict our analysis to features with can be determined without exhaustive search of large parts of the search space. The features we measure are:

- the solution density (number of solutions/search space size);
- the standard deviation of the objective function ($sdnclu$);
- the local minima branching along SLS trajectories ($blmin$).

As the solution density is known to be an important factor for SLS performance, it should be taken into account although it is generally time-consuming to measure. For counting the number of solutions of a given problem instance, we applied a modified version of the ASAT algorithm [Dubois *et al.*, 1993] to the SAT-encoded instances. To our best knowledge, both $sdnclu$ and $blmin$ have not been studied before in the context of search space structure.

Intuitively, large $sdnlcu$ values should indicate a rugged search space structure for which SLS approaches based on hill-climbing are more effective than for featureless, flat search spaces with many plateaus. To measure $sdnclu$ values, we determine the objective function value (number of unsatisfied clauses) for a sample of 100,000 randomly chosen assignments. To improve comparability of the results for different numbers of clauses, we scale objective function values to the interval $[0, 1]$ (by division by the number of clauses). The $sdnclu$ value is then computed as the empirical standard deviation over this sample. Although compared to the search space sizes of our test instances the sample size is very small, we get meaningful results — presumably because of the global effects of the encoding strategies on search space topology.

Since it is known that the number and size of local minima plays an important role for SLS performance on Random-3-SAT [Yokoo, 1997], we extended these results by studying the structure of local minima regions. We do this by measuring the average branching of local minima states, *i.e.*, for a given variable assignment, the number of neighbouring assignments[1] with the same objective function value. The underlying intuition is that highly branched local minima regions are more difficult to escape from, as there are fewer escape routes for random walk and similar plateau escape techniques, but more possibilities for non-deterministic loops to occur in the search trajectory. Local minima states are typically quite rare for the problem instances studied here; thus, $blmin$ cannot be measured by randomly sampling the search space. Instead we sampled the local minima states along SLS trajectories, using a sample size of 100,000. Again, to improve comparability of the results between different problem instances, we scale the values thus obtained to the interval $[0, 1]$ (by division by the number of variables).

For a given problem instance, WalkSAT's performance (denoted as lsc — local search cost) is measured as the expected number of flips per solution, determined from 100–1,000 runs per instance using an approximately optimal noise setting (see above) and a cutoff parameter high enough to guarantee that in every run a solution was found. [2]

4 Compact versus Sparse Encodings

Many combinatorial problems, such as Number Partitioning, Bin Packing, or Hamilton Circuit, can be quite naturally formulated as discrete constraint satisfaction problems. When encoding such CSP formulations into SAT, perhaps the most intuitive way is to encode each assignment of a value to a CSP variable by a different propositional variable [de Kleer, 1989]. We call this the *sparse encoding*, since it results in relatively sparse constraint graphs[3] for the resulting CNF formulae. This encoding strategy requires $|D| \cdot n$ variables, where $|D|$ is the domain size and n the number of CSP variables.[4]

Given the intuition that high solution densities should facilitate local search (see above), it seems to be worthwhile to consider encodings which minimise the number of propositional variables, and therefore the potential search space size without affecting the number of solutions. One encoding strategy which achieves that is the *compact encoding* obtained by representing each value assignment to a CSP variable binarily using a set of $\lceil log_2|D| \rceil$ propositional variables.Compared to the sparse encoding, this strategy significantly reduces the search space by a factor of $O(n \cdot (|D| - log|D|))$, while the number of clauses is usually similar, as it is usually dominated by clauses encoding the constraint relations (the number of which is identical for both encodings). The compact encoding is well-known from the literature; it has been proposed in the context of SAT variable complexity by [Iwama and Miyazaki, 1994] and is also used in the SAT-based MEDIC planning system, where it is called "factored representation" [Ernst *et al.*, 1997].

To investigate the impact of these two encoding schemes on SLS performance we measured WalkSAT's performance

[1]The neighbourhood relation is the same as used by all GSAT-type algorithms, *i.e.*, two assignments are neighbours if and only if they differ in the truth value of exactly one variable.

[2]Our actual experimental methodology is based on measuring run-time distributions (RTDs) as outlined in [Hoos and Stützle, 1998]; the RTD data is not reported here, but can be obtained from the author. A more detailed description of the empirical study and its results can be found in [Hoos, 1998] and will be presented in more detail in an extended version of this paper.

[3]The constraint graph consists of one node for each propositional variable and edges between nodes corresponding to variables which occur together in some clause of the given CNF formula.

[4]For simplicity's sake we assume that the domain sizes for all CSP variables are identical.

on the test-sets of Random Binary CSP and HCP instances described above. The HCP instances are encoded as CSPs by focussing on vertex permutations of the given graph as solution candidates (this idea has been proposed in [Iwama and Miyazaki, 1994]); in particular, for each vertex v we introduce a CSP variable the value of which represents v's position in the permutation. The constraint relations ensure that each vertex appears exactly once (type 1 constraints), *i.e.*, the candidate solution corresponds to a valid permutation of the vertices, and that each pair of neighbouring vertices in this (cyclic) permutation is connected by an edge in the given graph (type 2 constraints).

Figure 1 shows the correlation of the average local search cost (lsc) between the different encodings across the test-set `csp20-10`; each data point corresponds to the lsc for the sparse *vs* the compact encoding for one problem instance from the original test-set. As can be seen from the scatter plot, there is a strong linear correlation between the logarithm's average local search cost for both encodings (correlation coefficient $r = 0.95$), *i.e.*; instances which are relatively hard for WalkSAT when sparsely encoded also tend to be hard when using the compact encoding. But this analysis also shows that the compact encoding results in instances for which the lsc is generally ca. 7 times higher than for the corresponding sparsely encoded instances, although the compact encoding requires less then half the number of variables and significantly fewer clauses than the sparse encoding (*cf.* Table 1). This confirms earlier observations [Ernst *et al.*, 1997] that the compact encoding generates problem instances which are typically extremely hard for stochastic local search.

But what exactly makes this encoding so ineffective? Table 1 shows the solution density, $sdnclu$, and $blmin$ values for the easiest, median, and hardest (w.r.t. their lsc values for WalkSAT) problem instance from the test-set `csp20-10`. The data confirms that within the test-set, the solution density is the predominant factor affecting local search cost, as earlier observed for Random-3-SAT test-sets [Clark *et al.*, 1996]. However, between the two encodings, this does not hold. Instead, the $sdcnlu$ and the $blmin$ values indicate that the compact encoding induces a flatter, featureless search space topology characterised by considerably higher branched local minima states. As argued above, intuitively this makes local search more difficult, which is consistent with the considerably higher local search cost observed for WalkSAT on the correspondingly encoded CSP instances. Interestingly, this explanation is also quite consistent with the (relatively big) differences in local search cost within the corresponding test-sets, although the relatively small differences in $sdcnlu$ and $blmin$ seem to confirm the predominant role of the solution density.

Applying the same analysis to sparsely and compactly encoded versions of the HCP test-set `hcp10` gives exactly analogous results (*cf.* Table 1); the correlation coefficient for the correlation between $\log(lsc)$ for -s and -c is 0.85) and thus confirms our observations and interpretation given above. Apparently, the compact encoding induces rather flat, featureless search spaces which impede local search and local minima escape to such an extent that WalkSAT's performance is significantly reduced despite the much higher solution density achieved by reducing the number of variables.

There is, however, another factor to be considered. The

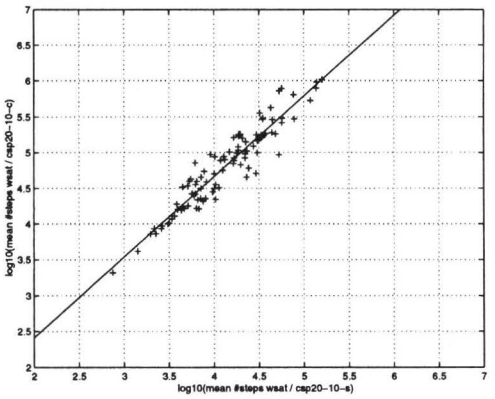

Figure 1: Correlation of average local search cost per instance between sparse (horizontal) and compact SAT-encoding (vertical) for CSP instances from test-set `csp20-10` when using WalkSAT with approx. optimal noise.

CPU-time required for each single flip performed by WalkSAT is roughly constant for a given problem instance; it depends, however, considerably on the syntactical features of the given formula (especially number and length of clauses). Comparing the CPU-time per variable flip between the formulae generated by the sparse and compact encoding schemes reveals that for the compact encoding also the CPU-time per flip is significantly higher (for both, `csp20-10-c` and `hcp10-c` instances approximately by a factor of 3) than for the sparse encoding. This can be explained by the fact that the compact encoding produces longer and more tightly connected clauses (in terms of number of clauses which share at least one variable with any given clause), such that each variable flip affects potentially more clauses (by breaking or fixing them). So the compact encoding induces both, a syntactic structure (the longer and more tightly connected clauses) and a semantic structure (the search space features discussed above) which reduce WalkSAT's performance by increasing the CPU-time per search step and decreasing the efficiency of these search steps.

5 To Encode or Not to Encode?

The results presented in the preceeding sections suggest that when solving combinatorial problems by encoding them into SAT and applying powerful SLS algorithms, there might be a tradeoff between the size of the representation and SLS performance. Apparently compact encodings can be used to generate formulae with small search spaces; however, these have characteristic features which impede the performance of WalkSAT and similar SAT algorithms. One obvious question therefore is the following. Given a formulation of a combinatorial problem as a CSP, should we encode into SAT at all, or rather apply SLS algorithms for CSP to directly solve the unencoded problem?

Here, we present some initial results on this question. Specifically, we analyse the correlation between the performance of WalkSAT and an WMCH, an analogous algorithm for CSP which is based on the Min Conflicts Heuristic [Minton *et al.*, 1992] and has been introduced in [Steinmann *et al.*, 1997]. Like WalkSAT, WMCH is a SLS algorithm

instance	encoding	variables	clauses	avg. *lsc*	num sln	sln density	*sdnclu*	*blmin*
csp20-10-s/easy	sparse	200	3,661	775.80	37,297	$2.01 \cdot 10^{-56}$	0.0366	0.015
csp20-10-s/med	sparse	200	4,367	11,162.69	6	$3.73 \cdot 10^{-60}$	0.0361	0.017
csp20-10-s/hard	sparse	200	4,412	134,777.38	2	$1.24 \cdot 10^{-60}$	0.0356	0.017
csp20-10-c/easy	compact	80	2,861	2,249.19	37,297	$3.09 \cdot 10^{-20}$	0.0011	0.131
csp20-10-c/med	compact	80	3,555	77,883.52	6	$4.96 \cdot 10^{-24}$	0.0011	0.157
csp20-10-c/hard	compact	80	3,612	1,000,456.82	2	$1.65 \cdot 10^{-24}$	0.0011	0.161
hc10-s/easy	sparse	100	1,060	215.97	60	$4.73 \cdot 10^{-29}$	0.0504	0.059
hc10-s/med	sparse	100	1,060	781.28	20	$1.58 \cdot 10^{-29}$	0.0502	0.056
hc10-s/hard	sparse	100	1,060	2,562.18	20	$1.58 \cdot 10^{-29}$	0.0502	0.047
hc10-c/easy	compact	40	1,110	548.37	60	$5.46 \cdot 10^{-11}$	0.0017	0.187
hc10-c/med	compact	40	1,110	1,258.55	20	$1.82 \cdot 10^{-11}$	0.0017	0.181
hc10-c/hard	compact	40	1,110	3,294.71	20	$1.82 \cdot 10^{-11}$	0.0016	0.172

Table 1: Easy, median, and hard problem instances from compactly and sparsely encoded CSP and HCP test-sets; the table shows the size of the instances, the expected local search cost for WalkSAT (in steps/solution, using approx. optimal noise), as well as the number of solutions, solution density, normalised *sdnclu* and average *blmin* value (for details, see text).

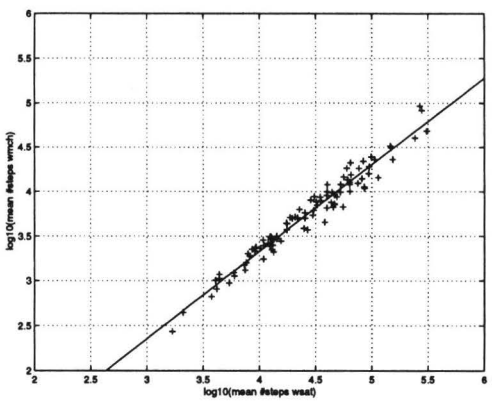

Figure 2: Correlation of average local search cost per instance for WalkSAT *vs* WMCH applied to test-set `csp20-10`; both algorithms use approx. optimal noise, WalkSAT is used with the sparse SAT-encoding.

which iteratively repairs violated constraints following the steepest gradient, but also allows to escape from local minima of the objective function by allowing occasional, randomised uphill moves (for details, see [Steinmann *et al.*, 1997]). Like for WalkSAT, the local search cost (*lsc*) for WMCH was measured as the expected number of local search steps to find a solution, based on 100 runs of the algorithm for each given problem instance. Generally, we always used approximately optimal noise parameter settings and a cutoff parameter high enough to guarantee that in every run a solution was found.

Analysing the correlation between the average local search cost for WalkSAT (using the sparse encoding) and WMCH across the test-set `csp20-10` reveals an extremely strong linear correlation between the logarithms of the average local search cost for both algorithms (*cf.* Figure 2; correlation coefficient $r = 0.98$), *i.e.*, instances which are relatively hard for WalkSAT also tend to be hard for WMCH and vice versa. A linear regression analysis of this data shows that WalkSAT tends to require ca. 5 times more steps than WMCH for solving the problem instances from our test-set (the coefficients of the regression analysis are $a = 0.98$ and $b = -0.57$).

However, neither WalkSAT nor WMCH are the best known SLS algorithms for SAT and CSP, respectively. We therefore performed the same analysis for the best-performing SLS-based SAT algorithm and the best SLS-based CSP algorithm we are aware of — Novelty [McAllester *et al.*, 1997], one of the most recent variants of WalkSAT, and the tabu search algorithm by Galinier and Hao [Galinier and Hao, 1997]. [5] Using optimal noise parameters for both algorithms, we find that when measuring the average number of local search steps per solution, Galinier's and Hao's CSP algorithm has generally only an advantage of a factor between 2 and 4 over Novelty. The correlation between both algorithm's performance is very strong (*cf.* Figure 3; correlation coefficient = 0.96); however, Novelty occasionally gets stuck in local optima which it cannot escape (causing the outliers in the scatter plot 7 instances), while the CSP algorithm shows no such behaviour. But this happens only for 7 of our 100 instances and for these only in a small number of the multiple tries we performed. We conjecture that by exending Novelty with a stronger stochastic escape mechanism, this phenomenon can be eliminated. But except for these outliers, the regression analysis indicates that both algorithms show approximately the same scaling behaviour w.r.t. instance hardness. This is somewhat surprising, since Galinier's and Hao's algorithm and Novelty are conceptually significantly less closely related than WMCH and WalkSAT. When performing an exactly analogous analysis for instances with 30 constraint variables and 10 values, we find a similar situation; only now, the performance advantage of the native CSP algorithm over Novelty is only a factor of ca. 1.7 on average (comparing the number of local search steps). Also, we observe no increased occurence of the outliers mentioned above. This suggests that when increasing the problem size, the SAT-based approach might have a slight scaling advantage over the native CSP algorithm.

Generally, our results from comparing the performance of SLS-based algorithms for CSP and SAT indicate that when measuring the average number of steps per solution, the dif-

[5] The considerably more complicated R-Novelty algorithm [McAllester *et al.*, 1997] for SAT, which shows an even better performance than Novelty on Random-3-SAT, is inferior to Novelty for the random binary CSPs used here.

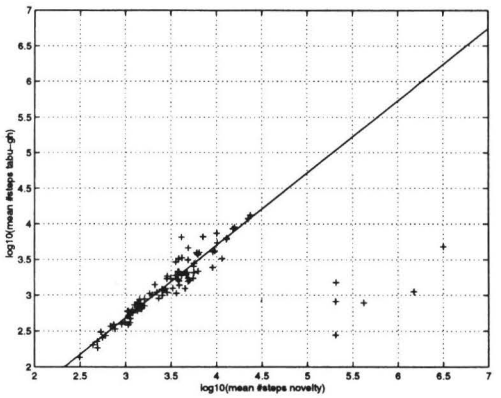

Figure 3: Correlation of average local search cost per instance for Novelty *vs* Galinier's and Hao's tabu search algorithm for CSP applied to test-set `csp20-10`; all algorithms use approx. optimal noise, Novelty is used with the sparse SAT-encoding.

ferences are rather surprisingly small. We deliberately refrained from comparing CPU-times for these algorithms, because the implementations for the SAT algorithms are significantly more optimised. The SAT algorithms generally have the advantage that they are conceptually easier, which facilitates their efficient implementation, evaluation, and further development. Of course, our analysis is too limited to give a conclusive answer, but the results reported here suggest that for solving hard CSP instances, encoding them into SAT and using a state-of-the-art SLS algorithm for SAT to solve the SAT-encoded instances might be very competitive compared to using more specialised SLS-based CSP solvers.

6 Conclusions and Future Work

In this paper we presented an initial investigation of how SAT-encoding strategies affect the topological structure of the induced search spaces and SLS performance. Our empirical results show that for two well-known classes of NP-hard combinatorial problems, random binary CSPs and HCPs in random directed graphs near the solubility phase transition, compact SAT-encodings, which minimise the number of propositional variables for the encoded problem instances, exhibit structural features which impede local search algorithms like WalkSAT. This observation is consistent with earlier observations for planning problems and HCPs, that compact encodings are extremely hard to solve for SLS algorithms. At the same time, it shows that the influence of solution density on SLS performance which has been observed for CSP and SAT, is of minor significance compared to features like the overall ruggedness of the search space (as measured by the standard deviation of the objective function) or the local minima branching. However, consistently with earlier results, it plays an important role in explaining the differences in SLS performance observed across test-sets of randomly generated instances, when using a fixed SAT-encoding. As a consequence of these results, compact encodings should be avoided in the context of using SLS algorithms for solving SAT-encoded combinatorial problems.

Furthermore, when comparing the performance of SLS algorithms directly applied to test-sets of random binary CSP instances to with the performance of SLS-based algorithms for SAT using the sparse encoding, we found that there is a strong correlation of the expected number of search steps required for finding a solution across the test-set. Although when comparing the best known SLS algorithms for CSP and SAT in this way, the direct CSP algorithm requires fewer local search steps per solution, we feel that this might very well be outweighed by other advantages of the SAT-encoding and solving approach, like the availability of extremely fast implementations or its conceptual simplicity. Also, due to the restrictions of the implementatins of the SAT-based CSP algorithms available for this preliminary study only binary CSP instances could be analysed. For more structured, none-binary CSP instance, one might encounter a different situation; although it is not clear that (a) existing CSP algorithms can exploit this structure, and (b) even if they could, the same would not also be possible without too much effort for the SAT-encoded problem instances.

It should also be noted that, according to the results reported here and elsewhere [Ernst *et al.*, 1997; Kautz and Selman, 1996], it seems that finding efficient SAT-encodings could very well be significantly easier than developping specialised algorithms for the respective domains. We do not believe that the generic problem solving approach based on SAT-encodings and extremely optimised SAT-algorithms will generally be competitive with specialised algorithms which make use of domain knowledge. However, it is quite possible that it can be established as another generic method for attacking problems for which domain-specific knowledge is not (yet) available or which are too specific and limited in their application to make the development of specialised algorithms worthwhile. In this sense, the SAT-encoding and -solving approach could be compared to other generic problem solving methods, like, e.g., Mixed Integer Programming (MIP), which has become a standard generic solving technique for combinatorial problems in Operations Research.

The novel tools and techniques presented here can be easily applied to other hard combinatorial problems and SAT-encoding schemes. Out investigation can be extended in various directions; in particular, it provides a basis for studying the following research questions:

- Based on the knowledge of the search space structure induced by specific encoding schemes, can we design SAT algorithms which exploit these features?
- Given the knowledge about which features affect SLS performance, can we devise efficient encoding strategies which improve SLS performance?
- For which problem classes is the SAT-encoding and -solving approach, using SLS algorithms for SAT, competitive with analogous, generic SLS algorithms directly applied to the unencoded problem instances?

Thus, while this study does certainly not provide the final answers to the challenge or the more specific questions stated in Section 1, it shows a way for systematically investigating the relation between encodings, search space structure, and SLS performance which will hopefully lead to improved encodings and SAT algorithms, as well as, in the long run, to a realistic assessment of the generic SAT-encoding and -solving approach for solving hard combinatorial problems.

Acknowledgements

The CSP instances used for the experimental part of this paper were generated using software provided by Thomas Stützle whom I also wish to thank for many interesting and stimulating discussions. Furthermore, I thank Bart Selman for his encouraging comments on an earlier version of this paper as well as Wolfgang Bibel and the Intellectics Group at TU Darmstadt for their support during a significant part of this work. Finally, I gratefully acknowledge valuable input by David Poole and the members of the Laboratory of Computational Intelligence at the University of British Columbia. This research was partially supported IRIS Phase-III Project BOU, "Preference Elicitation and Interactive Optimization."

References

[Cheeseman *et al.*, 1991] P. Cheeseman, B. Kanefsky, and W. Taylor. Where the Really Hard Problems Are. In *Proc. IJCAI'91*, pages 331–337, 1991.

[Clark *et al.*, 1996] D. Clark, J. Frank, I. Gent, E. MacIntyre, N. Tomov, and T. Walsh. Local Search and the Number of Solutions. In *Proc. CP'96*, LNCS, vol. 1118 , pages 119–133. Springer Verlag, 1996.

[Dubois *et al.*, 1993] O. Dubois, P. Andre, Y. Boufkhad, and J. Carlier. SAT versus UNSAT. In *2nd DIMACS Implementation Challenge*, 1993.

[de Kleer, 1989] J. de Kleer. A Comparison of ATMS and CSP Techniques. In *Proc. IJCAI'89*. Morgan Kaufmann, 1989.

[Ernst *et al.*, 1997] M. Ernst, T. Millstein, and D. Weld. Automatic SAT-Compilation of Planning Problems. In *Proc. IJCAI-97*, 1997.

[Frank and Martel, 1995] J. Frank C. Martel. Phase Transitions in the Properties of Random Graphs. In *CP'95 Workshop on Studying and Solving Really Hard Problems*, 1995.

[Frank *et al.*, 1997] J. Frank, P. Cheeseman, and J. Stutz. When Gravity Fails: Local Search Topology. *JAIR*, 7:249–281, 1997.

[Galinier and Hao, 1997] P. Galinier and J. Hao. Tabu Search For Maximal Constraint Satisfaction Problems. In *Proc. CP'97*, LNCS, vol. 1330, pages 196–208. Springer Verlag, 1997.

[Gu, 1992] J. Gu. Efficient Local Search for Very Large-scale Satisfiability Problems. *ACM SIGART Bulletin*, 3(1):8–12, 1992.

[Hoos, 1998] Hoos, H. H. 1998. *Stochastic Local Search — Methods, Models, Applications*. Ph.D. Dissertation, Fachbereich Informatik, Technische Universität Darmstadt.

[Hoos and Stützle, 1998] H. H. Hoos and T. Stützle. Evaluating Las Vegas algorithms — pitfalls and remedies. In *Proc. UAI'98*, 1998.

[Iwama and Miyazaki, 1994] K. Iwama and S. Miyazaki. SAT-Variable Complexity of Hard Combinatorial Problems. In *Proc. IFIP World Computer Congress*, pages 253–258. Elsevier Science B.V., North Holland, 1994.

[Kautz and Selman, 1996] H. Kautz and B. Selman. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Proc. AAAI'96*, volume 2, pages 1194–1201, 1996.

[Kautz and Selman, 1998] H. Kautz and B. Selman. The role of Domain-specific Knowledge in the Planning as Satisfiability Framework. In . *Proc. AIPS-98*, 1998.

[McAllester *et al.*, 1997] D. McAllester, B. Selman, and H. Kautz. Evidence for Invariants in Local Search. In *Proc. AAAI'97*, pages 321–326, 1997.

[Minton *et al.*, 1992] S. Minton, M.D. Johnston, A.B. Philips, and P. Laird. Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence*, 52:161–205, 1992.

[Prosser, 1996] P. Prosser. An Empirical Study of Phase Transitions in Binary Constraint Satisfaction Problems. *Artificial Intelligence*, 81:81–109, 1996.

[Selman *et al.*, 1992] B. Selman, H. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proc. AAAI'92*, pages 440–446, 1992.

[Selman *et al.*, 1997] B. Selman, H. Kautz, and D. McAllester. Ten Challenges in Propositional Reasoning and Search. In *Proc. IJCAI-97*, 1997.

[Smith and Dyer, 1996] B. Smith and M. Dyer. Locating the Phase Transition in Binary Constraint Satisfaction Problems. *Artificial Intelligence*, 81:155–181, 1996.

[Steinmann *et al.*, 1997] O. Steinmann, A. Strohmaier, and T. Stützle. Tabu Search vs. Random Walk. In *Advances in Artificial Intelligence (KI-97)*, LNAI, vol. 1303, pages 337–348. Springer Verlag, 1997.

[Yokoo, 1997] M. Yokoo. Why Adding More Constraints Makes a Problem Easier for Hill-Climbing Algorithms: Analyzing Landscapes of CSPs. In *Proc. CP'97*, LNCS, vol. 1330, pages 357–370. Springer Verlag, 1997.