Chapter 7

SAT Encodings and Search Space Structure

SLS algorithms for SAT have been successfully applied to hard combinatorial problems from different domains. On one hand, these applications have become possible by using powerful SLS algorithms like WalkSAT. But they also critically rely on the use of suitable SAT-encodings. Because of the $\mathcal{NP}$-completeness result for SAT [Coo71], we know that instances from any $\mathcal{NP}$-complete problem can be encoded in SAT, and even 3-SAT, in polynomial time; consequently, the size of the encoded instances is also polynomial in the size of the original representations. However, this alone does not guarantee the existence of suitable SAT-encodings, since sufficiently small and efficiently computable SAT representations could still be extremely hard for existing SLS algorithms. In particular, this might happen if the encoding induces search space features which impede local search. Thus, the analysis of how encoding strategies affect search space structure and SLS performance is a very interesting and relevant issue in the context of a generic problem solving approach based on SAT as an intermediate representation domain and SLS algorithms as solvers for this domain (*cf.* Chapter 1).

In this chapter, we investigate the impact of encoding strategies on SLS performance and search space structure. Since this is a complex issue which has rarely been studied before, we can only undertake an initial investigation here. After giving further motivation and background on SAT-encodings, we present two case studies. First, we study sparse versus compact encodings for hard Random Binary CSP instances. Furthermore, we compare the performance of a state-of-the-art SLS algorithm for CSP with the performance of some of the best SLS algorithms for SAT applied to instances from this domain. The second case study extends the analysis of sparse versus compact encodings to hard Random Hamilton Circuit instances. For this problem domain, we also investigate the effects of symmetry elimination on different problem representations and encodings. After a brief look on related work, the chapter closes with a summary of our main results and some conclusions.

7.1 Motivation and Background

Until now, our study of SLS algorithms, their behaviour, and their applications has been focussed on the algorithms themselves, the analysis and characterisation of their run-time behaviour, and on the factors which influence their performance. Coming back to the generic problem solving model from Chapter 1, one issue has not been addressed so far: the influence of the encoding strategies used for translating problems from other domains into the intermediate domain (in our case, SAT) on the performance of SLS algorithms for the encoded problem instances. In Chapter 4, where we evaluated a number of modern SLS algorithms for SAT on various problem classes, we always used encodings which are already known to be effective in the sense that they facilitate good SLS performance. However, devising such encodings is usually not simple. At the same time, finding effective encodings is a crucial condition for successfully applying the generic problem solving scheme.

When developing encodings, two fundamental issues are space and time complexity: As argued before, both encoding problem instances and decoding solutions have to be efficient enough that the overhead compared to directly using a solver for the original domain is not too high. Also, the encoded problem instances should not grow too large compared to the original representation. When using SAT as an intermediate domain, and applying SLS algorithms for solving the encoded instances, intuitively it seems that the number of propositional variables should be kept small, because this way, smaller search spaces are obtained. However, it is not clear whether encodings which are optimised for a minimal number of propositional variables are really advantageous, as they might induce search space features which significantly impede stochastic local search.

Furthermore, some problem domains (such as the Hamilton Circuit Problem discussed later in this chapter, or problems from various planning domains) allow different basic representations, where *a priori* it is not clear which of several representations is most efficient in conjunction with the generic problem solving approach outlined before. Another issue is symmetry elimination and pruning — depending on the original problem formulation, often the search space can be pruned before encoding the problem; also, problems have often symmetric solutions which can be collapsed before encoding them into the intermediate domain. This is the case for the Hamilton Circuit Problem, where for the basic representations we will use later, a symmetry is given by the starting point of the cyclic tour through a given graph. Again, *a priori* it is not clear, whether eliminating such symmetries is advantageous w.r.t. maximising SLS performance on the encoded problem instances.

Today, these and many other issues are not well understood. Clearly, at this point and in the context of this thesis we can only study some particular aspects of this widely open problem. In the following, we therefore restrict ourselves to three basic questions:

- Does it pay off to minimise the number of propositional variables, *i.e.*, are compact encodings which achieve small search spaces preferable to sparse encodings?
- Is it advisable to reduce the number of propositional variables required for a given en-

coding by eliminating symmetric solutions from the original problem formulation?

- Is the generic problem solving approach competitive with applying SLS algorithms directly to the un-encoded problem instances?

While we cannot give definitive answers to these questions here, the two case studies presented in the following demonstrate how these issues can be investigated and also suggest tentative answers which can and should be tested and further investigated in the future.

7.2 Constraint Satisfaction Problems

Constraint Satisfaction Problems have been formally introduced in Chapter 1. Like SAT, finite discrete CSP is an $\mathcal{NP}$-complete combinatorial problem. Conceptually it is very closely related to SAT; as a consequence, most SLS algorithms for SAT can be generalised to CSP in a rather straightforward way. Many combinatorial problems from different domains can be quite naturally represented as CSP instances; compared to SAT encodings of these problems, CSP representations are more compact and often easier to devise. Therefore, when transforming combinatorial problems into SAT, CSP is often implicitly or explicitly used as an intermediate representation (*cf.* Chapter 4, the SAT-encodings of the Graph Colouring and All-Interval-Series problems). Nevertheless, as mentioned before, SAT is conceptually simpler than CSP, and consequently, SAT algorithms are often easier to develop, implement, and evaluate. In this section, we investigate SAT-encodings of binary CSPs and compare the application of SLS-based SAT algorithms to SAT-encoded CSPs with directly solving CSP instances using SLS algorithms for CSP.

7.2.1 SAT Encodings of the CSP

CSP instances can be easily encoded into SAT by using propositional variables to represent the assignment of values to single CSP variables and clauses to express the constraint relations [dK89]. To simplify the formalism for specifying SAT encodings of discrete finite CSP instances, we assume (without loss of generality) that the variable domains D_i are all equal to $\mathbb{Z}_k = \{0, 1, \ldots, k-1\}$, where k is an arbitrary positive integer. Furthermore, we denote the arity of a constraint relation C_j by $\sigma(C_j)$.

A very natural SAT-encoding for a given CSP $P = (X, \mathcal{D}, \mathcal{C})$ with $X = \{x_1, \ldots, x_n\}$, $\mathcal{D} = \{D_1, \ldots, D_n\}$, and $\mathcal{C} = \{C_1, \ldots, C_k\}$ is based on propositional variables $c_{i,\nu}$ which, if assigned the value $\top$, represent the assignment $x_i = \nu$, where $\nu \in D_i$. P can then be represented by a CNF formula comprising the following sets of clauses:

(1)	$\neg c_{i,\nu_1} \vee \neg c_{i,\nu_2}$	$(1 \leq i \leq n; \nu_1, \nu_2 \in D_i; \nu_1 \neq \nu_2)$
(2)	$c_{i,1} \vee c_{i,2} \vee \ldots \vee c_{i,k}$	$(1 \leq i \leq n)$
(3)	$\neg c_{i_1,\nu_1} \vee \neg c_{i_2,\nu_2} \vee \ldots \vee \neg c_{i_s,\nu_s}$	$(x_{i_1} = \nu_1; x_{i_2} = \nu_2; \ldots; x_{i_s} = \nu_s$ violates some constraint $C \in \mathcal{C}$ with $\sigma(C) = s)$

Intuitively, these clause sets ensure that each constraint variable is assigned exactly one value from its domain (1, 2) and that this assignment is compatible with all constraints (3). Obviously, the number of propositional variables required for encoding a given CSP instance is linear in the number of constraint variables and their domain sizes while the number of clauses is at least linear in the number of constraint variables and depends critically on the domain sizes and the arity of the constraints. This encoding has been frequently used for translating problems from other domains into SAT (*cf.* Chapter 4, Section 4.4). We call it the *sparse encoding*, because it generates large SAT instances the models of which have only a small fraction of the propositional variables set to $\top$.

Since the search space size for a given formula is exponential in the number of propositional variables it comprises, it appears to be reasonable to minimise the number of variables required by a SAT-encoding. In the case of the SAT-encodings for CSP instances, the number of propositional variables required for encoding a given CSP instance can be significantly reduced compared to the sparse encoding. This is achieved by a binary encoding of the value assigned to a constraint variable x_i using a group of $\kappa = \lceil \log_2 k \rceil$ propositional variables $c_{i,j}$ (k is the domain size of the CSP instance). To formalise this *compact encoding*, we use an auxiliary function which maps a positive integer ν to a propositional clause which is true exactly if the corresponding assignment does *not* correspond to the binary representation of ν:

$$c(i,\nu) = \bigvee_{j=1}^{\kappa} l(\nu, j, c_{i,j})$$

$$l(\nu, j, z) = \begin{cases} \neg z & \textit{if } (\nu \text{ div } 2^{i-1}) \bmod 2 > 0 \\ z & \textit{otherwise} \end{cases}$$

Now, we can represent a CSP instance P by a CNF formula using the following sets of clauses:

(1)	$c(i,\nu)$	$(1 \leq i \leq n; k \leq \nu < 2^{\kappa})$
(2)	$c(i_1,\nu_1) \vee c(i_2,\nu_2) \vee \ldots \vee c(i_s,\nu_s)$	$(x_{i_1} = \nu_1, x_{i_2} = \nu_2, \ldots, x_{i_s} = \nu_s$ violates some constraint $C \in \mathcal{C}$ with $\sigma(C) = s)$

Intuitively, these groups of clauses ensure that no constraint variable is assigned an invalid value (1) and that the assignment does not violate any constraint in $\mathcal{C}$ (2). This compact encoding requires $O(n \cdot \log k)$ propositional variables which is a considerable reduction compared to the $O(n \cdot k)$ variables used for the sparse encoding. Usually, the clauses encoding

test-set	instances	variables	domain size	α	β	vars	clauses
csp20-10-s	100	20	10	0.5	0.38	200	$\approx$ 4,305
csp20-10-c	100	20	10	0.5	0.38	80	$\approx$ 3,504

Table 7.1: Random Binary CSP test-sets, using sparse (`-s`) and compact (`-c`) SAT-encoding; as the number of clauses varies between the instances, we report the average number of clauses here.

the constraint relations dominate the size of the formulae produced by both encodings, such that the number of clauses will be similar. But because of the difference in the number of propositional variables, the compact encoding generates search spaces which are smaller by a factor of $O(n \cdot (k - \log k))$.

7.2.2 Benchmark Instances

For our empirical study presented in the next sections, we focussed on Uniform Random Binary CSP, a random distribution of CSP instances based on random binary constraint relations. More specifically, the problem distribution is characterised by the *constraint graph density* α and the *constraint tightness* β; α specifies the fixed probability that between two randomly chosen constraint variables a constraint relation exists, while β is the expected fraction of value pairs which are compatible with a given constraint relation between two variables.

For this problem class, a phase transition phenomenon, similar to the one for Uniform Random-3-SAT, has been observed [Smi94]. To obtain a test-set of hard problem instances, CSP instances were sampled from the phase transition region of Uniform Random Binary CSP with 20 variables and domain size 10, 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. These instances were translated into corresponding sets of SAT instances using the sparse and the compact encodings (*cf.* Table 7.1). The generator and the test-sets used for the subsequent studies were kindly provided by Thomas Stützle (TU Darmstadt, Germany).

7.2.3 Two SLS Algorithms for CSP

To study the performance difference between SLS algorithms for SAT applied to SAT-encoded CSP instances and SLS algorithms for CSP directly applied to the CSP instances, we use two competetive variants of the well-known *Min-Conflicts Heuristic (MCH)* [MJPL92], which are obtained by extending the basic MCH with random walk *(WMCH)* and tabu-lists *(TMCH)*, respectively [SSS97]. These algorithms are conceptually closely related to

```
function WMC(a) is
  % stage 1: variable selection:
  randomly select a variable x_i occurring in a currently violated constraint;
  % stage 2: value selection:
  with probability wp do
    V' := set of values for x_i which minimise the number of conflicts;
  otherwise
    V' := D_i;
  end with;
  ν := draw(V');
  a' := a with x_i set to ν;
  return (a');
end WMC;
```

Figure 7.1: GLSM state for realising WMCH.

WalkSAT and WalkSAT+tabu and can be modelled as 1-state+init GLSMs using the same overall structure as GSAT (*cf.* Figure 4.1, page 75) but the WMC and TMC states defined in Figures 7.1 and 7.2 instead of the GD state, and using a modified initialisation state which assigns a randomly chosen value to each constraint variable and, for TMCH, initialises the tabu-list (initially empty).

For both algorithms, in each local search step, first a constraint variable involved in a violated constraint is randomly selected,[1] then a new value ν for this variable is chosen. The Min-Conflicts Heuristic's general approach for selecting the new value ν is to minimise the number of constraint violations (conflicts); thus, the scoring function is closely related to the one used by GSAT or Novelty. WMCH and TCMH use different mechanisms for selecting ν; while WMCH extends the basic selection strategy with random walk much in the spirit of WalkSAT, TCMH uses a tabu-list of variable/value pairs to avoid stagnation behaviour. Additionally, TMCH incorporates an aspiration criterion which allows values which are tabu to be selected when they improve on the overall best solution since the last restart.

7.2.4 SLS Performance

Our analysis of SLS performance is divided into three parts. First, we compare the performance of WalkSAT on the different encodings; as a result of this, we see that while the local search cost for the sparse encodings is uniformly lower than for the compact encoding, there is a tight correlation of SLS performance between both encodings of individual instances across the test-set. Next, we compare the different SLS-based SAT algorithms' performance when applied to the sparsely encoded test-set. As we will show, the performance of all SLS algorithms, when applied to the same problem instance, is closely correlated — which in-

[1] in this context, all random choices from sets are based on uniform distributions, *i.e.*, each element has the same probability for being selected.

```
function TMC(a) is
  % stage 1: variable selection:
  randomly select a variable x_i occurring in a currently violated constraint;
  % stage 2: value selection:
  s' := minimal number of conflicts when setting x_i to any value ∈ D_i;
  if s' < minimal number of conflicts ever encountered during this search then
    V' := set of values for x_i which minimise the number of conflicts;
  else
    V' := set of values for x_i which are currently not tabu;
    V'' := elements of V' which minimise the number of conflicts;
  end if;
  ν := draw(V');
  a' := a with x_i set to ν;
  if length of tabu-list > tl then
    remove oldest element from tabu-list;
  add (x_i, ν) to tabu list;
  return (a');
end TMC;
```

Figure 7.2: GLSM state for realising TMCH.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{25}	Q_{75}	$\frac{Q_{75}}{Q_{25}}$
csp20-10-s	23,408.70	28,770.74	1.23	12,516.37	6,168.41	30,260.64	4.91
csp20-10-c	137,689.53	199,287.07	1.45	70,042.56	22,841.89	162,885.79	7.13

Table 7.2: Test-set `csp20-10`, basic descriptive statistics of hardness distributions for sparse and compact SAT-encoding, using WalkSAT (approx. optimal noise, 100 tries/instance).

dicates that the performance of the different SLS methods depends on the same intrinsic features of the problem instances. Finally, we compare the performance of SLS algorithms for CSP applied to the original CSP instances with that of the SLS-based SAT algorithms applied to the sparsely encoded test-set; the result of this analysis indicates that the latter approach is quite competitive.

Sparse *vs* Compact SAT Encodings

To investigate the performance differences for SLS-based SAT algorithms between the two different encodings, we first determined the distribution of the average local search cost (hardness distribution) for WalkSAT across the test-sets `csp20-10-s` and `csp20-10-c`. For these and all the following experiments, we used approx. optimal noise settings and cutoff parameters high enough ($\mathsf{maxSteps} \geq 10^6$) to guarantee maximal success probabilities. The average local search cost for each instance is estimated from 100 tries. The results, as can be seen in Figure 7.3, clearly indicate that WalkSAT works significantly more efficient on the sparse than on the compact encoding. Furthermore, it can be seen from the normalised

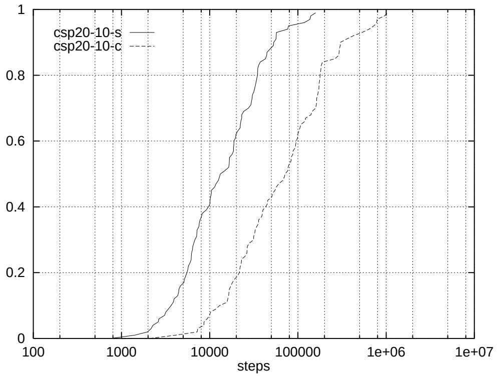

Figure 7.3: Hardness distributions (average local search cost per instance) for WalkSAT, using approx. optimal noise, when applied to test-sets `csp20-10-s` and `csp20-10-c` (sparse *vs* compact SAT-encoding for CSP).

standard deviation and the percentile ratios of the hardness distributions (*cf.* Table 7.2) that the compact encoding induces more variability between the instances than the sparse encoding. Note that the variability for the sparse encoding is comparable to Random-3-SAT at the phase transition (*cf.* Table 4.10, page 96).

To investigate this in more detail, next, we analysed the correlation between WalkSAT's performance on the different encodings of the individual instances, using the same methodology as in Chapter 4 and the parameter settings described above. Figure 7.4 shows the correlation results as a scatter plot; as can be easily seen, there is a strong linear correlation between the logarithm of WalkSAT's mean local search cost on the differently encoded test-sets. This indicates that generally, the algorithm's behaviour depends more on properties of the original problem instance than on features induced by the encoding. However, the correlation shows some noise, especially for instances with high local search cost, which means that independent from instance hardness, the encoding has an additional influence on SLS performance. But since the correlation is very strong (correlation coefficient ≈ 0.95), this influence seems to be rather limited.

As we have seen from our analysis of the hardness distributions, w.r.t. SLS performance, the compact encoding seems to be generally inferior compared to the sparse encoding. Performing regression analyses (as introduced in Chapter 4) for the performance correlation in Figure 7.4 shows additionally that the performance difference slightly decreases as the instances get harder. Thus, the compact encoding is particularly inferior for relatively easy problem instances, while for intrinsically hard instances, the influence of the encoding on SLS performance is somewhat weaker.

To make sure that WalkSAT behaves regularly on both test-sets, we analysed the RLDs for the easy, medium, and hard instance (determined as described in Chapter 4) in both encodings. As expected, the results show that for RLDs, best-fit *ged* approximations (*cf.*

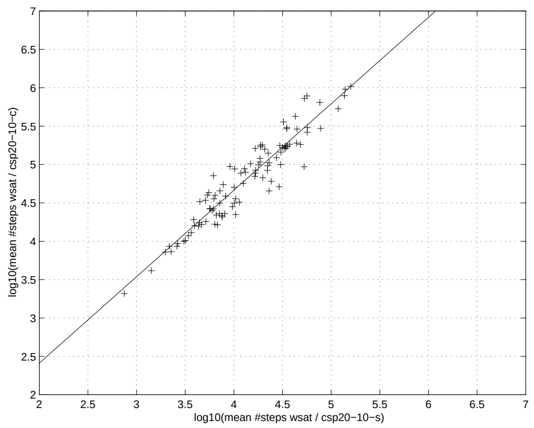

Figure 7.4: Correlation of average local search cost per instance between sparse (horizontal) and compact SAT-encoding (vertical) for CSP instances from test-set `csp20-10` when using WalkSAT with approx. optimal noise.

Chapter 5) pass a standard χ^2 test for the $\alpha = 0.05$ acceptance level. Interestingly, when using standard exponential distributions (which do not model the initial search phase), the approximations for the medium and hard instances for both encodings still pass the test. This indicates that these instances are relatively hard, such that the initial search phase has no significant influence on WalkSAT's behaviour.

Comparison of SLS Performance

Next, we compared the performance of different SLS algorithms on the sparsely encoded test-set `csp20-10-s`. We applied the same methodology as in Chapter 4, Section 4.5, when we evaluated SLS algorithms on Random-3-SAT test-sets. In particular, for each SLS algorithm we measured the distribution of the average local search cost across the test-set, using approximately optimal noise settings and sufficiently high maxSteps settings to guarantee maximal success rates in all cases. The resulting hardness distributions are shown in Figure 7.5. Obviously, for the major part of the test-set, the following performance ordering can be observed: Novelty shows almost identical performance as GSAT+tabu, followed closely by R-Novelty and WalkSAT+tabu; next comes WalkSAT, which performs significantly worse, and finally GWSAT. Not very surprisingly, given their essential incompleteness, Novelty, R-Novelty, and WalkSAT+tabu suffer from stagnation behaviour on 3–7% of the problem instances. Interestingly, for GSAT+tabu, although most probably essentially incomplete as well, such stagnation behaviour could not be observed on the given test-set.

In a next step, we analysed the correlation between the different algorithm's average performance across the test-set. As for Random-3-SAT (Chapter 4, Section 4.5), we chose

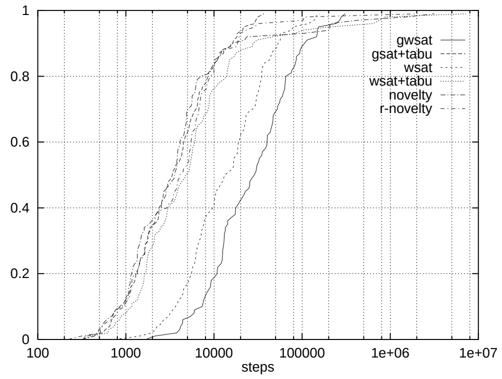

Figure 7.5: Hardness distributions (average local search cost per instance) for different SLS algorithms, when applied to test-set `csp20-10-s` (using approx. optimal noise).

encodings	r	a	b	o
gwsat *vs* gsat+tabu	0.9845	0.9471	**-0.6609**	0
gwsat *vs* wsat	0.9874	0.9587	-0.1108	0
gwsat *vs* wsat+tabu	0.9676	0.9047	-0.4128	6
gwsat *vs* novelty	0.9809	**0.8571**	-0.3775	7
gwsat *vs* r-novelty	0.9567	0.9105	-0.4939	3

Table 7.3: Test-set `csp20-10-s`, pairwise hardness correlation for various algorithms with approx. optimal noise, based on 100 tries / instance; r is the correlation coefficient, a and b are the parameters of the lms regression analysis, and o the number of outliers which have been eliminated before the analysis (see text).

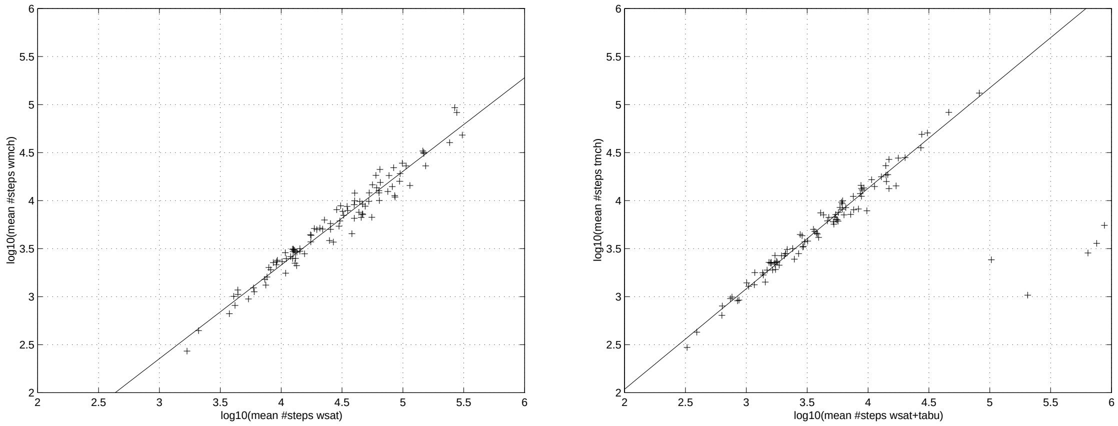

Figure 7.6: Correlation of average local search cost per instance for WalkSAT *vs* WMCH *(left)* and WalkSAT+tabu *vs* TMCH *(right)* applied to test-set `csp20-10`; all algorithms use approx. optimal noise, the SAT algorithms use the sparse encoding.

encodings	r	a	b
wsat *vs* wmch	0.9793	0.9752	-0.5715
wsat+tabu *vs* tmch	0.9894	1.0458	-0.0556
novelty *vs* tabu-gh	0.9614	1.0153	-0.3615

Table 7.4: Test-set `csp20-10`, pairwise hardness correlation for CSP and SAT algorithms (applied to sparse encoding) with approx. optimal noise, based on 100 tries / instance; r is the correlation coefficient, a and b are the parameters of the *lms* regression analysis, outliers have been eliminated before the analysis (see text).

GWSAT as a reference algorithm. The results of the correlation analysis are reported in Table 7.3. When ignoring the cases, where stagnation behaviour occurred, there is generally a strong correlation between the algorithms' average local search cost across the test-set, indicating that the hardness of instances w.r.t. to the given set of algorithms is an intrinsic property of the instances. The data from the regression analysis (a and b values in Table 7.3) indicate that Novelty scales best with instance hardness, followed by WalkSAT+tabu and R-Novelty, WalkSAT and GSAT+tabu, and finally, GWSAT. The interesting observation here is that, analogously to the results on Random-3-SAT instances (*cf.* Table 4.11, page 97), GSAT+tabu shows very competitive average performance but scales worse with instance hardness than the WalkSAT variants.

SLS Algorithms for SAT *vs* CSP

Given the close conceptual relation between SLS algorithms for SAT and CSP, directly comparing the performance of the corresponding algorithms is interesting in the light of

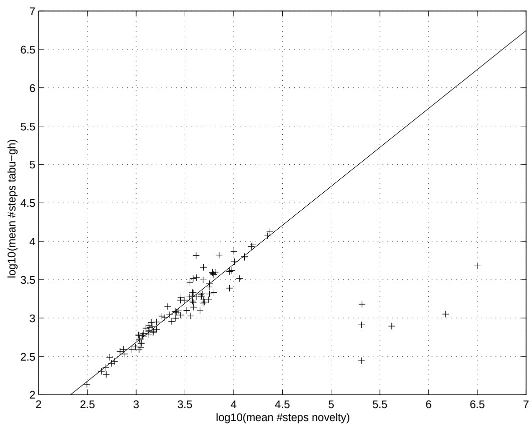

Figure 7.7: Correlation of average local search cost per instance for Novelty *vs* Galinier's and Hao's tabu search algorithm for CSP applied to test-set `csp20-10`; all algorithms use approx. optimal noise, Novelty uses the sparse encoding.

investigating the effectivity of the generic problem solving approach discussed in Chapter 1. Specifically, we first compare the performance of WalkSAT and WMCH as well as WalkSAT+tabu and TMCH, where the CSP algorithms work directly on the instances of test-set `csp20-10`, while the SAT algorithms are applied to the sparsely encoded problem instances (test-set `csp20-10-s`). The TMCH and WMCH implementations used for our empirical study were kindly provided by Thomas Stützle (TU Darmstadt, Germany).

Our performance comparison is based on correlation analyses, using the same methodology as in the previous section. The results of these correlation analyses are graphically depicted in Figure 7.6 and summarised in Table 7.4. Clearly, there is a very strong correlation between the performance of the SAT algorithms and the corresponding CSP algorithms across the test-set. While WMCH is slightly better than WalkSAT (both using approx. optimal noise), the performance difference between TMCH and WalkSAT+tabu is practically negligible, except for the fact that WalkSAT+tabu suffers from stagnation behaviour for 6 instances, while TCMH shows no such outliers. However, as argued in Chapter 5, this effect of WalkSAT+tabu's essential incompleteness should be easy to overcome by slightly modifying the algorithm.

The results from the corresponding regression analyses (*cf.* Table 7.4) indicate that while WMCH scales slightly better than WalkSAT with instance hardness, for TMCH and WalkSAT+tabu, the situation is reversed: here, WalkSAT+tabu has a slight scaling advantage, as can be seen from the fact that the slope of the regression line (a parameter) is greater than one.

Finally, we apply the same analysis to the best-performing SLS-based SAT algorithm and the best SLS-based CSP algorithm we are aware of — the tabu search algorithm by Galinier and Hao [GH97]. Again, using optimal noise parameters for both algorithms, we find that, measuring the average number of local search steps per solution, Galinier's and Hao's CSP algorithm has an advantage of a factor between 2 and 4 over Novelty (ignoring the outliers, see below). The correlation between both algorithm's performance is very strong (*cf.* Figure 7.6, Table 7.4); however, like WalkSAT+tabu, Novelty suffers from stagnation behaviour (on 7 instances), while the CSP algorithm shows no such behaviour. Interestingly, 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, or TMCH and WalkSAT+tabu.

Generally, the results from comparing the performance of SLS-based algorithms for CSP and SAT on the test-set of problem instances indicate that when measuring the average number of steps per solution, the differences are surprisingly small. We deliberately refrained from comparing CPU-times for these algorithms, because the implementations for the SAT algorithms are significantly more optimised. Furthermore, while the SAT algorithms can be applied to arbitrary CNF formulae, the implementations of the CSP algorithms are restricted to binary CSPs — which severely limits the range of their application. 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 specialised SLS-based CSP solvers.

7.2.5 Search Space Structure

After investigating the influence of the encoding strategy on SLS performance, we now analyse the search space structure using the methods from Chapter 6. In particular, we measure the number of solutions, the standard deviation of the objective function, and local minima branching for both encodings of the easy, medium, and hard problem instance from the `csp20-10` test-set using the same methodology as in Chapter 6. Only here, because the formulae generated by the different encodings have different numbers of clauses and variables, these measures have to be normalised to make them comparable. Specifically, the normalised *sdnclu* value is defined as the standard deviation of the objective function after it has been scaled to the interval $[0, 1]$, while for a LMIN state, the normalised *blmin* value measures the fraction of neighbouring states with the same objective function value.

Table 7.5 summarises the results from the analyses of the various search space features. Obviously, the number of solutions is identical for both encodings (this is guaranteed by the definition of the encodings). Since at the same time, the compact encoding achieves a significant reduction of search space size, we observe a vast difference in solution density between the sparse and compact encoding. Given the dominant role of this factor on SLS performance established earlier (*cf.* Chapter 6), it is rather surprising that WalkSAT's performance is so much worse for the compact encoding. However, comparing the normalised *sdnclu* and average *blmin* values between the two encodings, we see that for the compact

instance	solutions	solution density	norm. *sdnclu*	norm. *blmin*	avg *lsc*
csp20-10-s/easy	37,297	$2.01 \cdot 10^{-56}$	0.03655	0.015	775.80
csp20-10-s/medium	6	$3.73 \cdot 10^{-60}$	0.03606	0.017	11,162.69
csp20-10-s/hard	2	$1.24 \cdot 10^{-60}$	0.03597	0.017	134,777.38
csp20-10-c/easy	37,297	$3.09 \cdot 10^{-20}$	0.00114	0.131	2,249.19
csp20-10-c/medium	6	$4.96 \cdot 10^{-24}$	0.00111	0.157	77,883.52
csp20-10-c/hard	2	$1.65 \cdot 10^{-24}$	0.00113	0.161	1,000,456.82

Table 7.5: Easy, medium, and hard instance from test-set `csp20-10`, number of solutions, solution density, normalised *sdnclu* and average *blmin* values, as well as average local search cost (w.r.t. WalkSAT, using approx. optimal noise, 1,000 tries) for easy, medium, hard instance using different encodings.

encoding, the standard deviation of the objective function is much lower while the local minima are significantly more branched than for the sparse encoding. Intuitively, as argued in Chapter 6, this explains why WalkSAT has significantly more difficulty finding solutions in the search spaces induced by the compact encoding.

Regarding the differences in local search cost between the easy, medium, and hard problem instances when the same encoding is used, these are most probably mainly caused by the differences in solution density, as both, *sdnclu* and average *blmin* values, are relatively similar. This shows that while for a given encoding, the number of solutions is the dominant factor regarding local search cost, the different features induced by the encodings (some of which are measured by the *sdnclu* and average *blmin* values) have a drastically stronger influence on SLS performance.

7.3 Hamilton Circuit Problems

The *Hamilton Circuit Problem (HCP)* is a well-known combinatorial decision problem from graph theory. It is based on the notion of a *Hamilton Circuit* (sometimes also called *Hamilton Cycle*), *i.e.*, a cyclic path in a given graph G which visits evey vertex of G exactly once. For a given graph G, the Hamilton Circuit Problem is to decide whether G contains at least one Hamilton Circuit. The HCP is closely related to the Traveling Salesperson Problem (TSP), which can be regarded as a generalisation of the HCP where additionally the graph is weighted and the goal (in the optimisation variant) is to find a Hamilton Circuit with minimal weight, *i.e.*, a shortest round trip. Like the TSP, the HCP can be formulated for directed and undirected graphs and is $\mathcal{NP}$-complete in both cases. In this section we study the differences in SLS performance and search space structure for different SAT-encodings of directed HCP.

7.3.1 SAT Encodings of the HCP

The fundamental idea for encoding HCP into SAT is based on the observation that for a given graph $G = (V, E)$, each Hamilton Circuit (HC) corresponds to a cyclic permutation of G's vertex set V. However, a cyclic vertex permutation π_V corresponds only to a HC, if additionally for each pair of successive elements of π_V, there is an edge in E connecting the corresponding vertices. This can be formalised as a CSP the variables of which represent the positions of the permutation and their values determine the vertices appearing at these positions:

For a given directed graph $G = (V, E)$, $HCP_p(G)$ is represented by the CSP $(X, \mathcal{D}, \mathcal{C})$, where $X = \{p_0, \ldots, p_{\#V-1}\}$, for all i, $\mathcal{D}_{p_i} = V$, and $\mathcal{C}$ consists of the following constraints:[2]

(1) $p_i = x \wedge p_k = y \iff x \neq y \vee i = k \quad (i, k \in \mathbb{Z}_{\#V})$
(2a) $p_{i-1} = x \wedge p_i = y \iff (x, y) \in E \quad (i \in \mathbb{Z}_{\#V} - \{0\})$
(2b) $p_{\#V-1} = x \wedge p_0 = y \iff (x, y) \in E$

This can be transformed into SAT using the sparse or compact SAT-encodings for CSP from Section 7.2. We refer to this encoding as the *position-based encoding*, since in the CSP formulation we assign vertices to permutation positions. However, there is a dual representation, based on the idea of representing the permutations by assigning permutation positions to vertices rather than vertices to permutation positions. The corresponding CSP formalisation of this *vertex-based encoding* is very similar to the one given above:

For a given directed graph $G = (V, E)$ with $V = \{0, \ldots, n-1\}$, $HCP_v(G)$ is represented by the CSP $(X, \mathcal{D}, \mathcal{C})$, where $X = \{v_0, \ldots, v_{n-1}\}$, for all i, $\mathcal{D}_{v_i} = \mathbb{Z}_n$, and $\mathcal{C}$ consists of the following constraints:

(1) $v_i = x \wedge v_k = y \iff x \neq y \vee i = k \quad (i, k \in \mathbb{Z}_n)$
(2a) $v_i = x - 1 \wedge v_k = x \iff x \geq 1 \wedge (i, k) \in E \quad (i, k \in \mathbb{Z}_n)$
(2b) $v_i = n - 1 \wedge v_k = 0 \iff (i, k) \in E \quad (i, k \in \mathbb{Z}_n)$

As for the position-based encoding, this CSP representation can be easily translated into SAT using one of the CSP encodings discussed before. Thus, we obtain four different SAT-encodings for the Hamilton Circuit Problem.

When re-examining these CSP-encodings, one might notice that they contain a certain redundancy. For any Hamilton Circuit, it is irrelevant which vertex is visited first, while the permutation-based encodings differentiate between solutions with different starting points. Thus, for each Hamilton Circuit in a given graph with n vertices, the corresponding CSP instance has n symmetrical solutions. This symmetry can be exploited to reduce the problem

[2] For clarity's sake, we are using a different notation here than in Definition 1.6; however, both are equivalent and can be easily transformed into each other.

test-set	instances	vertices	edges	d	vars	clauses
hc10-p	100	10	15	3	100	1,060
hc10-v	100	10	15	3	100	1.060
hc10-vc	100	10	15	3	40	1,110
hc10-pe	100	10	15	3	81	≈ 733
hc10-ve	100	10	15	3	81	≈ 733
hc10-vce	100	10	15	3	36	≈ 781

Table 7.6: Random HCP test-sets, using different SAT-encodings; v = vertex-based, p = permutation based, c = compact (if not specified, encoding is sparse), e = symmetry eliminated (only, if specified); d is the mean vertex degree.

size of the CSP (and SAT) instances generated by our two encoding schemes. To achieve this *symmetry elimination*, we just fix an arbitrary element of the permutation to an arbitrary vertex; this eliminates one CSP variable for both encodings and reduces the size of all the remaining variables' domains by one.

For a graph with n vertices, we thus reduce the number of propositional variables from n^2 to $(n-1)^2$ when using the sparse encoding, and from $n\lceil\log_2 n\rceil$ to $(n-1)\lceil\log_2(n-1)\rceil$ when using the compact encoding. The number of clauses also shrinks because for the fixed element of the permutation, the number of constraints of type (2a) and (2b) is reduced. At the same time, the number of solutions is reduced by a factor of n. Consequently, for the sparse encoding, the net result of symmetry elimination is a reduction of the search space size by a factor of 2^{2n-1} ($= 2^{n^2} / 2^{(n-1)^2}$ = original search space size / reduced search space size) and an *increase* in solution density (solutions / search space states) by a factor of $2^{2n-1}/n$. For the compact encoding, the search space shrinks only by a factor of ca. 2, while the solution density *decreases* by a factor of ca. $n/2$. Considering the strong correlation between the number of solutions and local search cost observed in Chapter 6, one might expect that symmetry elimination reduces search cost for the sparse encoding, while for the compact encoding, savings are less likely.

7.3.2 Benchmark Instances

For empirically investigating SLS performance for different SAT-encodings of the HCP, we focussed on the Hamilton Circuit Problem in random graphs. 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 [CKT91]. 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 [FM95]. Based on these results, we created a test-set by randomly sampling soluble HCP instances from distributions of random graphs with $n = 10$ and $\kappa = 1$.

The test-set was generated using a HCP generator and solver developed and provided by Joe

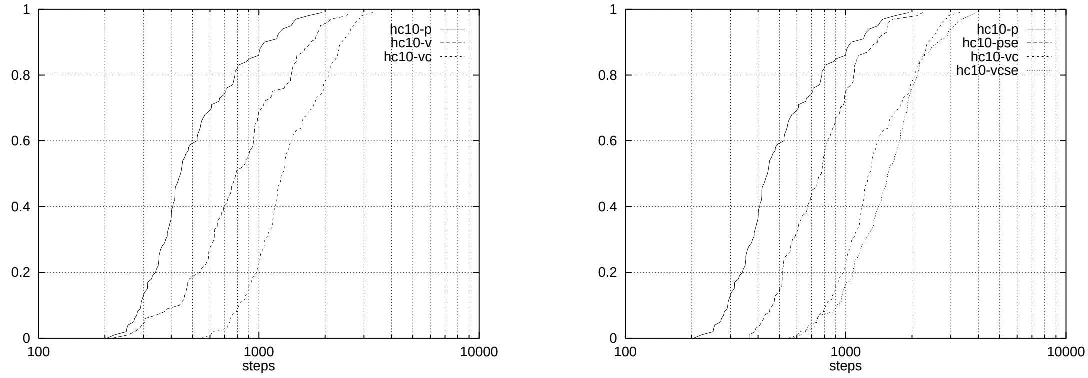

Figure 7.8: Hardness distributions (average local search cost per instance) for WalkSAT (using approx. optimal noise), when applied to different encodings of test-set `hc10`.

Culberson; insoluble instances were filtered out using the integrated systematic HCP solver, such that the test-set contains only soluble instances. This test-set (named `hc10`) was then encoded into SAT using an encoder implemented in the course of this work. This encoder allows to generate position- and vertex-based as well as sparse and compact encodings with and without symmetry elimination. For the study conducted here, we generated the test-sets described in Table 7.6; each test-set comprises 100 instances. Note that for the encodings using symmetry elimination, the number of clauses varies depending on the actual in- and out-degree of the vertex which is clamped to a fixed position in the permutation; in Table 7.6, we therefore report the average number of clauses across the test-set.

To reduce computational and overall experimental expenses, for some of our experiments we only used the minimum, median, and maximum elements (denoted as `easy`, `medium`, and `hard` instance) from the hardness distributions for WalkSAT with approx. optimal noise setting, using the position-based, sparse encoding. As we will see later, the hardness (here: average local search cost for WalkSAT with approx. optimal walk) is rather tightly correlated for the different encodings.

7.3.3 SLS Performance

In a first series of experiments, we studied the distribution of instance hardness (average local search cost) for WalkSAT over different encoding of the ten-vertex test-set (`hc10-`$*$). Since we are mainly interested in optimal performance here, we used approximately optimal noise settings for each of the encodings. These were determined based on measuring average local search cost for a particular, arbitrarily chosen instance from the original HCP test-set. Interestingly, we observed significant differences in the optimal noise for the various encodings; generally, the optimal noise was significantly higher for the compact encodings as well as for those using symmetry elimination.

Figure 7.8 shows these hardness distributions; the corresponding basic descriptive statistics

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{25}	Q_{75}	Q_{10}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
p	572.56	339.75	0.59	435.21	351.94	694.32	289.45	1034.35	1.97	3.57
v	931.11	504.98	0.54	781.28	589.54	1148.76	386.17	1680.76	1.95	4.35
vc	1463.52	606.03	0.41	1258.44	1013.17	1904.27	800.31	2324.88	1.88	2.90
pe	826.46	364.63	0.44	753.26	524.86	989.15	452.85	1318.97	1.88	2.91
ve	935.02	433.64	0.46	883.92	616.22	1076.89	514.13	1496.82	1.75	2.91
vce	1648.54	680.20	0.41	1545.27	1113.37	1960.47	896.29	2514.87	1.76	2.81

Table 7.7: Test-set `hc10`, basic descriptive statistics of hardness distributions for different SAT-encodings, using WalkSAT (approx. optimal noise, 100 tries/instance).

are reported in Table 7.7. As we can see, the position-based, sparse encoding (`p`) is most efficient w.r.t. local search cost, followed by the vertex-based, sparse encoding (`v`); the vertex-based compact encoding (`vc`) induces significantly higher local search cost. Also, symmetry elimination generally leads to significantly increased local search cost (compare `p` *vs* `pse`, *etc.*). However, this increase is strongest for the sparse position-based encoding, and relatively moderate for the vertex-based compact encoding. This is quite plausible, since in the latter case, SLS behaviour is apparently already significantly hindered by the basic encoding, such that the additional impediments induced by symmetry elimination are less relevant to SLS behaviour. Note that the observed variation in instance hardness across the test-sets is significantly smaller than for Random-3-SAT or Random Binary CSP (*cf.* Tables 4.10, page 96 and 7.2, page 183).

Another interesting observation from the data presented in Table 7.7 is the fact that the relative variability of instance hardness across the test-set is negatively correlated with the efficiency of the encoding w.r.t. local search cost. In particular, the normalised standard deviation (stddev/mean) and the percentile rations are lower for the compact encodings than for the sparse encodings; also, these values are lower when symmetry elimination is applied. This suggests that for relatively easy problem instances, the influence of the encoding on SLS performance is much higher than for instances which are intrinsically harder — a fairly plausible assumption.

In a next step, we perform the same correlation analysis as for the Random-3-SAT and Graph Colouring test-sets in Chapter 4; only now, we don't compare different algorithms, but rather the same algorithm (WalkSAT with approx. optimal noise) applied to different encodings of the same set of HCP instances. The results of this analysis are shown in Table 7.8. As in the case of CSP (*cf.* Section 7.2), we observe a relatively strong correlation between the local search cost for different encodings of the same instance, indicating that features of the original instance, rather than those induced by the encoding, dominate SLS behaviour. Interestingly, the influence of symmetry elimination seems to be stronger for sparse than for compact encodings, as can be seen from the lower correlation coefficients for the corresponding hardness correlations (the higher noise in these correlations is caused by the encoding).

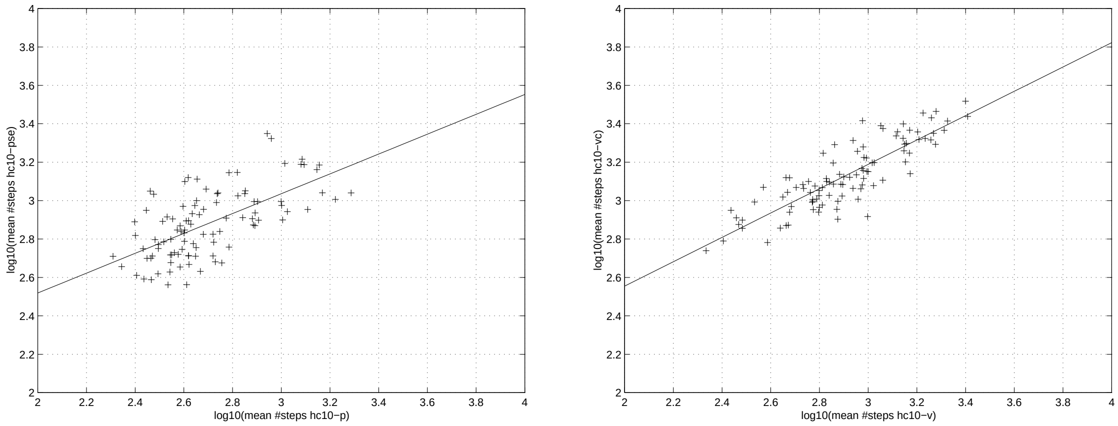

Figure 7.9: *Left:* Correlation between average local search cost for WalkSAT for test-sets `hc10-p` and `hc10-pse`, using approx. optimal noise. *Right:* Same for `hc10-v` *vs* `hc10-vc`.

encodings	r	a	b
hc10p *vs* hc10v	0.74	0.81	0.72
hc10v *vs* hc10vc	0.85	0.63	1.29
hc10p *vs* hc10pe	0.63	0.52	1.49
hc10v *vs* hc10ve	0.59	0.44	1.64
hc10vc *vs* hc10vce	0.79	0.80	0.67

Table 7.8: Test-set `hc10`, pairwise hardness correlation for WalkSAT with approx. optimal noise applied to different encodings, based on 100 tries / instance; r is the correlation coefficient, a and b are the parameters of the lms regression analysis.

Furthermore, the results from the regression analyses (a and b values in Table 7.8, *cf.* Chapter 4) show that the more efficient encodings (w.r.t. resulting local search cost for WalkSAT) scale worse with instance hardness (compare a values for p, v, and vc) — confirming our earlier observation for CSP that the influence of the encoding decreases with instance hardness. When applying symmetry elimination, this effect can also be observed: while generally, the local search cost tends to increase, this increase is stronger for relatively easy than for harder instances.

We also analysed the individual RLDs for different encodings of the medium hardness problem instance, using *ed* and *ged* approximations as described in Chapter 5. Since regardless of the encoding, this instance is rather easy to solve for WalkSAT, it is not surprising that the approximations using exponential distributions did not pass a standard χ^2 test (except for the vertex-based, compact encoding) because of the effect of the initial search phase on SLS behaviour for short run-times. But when using generalised exponential distributions which take into account the initial search phase (*cf.* Chapter 5), all RLD approximations pass the χ^2 test for the $\alpha = 0.05$ confidence level. This indicates that for HCP, independently from the encoding chosen, WalkSAT shows the typical behaviour as characterised in Chapter 5 for a number of different problem domains.

Generally, our results indicate that, as for CSP, sparse encodings are significantly superior w.r.t. SLS performance than compact encodings. Also, SLS performance is generally rather impeded than improved by symmetry elimination. Finally, there is a significant performance advantage for the vertex-based encoding over the position-based encoding.

7.3.4 Search Space Structure

After analysing the influence of the different encodings on local search cost, in this section, we investigate the role of some search space features (*cf.* Chapter 6) in this context. As we have seen before, SLS performance strongly depends on the number of solutions, but is also affected by other factors such as the standard deviation of the objective function and the average local minima branching.

Obviously, the number of solutions of a given HCP instance is identical for the position-based and vertex-based encodings, and it is not affected by using compact instead of sparse encodings. Symmetry elimination, however, reduces the number of solutions by a factor n (where n is the number of vertices in the given graph), as can be easily seen from the description of the encodings. At the same time, the search space is much smaller when symmetry elimination has been applied. As detailed in Section 7.3.1, this results in a net increase in the average solution density for the sparse encodings, while for the compact encoding, the net solution density descreases.

Table 7.9 shows the number of solutions and solution density for the easy, medium, and hard instances from the `hcp10` test-set for vertex-based, sparse encodings with and without symmetry elimination as well as the corresponding average local search cost for WalkSAT. Clearly, despite the higher solution density, the average local search cost increases when

instance	symmetry elim.	solutions	solution density	avg. *lsc*
hc10-p/easy	no	60	$4.73 \cdot 10^{-29}$	174.45
hc10-pse/easy	yes	6	$2.48 \cdot 10^{-24}$	529.33
hc10-p/medium	no	20	$1.58 \cdot 10^{-29}$	361.78
hc10-pse/medium	yes	2	$8.27 \cdot 10^{-25}$	740.10
hc10-p/hard	no	20	$1.58 \cdot 10^{-29}$	917.40
hc10-pse/hard	yes	2	$8.27 \cdot 10^{-25}$	935.06

Table 7.9: Test-sets `hc10-p` and `hc10-pse`; number of solutions, solution density and average local search cost for WalkSAT (using approx. optimal noise, based on 1,000 tries / instance) fo easy, medium, and hard problem instance.

instance	norm. *sdnclu*	norm. *blmin*	avg. *lsc*
hc10-p/medium	0.0503	0.057	361.78
hc10-v/medium	0.0502	0.056	476.30
hc10-vc/medium	0.0017	0.181	1,255.02
hc10-pe/medium	0.0561	0.063	740.10
hc10-ve/medium	0.0561	0.063	858.06
hc10-vce/medium	0.0021	0.196	1,776.35

Table 7.10: Medium hardness instance from test-set `hc10`, normalised sdnclu and blmin measures and average local search cost (w.r.t. WalkSAT, using approx. optimal noise, $1,000$ tries) for different encodings.

using symmetry elimination — a result, which is not consistent with our intuition about the dominant influence of the solution density on SLS performance. However, as observed before (*cf.* Section 7.3.3), the performance loss is most drastic for the easy instance, while for the hard instance the detrimental effect of symmetry elimination is very small. Similar observations can be made for the compact encodings; there, however, because symmetry elimination decreases the solution density, the increase in local search cost matches our intuition.

Next, we analysed the two other measures of search space structure discussed in Chapter 6: the standard deviation of the objective function (*sdnclu*) and the average local minima branching (*blmin*). As in Section 7.2, we normalise these measures to enhance their comparability between the different encodings. Table 7.10 shows the normalised *sdnclu* and average *blmin* values, as well as the average local search cost (w.r.t. WalkSAT, using approx. optimal noise) for the medium instance of test-set `hc10` under all six encodings discussed before. The results confirm our original intuition regarding the influence of these search space features on SLS performance: the local search cost seems to be negatively correlated to the normalised sdnclu value, and positively correlated to the normalised average local minima branching. In the analysis of SAT-encodings for the given HCP instance, this can be clearly seen from the results reported in Table 7.10, when comparing the corresponding values between the sparse and compact encodings, as well as between encodings with and without symmetry elimination. In the latter case, the effect of the increased local minima branching seems to outweigh a possible beneficial effect of a slightly increased *sndclu* value. Applying the same analysis to the easy and hard problem instances from the same test-set confirms these results.

In the light of these results, WalkSAT's performance differences on the various encodings can be explained in the following way: As for CSP, compact encodings induce search spaces characterised by extremely flat and feature-less objective functions and relatively highly branched local minima regions — features, which impede stochastic local search. A similar effect, but considerably weaker, is generally induced by symmetry elimination. This demonstrates that analysing relatively simple features of the search space can be useful for assessing different strategies for SAT-encoding instances from other problem domains, such as HCP.

7.4 Related Work

SAT-encodings of combinatorial problems are frequently used in the context of $\mathcal{NP}$-completeness proofs in complexity theory [GJ79]. A systematic approach to transforming hard combinatorial problems into SAT has been proposed by Stamm-Wilbrandt [SW93], who also gives space-efficient SAT-encodings for many $\mathcal{NP}$-complete problems from different domains. Iwama and Miyzaki introduced the concept of SAT-variable complexity and discussed different encodings of hard combinatorial problems, including the Hamilton Circuit Problem, which are optimised for a minimal number of propositional variables [IM94].

In contrast, the influence of different encoding strategies on the performance of SAT solvers is mainly unexplored. Although from the beginning of the development of modern SLS algorithms for SAT, these have been tested on SAT-encoded problems from domains like Graph Colouring or Blocks World Planning [SLM92, KS92], the influence of the encodings, to our best knowledge, has rarely been systematically investigated. Only recently, when it became clear that hard Blocks World Planning instances can be effectively solved using SLS algorithms for SAT, the issue of the influence of SAT-encodings on SLS performance has been further studied. Kautz and Selman use various techniques for optimising SAT-encodings for various well-known planning domains (including Blocks World) to achieve maximal SLS performance [KS96, KMS96]. The first systematic study of SAT-encodings and their impact on the average solution cost for state-of-the-art SAT algorithms we are aware of, is the study on automatic SAT-compilation for planning problems by Ernst, Millstein, and Weld [EMW97]. Different from [KS96], who use hand-crafted encodings, [EMW97] systematically evaluate a number of general encoding strategies which have been built into a fully automated SAT-encoder for planning problems from various domains. One of their results states that compact encodings of planning instances are typically very difficult for SLS-based SAT algorithms — an observation which is confirmed by our results for CSP and HCP.

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 [Smi94, SD96, Pro96]. The sparse SAT-encoding of CSP has been introduced by de Kleer [dK89], who used it in the context of a comparative study of CSP and assumption-based truth maintainance system (ATMS) concepts and techniques. The compact CSP encoding is inspired by Iwama's and Miyazaki's work on SAT-variable complexity (see also below). Regarding SLS algorithms for CSP, one of the first and most influential algorithms is the Min-Conflicts Heuristic (MCH), which has been introduced by Minton *et al.* in 1992 [MJPL92]. Recently, the basic algorithm, which is conceptually closely related to WalkSAT [SKC94], has been extended with random walk and tabu search techniques by Stützle *et al.* [SSS97]. Analogous to the corresponding SAT algorithms, these extensions show a superior performance compared to the original MCH. The best currently known algorithm for the subclass of CSP studied here is the tabu search algorithm by Galinier and Hao [GH97], which is conceptually closely related to GSAT, as it is based on an analogous neighbourhood relation. Despite the close relationship between SLS algorithms for SAT and CSP, we are not aware of any previous comparative analysis of their performance on an identical set of problem instances. The problem instances, as well as the implementations of the CSP algorithms used in the context of this work have been kindly provided by Thomas Stützle (TU Darmstadt, Germany).

The Hamilton Circuit Problem in random graphs has been studied by Cheeseman *et al.* [CKT91] as well as by Frank and Martel [FM95]. Both studies focus on phase transition phenomena and their influence on the performance of complete HCP algorithms. Although the solubility phase transition does not as nicely coincide with the peak in difficulty as for other problem domains, such as Random-3-SAT, we used the results from [FM95] for generating the test-sets used in the context of this work. The actual test-sets were created using an HCP instance generator kindly provided by Joseph Culberson (University of Alberta, Canada). SAT-encodings of HCP were previously discussed in [IM94], who developed the compact, vertex-based SAT-encoding with symmetry elimination in the context of minimising the number of propositional variables required for translating hard combinatorial problems to SAT. In [Hoo96c, Hoo96b], we analysed and characterised the performance of GSAT and GWSAT on sets of randomly generated HCP instances using the compact, vertex-based encoding with symmetry elimination, showing that these problem instances are exceptionally difficult for both algorithms. However, in [Hoo96b], we also developed an HCP-specific SLS algorithm (GHC) which is conceptionally closely related to the Min-Conflicts Heuristic as well as to GWSAT and shows significantly improved performance on the given test-sets.

7.5 Conclusions

In this chapter, we studied the impact of encoding strategies on search space structure and SLS performance. Based on two case studies, covering Constraint Satisfaction and Hamilton Circuit Problems, we exemplified how some of the methods developed in the previous chapters can be used to investigate how different SAT encodings induce certain search space features which, in turn, affect the behaviour of SLS algorithms.

At the beginning of this chapter, in Section 7.1, we formulated three questions regarding the effectivity of specific encoding strategies and the competitiveness of the generic problem solving approach in general. Based on the results of our case studies for CSP and HCP, we will give some tentative answers here — however, as the underlying empirical evidence is very limited in its scope, these answers should be rather understood as testable hypotheses.

Does it pay off to minimise the number of propositional variables, i.e., are compact encodings which achieve small search spaces preferable to sparse encodings?

No. According to our results for CSP and HCP, while compact encodings have a significantly higher solution density, they induce search spaces which are rather flat and feature-less (characterised by low *sdnclu* values). At the same time, their local minima are significantly more branched, which makes escaping from these minima considerably more difficult for SLS algorithms like WalkSAT. In contrast, sparse encodings induce rugged search spaces with lowly branched local minima — features which generally increase the performance of SLS algorithms. So, in a nutshell, it seems to be much more advisable to use sparse rather than compact encodings, as the former tend to induce vast, but well-structured search spaces which are much easier to search than the search spaces produced by compact encodings, which are much smaller, but provide far less guidance for local search.

Is it advisable to reduce the number of propositional variables required for a given encoding by eliminating symmetric solutions from the original problem formulation?

Apparently not. Although, for the given example (HCP), when using sparse encodings, symmetry elimination increases the solution density, it seems to have detrimental effects on SLS performance which are similar to those induced by compact encodings, although the impact on SLS performance is less drastic. However, since we did not study the scaling of these effects with problem size, it cannot be ruled out that, for bigger instances, a different situation can be found. Preliminary experimentation indicates that for the problem class studied here, the detrimental effect of symmetry elimination can be also observed for larger problem sizes.

Is the generic problem solving approach competitive with applying SLS algorithms to the un-encoded problem instances?

Our results for Random Binary CSP instances indicate that this is the case. Although, when comparing the number of steps per solution for state-of-the-art algorithms, we observed a slight advantage for the direct CSP approach, this is easily outweighed by other advantages of the generic problem solving approach, such as the availability of extremely efficient implementations and its more general applicability. However, again it is not clear how the observed performance difference scales with problem size. Nevertheless, our result is good news regarding the generic problem solving approach based on efficient SAT-encodings and modern SLS algorithms for SAT, especially, since the same, simple encoding strategies and SLS algorithms could be successfully used to competitively solve a number of hard combinatorial problems from different domains.