Chapter 4

Phase Transitions in *Regular Random 3-SAT*

4.1 Introduction

Phase transitions occur in *SAT*, as well as in other NP-hard problems (Cheeseman et al., 1991). Mitchell et al. (1992) report results from experiments on testing the satisfiability of classical *Random 3-SAT* instances with the Davis-Putnam (DP) procedure (Davis et al., 1962). They observed that (i) there is a sharp phase transition from satisfiable to unsatisfiable instances for a certain value of the ratio of the number of clauses to the number of variables. At lower ratios, most of the instances are under-constrained and are thus satisfiable. At higher ratios, most of the instances are over-constrained and are thus unsatisfiable. The value of that ratio where 50% of the instances are expected to be satisfiable is referred to as the threshold; and (ii) there is an easy-hard-easy pattern in the computational difficulty of solving problem instances as that ratio is varied; the hard instances occur in the area near the threshold. Nowadays, there is strong experimental evidence that the value of the threshold is around 4.25, but there are no analytical results. Only lower and upper bounds on the location of the threshold are known (Kirousis et al., 1996).

In this chapter we first provide experimental evidence of the existence of phase transitions in *Regular Random 3-SAT* when the satisfiability of the instances is determined with Regular-DP. Then, we report a series of experiments that indicate that the location of the threshold increases logarithmically in the cardinality of the truth value set. We also provide a theoretical explanation of this fact by providing upper bounds on the location of the unsatisfiability threshold.

Our original interest in this problem was motivated by the search of challenging benchmarks for analysing and comparing algorithms for *Regular-SAT*. The experimental results we report here suggest that it makes sense to evaluate satisfiability algorithms for regular CNF formulas on instances of the hard region of the phase transition. We designed and implemented a generator of *Regular Random 3-SAT* instances which is able to provide a large number of such hard instances easily. We used this generator to compare the performance of Regular-DP/RH-1, Regular-DP/RH-2, Regular-GSAT/Basic, and Regular-WalkSAT/Basic. The experimental results obtained are reported in the last section, and can be summarized as follows: Regular-DP/RH-2 outperforms Regular-DP/RH-1 on instances of the hard region of the phase transition, Regular-GSAT/Basic outperforms Regular-DP/RH-2 on satisfiable instances of the hard region of the phase transition, and Regular-WalkSAT/Basic outperforms Regular-GSAT/Basic on satisfiable instances of the hard region of the phase transition.

This chapter is organized as follows. In Section 4.2 we present the generator of *Regular Random 3-SAT* instances; in Section 4.3 we provide experimental evidence of the existence of phase transitions and of the fact that the location of the threshold increases logarithmically in the cardinality of the truth value set; in Sections 4.4 and 4.5 we derive upper bounds on the location of the unsatisfiability threshold; in Sections 4.6 we present the comparison of the performance of Regular-DP/RH-1, Regular-DP/RH-2, Regular-GSAT/Basic, and Regular-WalkSAT/Basic. The experiments were performed on Sun Ultra-5 workstations.

4.2 Generator of *Regular Random 3-SAT* instances

The generator of *Regular Random 3-SAT* instances we designed and implemented for conducting the experimental investigation has as parameters the number of clauses (C), the number of propositional variables (V), and the cardinality of the truth value set ($|N|$). Given C and V, an instance of *Regular Random 3-SAT* is produced by generating C non-tautological regular clauses with three literals per clause. Each regular clause is produced by choosing uniformly at random three literals with different propositional variables from the set of regular literals of the form $\downarrow i : p$ or $\uparrow j : p$, where p is a propositional variable, $i \in N \setminus \{\top\}$, $j \in N \setminus \{\bot\}$, and $\top$ and $\bot$ denote the top and bottom elements of N. Observe that regular literals of the form $\downarrow \top : p$ or $\uparrow \bot : p$ are tautological.

The generator of regular *Random* 3-*SAT* instances was implemented in C++. It uses the random number generator `ran1` described in (Press et al., 1988).

4.3 Phase transitions

In this section we first provide experimental evidence of the existence of phase transitions in *Regular Random 3-SAT* when the satisfiability of the instances is determined with Regular-DP/RH-1. Then, we report a series of experiments that indicate that the location of the threshold increases logarithmically in the cardinality of the truth value set.

Let us first describe a particular experiment to illustrate the phase transition phenomenon. We used our generator to produce sets of 300 *Regular Random 3-SAT* instances. For all the sets, the input parameter $|N|$ of the generator was set to 7, and the input parameter V was set to 60. The input parameter C was different for every set of instances: C was 300 in the first set, 305 in the second, 310 in the third, ... , and 960 in the last; we

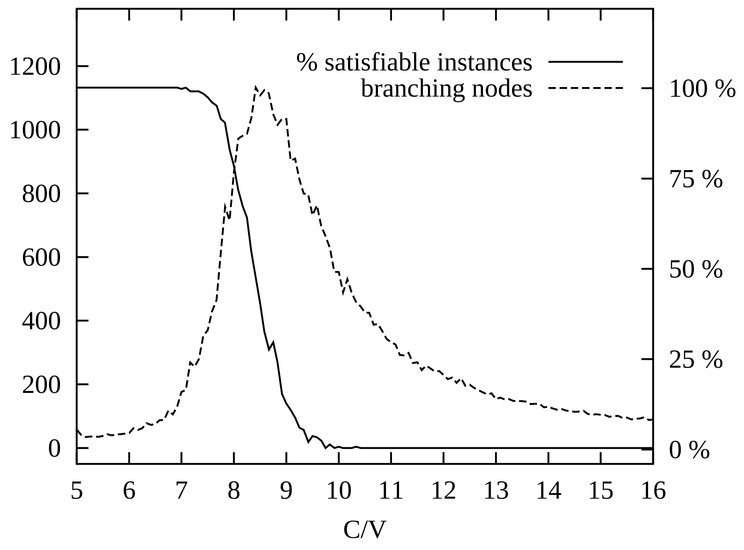


Figure 4.1: *Regular Random 3-SAT*, $|N| = 7$, $V = 60$

incremented the number of clauses by 5 in each step. This way we obtained 133 sets whose ratio of the number of clauses (C) to the number of variables (V) ranged from 5 to 16. Then, we applied Regular-DP/RH-1 to all the instances.

Figure 4.1 summarizes the experimental results obtained and visualizes the phase transition phenomenon in *Regular Random 3-SAT*. The ratio C/V is shown along the horizontal axis. The number of branching nodes is shown along the left vertical axis. For each set of instances, the figure shows the average number of branching nodes per instance in the Regular-DP proof tree as a function of the ratio C/V. Looking at the dashed line, one can clearly observe the easy-hard-easy pattern in the computational difficulty of solving instances as the ratio C/V is varied. The percentage of satisfiable instances is shown along the right vertical axis. The solid line indicates the percentage of instances that were found to be satisfiable.

Let us describe the experiment we conducted to investigate how the location of the threshold varies as a function of the cardinality of the truth

$\lvert N\rvert$	2	3	4	5	6	7	8	9	10
Threshold	4.24	6.04	7.04	7.64	8.14	8.34	8.74	8.84	9.04
$\lvert N\rvert$	11	12	13	14	15	16	17	18	19
Threshold	9.14	9.34	9.34	9.44	9.54	9.54	9.64	9.74	9.74
$\lvert N\rvert$	20	25	30	35	40	45	50	60	70
Threshold	9.74	10.04	10.14	10.24	10.24	10.34	10.44	10.54	10.54

Table 4.1: Location of the threshold for different cardinalities of N

value set. We applied Regular-DP/RH-1 to 60-variable $|N|$-valued *Regular Random 3-SAT* instances for $|N| = 2$–$20, 25, 30, 35, 40, 45, 50, 60, 70$. For each $|N|$, we varied the ratio C/V from 1 to 12 for $|N| < 7$ and from 5 to 16 for $|N| \geq 7$, incrementing the number of clauses by 5. At each setting we ran the algorithm on 300 randomly generated instances. Therefore, the number of instances executed for each $|N|$ was $39,900$, and the total number of instances executed was $1,077,300$. The thresholds obtained are shown in Table 4.1.

The data indicates that the location of the threshold increases logarithmically in the cardinality of the truth value set $|N|$ as the following equation states:

$$L\big(|N|\big) = 6.26 \ln^{0.39}\big(|N|\big).$$

This equation was derived from the experimental thresholds by using the Levenberg-Marquardt method for obtaining a non-linear regression model implemented in the symbolic calculator Mathematica.

Since we had to run Regular-DP/RH-1 on a very large sample of *Regular Random 3-SAT* instances, we considered instances with 60 variables in order to get experimental results in a reasonable amount of time. The more propositional variables occur in an instance of the hard region of the phase transition, the more time is needed to determine whether it is satisfiable.

In the remaining of this section we provide more details about our exper-

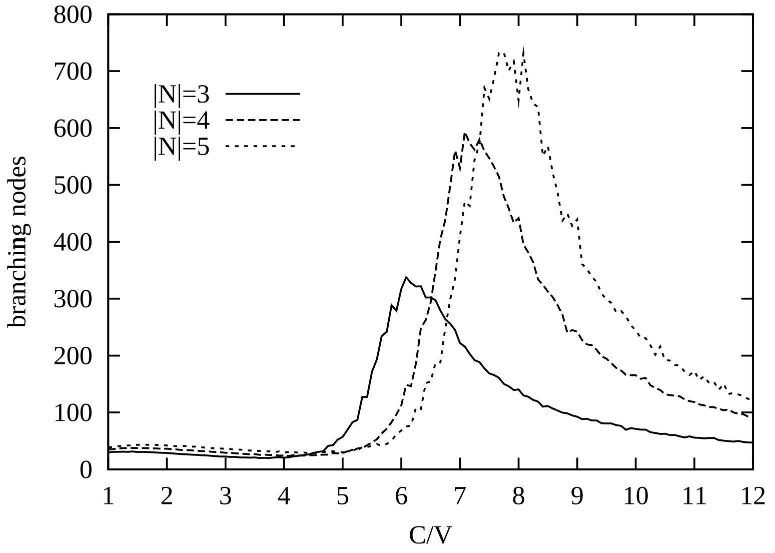


Figure 4.2: *Regular Random 3-SAT*, $|N| = 3, 4, 5$, $V = 60$

imental results by means of figures:

Figure 4.2 visualizes how the location of the threshold increases as the cardinality of the truth value set increases. It shows the average number of branching nodes in the proof tree created by Regular-DP/RH-1 when $|N| = 3$, $|N| = 4$ and $|N| = 5$. The ratio C/V in the *Regular Random 3-SAT* instances tested is shown along the horizontal axis. Figure 4.3 is like Figure 4.2 but for $|N| = 9$, $|N| = 11$, $|N| = 13$ and $|N| = 15$. One can observe in both figures that the average number of branching nodes in the proof tree increases in the area near the threshold as the cardinality of the truth value set increases.

Figure 4.4 displays the percentage of satisfiable instances as a function of the ratio C/V for some of the cardinalities considered in our experiments. Figure 4.5 shows the location of the threshold as a function of $|N|$. The experimental thresholds obtained and the equation derived with Mathematica are both plotted in the graph. One can observe in both figures that the increase of the location of the threshold is not linear.

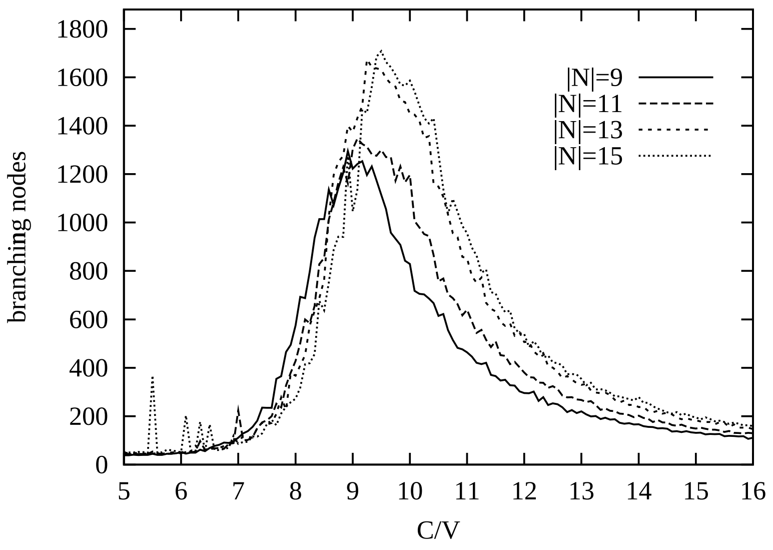


Figure 4.3: *Regular Random 3-SAT*, $|N| = 9, 11, 13, 15$, $V = 60$

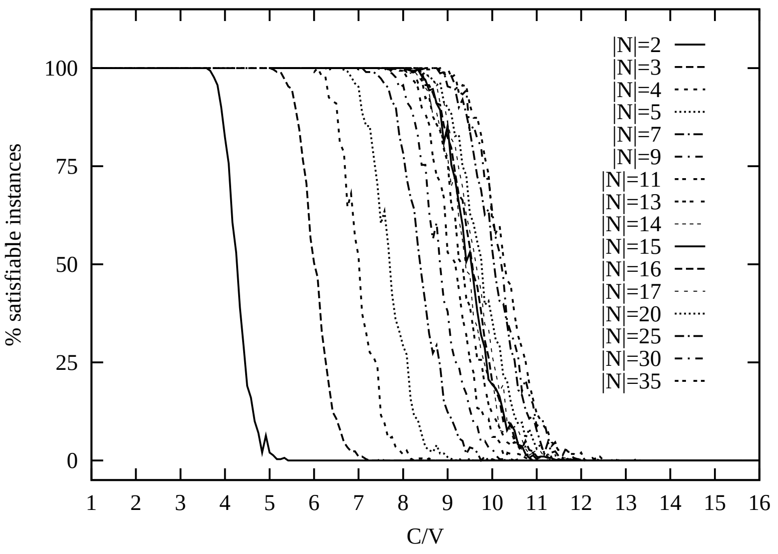


Figure 4.4: Percentage of satisfiable instances as a function of C/V

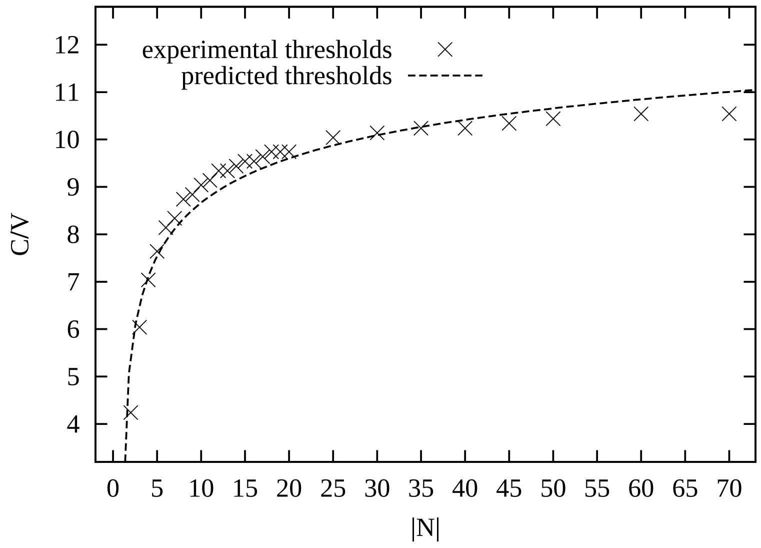


Figure 4.5: Location of the threshold as a function of $|N|$

4.4 An upper bound on the unsatisfiability threshold

Let Γ be a *Regular Random 3-SAT* instance, let V be the number of propositional variables in Γ, let C be the number of regular clauses in Γ, and let $r = C/V$ be the ratio of the number of clauses to the number of variables in Γ. The problem we consider is to compute the least real number k such that if r strictly exceeds k, then the probability that Γ is satisfiable converges to 0 as V approaches infinity. We say in this case that Γ is asymptotically almost certainly unsatisfiable. A proposition stating that if r exceeds a certain constant, then Γ is asymptotically almost certainly unsatisfiable has as an immediate corollary that this constant is an upper bound for k. In this section we obtain an upper bound on the unsatisfiability threshold as a function of the cardinality of the truth value set. This give us a theoretical justification of our experimental results.

We consider the following model of *Regular Random 3-SAT* instances:

from the sample space of non-tautological regular clauses of size three, with different variable in each literal, over V propositional variables and $|N|$ truth values, uniformly, independently, and with replacement select C clauses to form a regular CNF formula Γ. Given an interpretation I, the probability that I satisfies a regular random literal L is $\frac{1}{2}$; observe that the number of regular literals with positive polarity satisfied by an interpretation coincides with the number of regular literals with negative polarity that are not satisfied and vice versa. The probability that I satisfies a regular random clause (with three literals) is $1 - \left(\frac{1}{2}\right)^3 = \frac{7}{8}$. The probability that I satisfies a *Regular Random 3-SAT* instance Γ with C clauses is $\left(\frac{7}{8}\right)^C$. Since there are $|N|^V$ possible interpretations for Γ, the expected number of interpretations that satisfy Γ is

$$E[|\{I \mid I \text{ satisfies } \Gamma\}|] = |N|^V \left(\frac{7}{8}\right)^C.$$

Since the expected number of interpretations that satisfy Γ is an upper bound on the probability that Γ is satisfiable, it holds that

$$Pr[\Gamma \text{ is satisfiable}] \leq E[|\{I \mid I \text{ satisfies } \Gamma\}|] = |N|^V \left(\frac{7}{8}\right)^C;$$

letting $C = rV$, an upper bound for k is found by choosing r so that the expected number of interpretations that satisfy Γ converges to 0 as V approaches infinity. Thus, if $r > \log_{\frac{8}{7}} |N|$, then Γ is almost certainly unsatisfiable.

Therefore, if an upper bound on the unsatisfiability threshold increases logarithmically in the cardinality of the truth value set, the location of the threshold cannot increase quicker than the upper bound does. For that reason, we should get experimental thresholds increasing logarithmically or less than logarithmically in the cardinality of the truth value set.

4.5 A better upper bound on the unsatisfiability threshold

In this section we derive a better upper bound for the location of the unsatisfiability threshold by generalising the technique for *Random 3-SAT* described in (Kirousis et al., 1998). The regular CNF formulas considered in this section have clauses of size three, and the same propositional variable never occurs more than once in the same clause.

In order to generalise the above technique, we first define a subclass of satisfying interpretations of *Regular Random 3-SAT* instances. This subclass is a subset of the total set of satisfying interpretations of a formula, and the idea of the technique is to analyse when the expected value for the cardinality of this subset converges to zero.

Definition 4.1 *Given a Regular Random 3-SAT instance Γ over V variables, $\mathcal{I}_V$ is the set of interpretations for Γ, f_V is the set of satisfying interpretations of Γ, and $f_V^\star$ is the subset of satisfying interpretations I from f_V such that any interpretation obtained from I by changing exactly the truth value of one variable to any truth value greater than the current one does not satisfy Γ.*

We call the process of changing the truth value of one variable to any of its greater truth values a single change (sc); and denote this process, for a particular interpretation I, by I^{sc}. Observe that any variable with an assigned truth value smaller than the top element of N can be changed. We denote by $sc(I)$ the number of variables that can be changed in an interpretation I. As in the classical case, this subclass of satisfying interpretations is a subset of f_V with elements that are local maxima in the lexicographic ordering.

Lemma 4.1 *The expected value of the random variable $|f_V^\star|$ is given by the formula*

$$E[|f_V^\star|] = (7/8)^{rV} \sum_{I \in \mathcal{I}_V} Pr\left[I \in f_V^\star \mid I \in f_V\right]$$

In the lemma, r is the ratio of the number of clauses to variables and therefore rV is the number of clauses of the formula. Remember that the first term in the expression is the probability that a fixed interpretation satisfies a *Regular Random 3-SAT* instance with rV clauses. Since this lemma is a straightforward generalization of Lemma 2 in (Kirousis et al., 1998), its proof remains valid for our lemma.

The following theorem is the result of the generalization of the proof in (Kirousis et al., 1998) for our case.

Theorem 4.1 *The expected value $E[|f_V^\star|]$ is at most*

$$(7/8)^{rV}\left(|N| - e^{\left(\frac{-3r}{7(|N|-1)}\right)} + o(1)\right)^V .$$

Then, the unique positive solution of the equation

$$(7/8)^{r}\left(|N| - e^{\left(\frac{-3r}{7(|N|-1)}\right)}\right) = 1$$

is an upper bound for k, the unsatisfiability threshold.

The proof is very similar to the proof for classical CNF formulas. First, we need an expression for the probability that, given an interpretation I and a fixed single change sc, $I^{sc} \not\models \Gamma$ under the assumption that $I \models \Gamma$.

Observe that the assumption $I \models \Gamma$ excludes $(|N|-1)^3\binom{V}{3}$ clauses from the set of possible clauses that can appear in Γ, because this is the number of different clauses that we can build with V variables and $|N|$ truth values that are not satisfied by a particular interpretation I. So, we have $7(|N|-1)^3\binom{V}{3}$ different clauses that are satisfied by I. From these clauses, the number of clauses that contain the changed variable and that become unsatisfied if we change the truth value of the changed variable to any of its greater values is equal to

$$(|N|-1)^2\binom{V-1}{2},$$

because there is only one possible regular literal with the changed variable that is not satisfied by I^{sc}. Then, for any clause C of Γ we have that

$$P[I^{sc} \not\models C | I \models C] = \frac{(|N|-1)^2\binom{V-1}{2}}{7(|N|-1)^3\binom{V}{3}} = \frac{3}{7(|N|-1)V}.$$

The probability that it does not satisfy Γ is equal to

$$1 - p(V, r, |N|),$$

where $p(V, r, |N|)$ is the probability that I^{sc} satisfies all the clauses of Γ; i.e.,

$$p(V, r, |N|) = \left(1 - \frac{3}{7(|N|-1)V}\right)^{(rV)}.$$

Since the number of single changes for I is $sc(I)$, and following the same argument as Kirousis et al. (1998), we derive the following upper bound on the probability that no single change satisfies Γ:

$$Pr\left[I \in f_V^\star \mid I \in f_V\right] \leq (1 - p(V, r, |N|))^{sc(I)} = (1 - e^{\left(\frac{-3r}{7(|N|-1)}\right)} + o(1))^{sc(I)}.$$

Since the maximum number of single changes is V, we derive the following upper bound on the expected value of $E[|f_V^\star|]$ (when $V \to \infty$):

$$\begin{aligned} E[|f_V^\star|] &\leq (7/8)^{rV}\left(|N|^V\left(1 - e^{\left(\frac{-3r}{7(|N|-1)}\right)}\right)^V\right) \\ &\leq (7/8)^{rV}\left(|N| - e^{\left(\frac{-3r}{7(|N|-1)}\right)}\right)^V \end{aligned}$$

Then, it follows that $E[|f_V^\star|]$ converges to zero for values of r that strictly exceed the unique positive solution of the equation $(7/8)^r\left(|N| - e^{\left(\frac{-3r}{7(|N|-1)}\right)}\right) = 1$. Therefore, as the expected value of $E[|f_V^\star|]$ is an upper bound on the probability that the formula is satisfiable (because $f_V^\star \neq \emptyset$ for a satisfiable instance) this solution provides an upper bound for k.

4.6 Experimental results

In this section we report on an experimental investigation performed in order to compare (i) Regular-DP/RH-1 with Regular-DP/RH-2, and (ii) Regular-DP/RH-2 with Regular-GSAT/Basic and Regular-WalkSAT/Basic when solving *Regular Random 3-SAT* instances

$\lvert N\rvert$		V = 100	120	140	160	180
3	C	609	720	848	976	1100
	C/V	6.09	6.00	6.06	6.10	6.11
4	C	712	855	997	1139	1284
	C/V	7.12	7.13	7.12	7.12	7.13
5	C	772	926	1090	1248	1406
	C/V	7.72	7.72	7.79	7.8	7.81

Table 4.2: *Regular Random 3-SAT* test-sets: parameter settings of the generator

from the hard region of the phase transition. Such experiments were performed on Sun Ultra-5 workstations.

We considered 15 test-sets of *Regular Random 3-SAT* instances in which the ratio C/V corresponds to the location of the threshold of the phase transition. The instances were generated using the setting of C, V and $|N|$ shown in Table 4.2. Observe that the location of the threshold, for a fixed $|N|$, varies slightly as we increase the parameter V; this also happens in (classical) *Random 3-SAT* (Crawford and Auton, 1996). Every test-set contains 100 satisfiable instances and 100 unsatisfiable instances.

Comparison of branching heuristics

We performed an experiment in order to compare the performance of Regular-DP/RH-1 and Regular-DP/RH-2. To this end, we ran the two algorithms on both the satisfiable and unsatisfiable instances of the above described test-sets for $V = 100$, $V = 120$, and $V = 140$. Table 4.3 shows the results obtained for Regular-DP/RH-1, and Table 4.4 shows the results obtained for Regular-DP/RH-2. We measured both the average time needed to solve an instance, and the average number of branching nodes (br. nodes)

$\|N\|$		V = 100 sat	100 unsat	120 sat	120 unsat	140 sat	140 unsat
3	br. nodes	2715	4708	9755	20912	33519	71546
	time (sec)	4.21	7.46	17.16	37.31	69.40	151.49
4	br. nodes	7293	11695	23523	51295	88212	203382
	time (sec)	12.69	21.64	50.75	113.43	226.87	528.36
5	br. nodes	11420	21352	41536	90294	206011	465440
	time (sec)	23.13	44.03	100.75	223.13	596.27	1370.13

Table 4.3: Results of Regular-DP/RH-1 on *Regular Random 3-SAT*

per instance in the Regular-DP proof tree. The results are shown separately for satisfiable (sat) and unsatisfiable (unsat) instances.

For comparing the relative performance of Regular-DP/RH-1 with respect to Regular-DP/RH-2, we measured the ratio of the average number of branching nodes per instance obtained with Regular-DP/RH-1 to the average number of branching nodes per instance obtained with Regular-DP/RH-2. We also measured the ratio of the average time needed to solve an instance with Regular-DP/RH-1 to the average time needed to solve an instance with Regular-DP/RH-2. These ratios give us a measure of the scaling behaviour of the relative performance between both algorithms.

Table 4.5 shows the results obtained. We observe that both ratios increase as we increment V and $|N|$, and provide experimental evidence that, at least for the class of instances tested, Regular-DP/RH-2 outperforms Regular-DP/RH-1.

$\|N\|$		V					
		100		120		140	
		sat	unsat	sat	unsat	sat	unsat
3	br. nodes	1617	2823	5320	10930	16232	34135
	time (sec)	1.48	2.63	5.57	11.51	21.06	44.58
4	br. nodes	2691	4528	9454	18066	25927	55695
	time (sec)	2.95	5.06	12.29	23.48	44.23	93.84
5	br. nodes	3350	6102	11404	21961	34323	76144
	time (sec)	4.28	7.96	18.16	35.57	64.99	145.47

Table 4.4: Results of Regular-DP/RH-2 on *Regular Random 3-SAT*

$\|N\|$		V					
		100		120		140	
		sat	unsat	sat	unsat	sat	unsat
3	br. nodes	1.68	1.67	1.83	1.91	2.06	2.10
	time (sec)	2.84	2.84	3.08	3.24	3.30	3.40
4	br. nodes	2.71	2.58	2.49	2.84	3.40	3.65
	time (sec)	4.30	4.28	4.13	4.83	5.13	5.63
5	br. nodes	3.41	3.50	3.64	4.11	6.00	6.11
	time (sec)	5.40	5.53	5.55	6.27	9.17	9.42

Table 4.5: Relative performance of Regular-DP/RH-1 wrt Regular-DP/RH-2 on *Regular Random 3-SAT*

		V				
$\|N\|$		100	120	140	160	180
3	MaxChanges	6000	8000	12000	17000	24000
	ω	0.53	0.50	0.45	0.42	0.40
4	MaxChanges	9000	14000	18000	24000	35000
	ω	0.45	0.43	0.42	0.42	0.38
5	MaxChanges	14000	17000	24000	36000	47000
	ω	0.40	0.40	0.38	0.38	0.38

Table 4.6: Regular-GSAT and Regular-WalkSAT: parameter settings for *Regular Random 3-SAT*

Systematic versus local search

We performed an experiment in order to compare the performance of Regular-DP/RH-2 with the performance of Regular-GSAT/Basic and Regular-WalkSAT/Basic on the satisfiable instances of the test-sets of Table 4.2. Since local search algorithms are incomplete, we did not consider unsatisfiable instances. Table 4.6 shows, for each test-set, the MaxChanges parameter used by Regular-GSAT/Basic and Regular-WalkSAT/Basic, and the noise parameter (ω) used by Regular-WalkSAT/Basic. The displayed values of MaxChanges and ω are the best parameter settings we obtained experimentally for the satisfiable instances of the test-sets. To obtain more accurate results, each instance was run 25 times with Regular-GSAT/Basic and 25 times with Regular-WalkSAT/Basic.

Table 4.7 shows the results obtained with Regular-GSAT/Basic. For each test-set, it shows the average number of tries and the average time needed to solve an instance of the test-set. The run time of each instance solved with Regular-GSAT/Basic is the average run time of 25 runs on that instance. Table 4.8 shows the same results but for Regular-WalkSAT/Basic. Finally,

		V				
$\|N\|$		100	120	140	160	180
3	tries	35	71	67	97	169
	time (sec)	1.7	9.7	13.4	29.2	71.2
4	tries	97	239	276	381	443
	time (sec)	9.3	42.3	82.3	172.6	277.1
5	tries	183	265	610	394	460
	time (sec)	25.6	74.2	245.6	270.5	412.2

Table 4.7: Results of Regular-GSAT on *Regular Random 3-SAT*

		V				
$\|N\|$		100	120	140	160	180
3	tries	9	12	9	10	8
	time (sec)	0.9	1.8	1.7	2.6	3.3
4	tries	12	13	10	5	6
	time (sec)	1.6	3.1	3.5	1.6	3.6
5	tries	8	8	10	4	5
	time (sec)	1.8	2.4	4.3	2.7	4.2

Table 4.8: Results of Regular-WalkSAT on *Regular Random 3-SAT*

		V				
$\|N\|$		100	120	140	160	180
3	Regular-DP/RH-2	1.5	5.5	21.1	54.2	187.5
	Regular-GSAT/Basic	1.7	9.7	13.4	29.2	71.2
	Regular-WalkSAT/Basic	0.9	1.8	1.7	2.6	3.3
4	Regular-DP/RH-2	2.9	12.3	44.2	181.5	472.1
	Regular-GSAT/Basic	9.3	42.3	82.3	172.6	277.1
	Regular-WalkSAT/Basic	1.6	3.1	3.5	1.6	3.6
5	Regular-DP/RH-2	4.3	18.2	64.8	286.3	899.4
	Regular-GSAT/Basic	25.6	74.2	245.6	270.5	412.2
	Regular-WalkSAT/Basic	1.8	2.4	4.3	2.7	4.2

Table 4.9: Comparison of the average time (sec) needed to solve an instance with Regular-DP, Regular-GSAT and Regular-WalkSAT

Table 4.9 shows a comparison of the average time needed to solve an instance with Regular-DP, Regular-GSAT/Basic and Regular-WalkSAT/Basic for each test-set. From the experimental results we can conclude that local search algorithms (Regular-GSAT and Regular-WalkSAT) outperform systematic search algorithms (Regular-DP) on satisfiable instances of the hard region of the phase transition. Moreover, Regular-WalkSAT scales better than Regular-GSAT as the number of variables and the number of truth values in the problem instances increase.

Such results indicate that local search algorithms can extend the range and size of satisfiability problems that can be efficiently solved in many-valued logics.

Chapter 5

Experimental Investigation

5.1 Introduction

In this chapter we describe a comprehensive experimental investigation conducted for analysing and comparing the algorithms for *Regular-SAT* that we designed and implemented, as well as for providing experimental evidence of the practical usefulness of the generic problem solving approach which consists in modeling hard combinatorial problems as *Regular-SAT* instances and then solving the resulting encodings using algorithms for *Regular-SAT*.

There are publicly available sets of benchmark instances for *SAT* which are widely used to conduct experimental investigations. The most complete and recently created benchmark library is SATLIB (Hoos and Stützle, 2000b). To our best knowledge, there is no publicly available set of benchmark instances for *Regular-SAT*. We thus started our experimental investigation by constructing the first collection of benchmark instances for this problem. To this end, we selected a representative sample of combinatorial problems and defined suitable *Regular-SAT* encodings for them. We also defined a data format for *Regular-SAT* instances which is an adaptation of the file format defined by the DIMACS Challenge Committee for classical CNF formulas (DIM, 1993). This format, described in Appendix A, is accepted by the regular satisfiability algorithms described in this thesis.

The combinatorial problems selected were: the k-colourability problem of graphs, the round robin problem, and the all-interval-series problem. We selected such problems because they have computationally difficult instances which are standard benchmarks in the communities of constraint programming, operations research, and *SAT*; see (Johnson et al., 1991; Selman and Kautz, 1993; McAloon et al., 1997; Gomes et al., 1998b; Hoos and Stützle, 2000a; Schuurmans and Southey, 2000). We considered the same set of benchmark instances that were used in previous empirical studies in order to compare our experimental results with other published results.

We encoded all the benchmark instances considered in our experiments as *SAT* and as *Regular-SAT* instances, and we then tested both kinds of instances in the same computing environment. The *SAT* instances were tested using the fastest existing *SAT* solvers whereas the *Regular-SAT* instances were tested using our implementations of the algorithms for *Regular-SAT* described in Chapter 3.

For the above combinatorial problems, the performance of the fastest *SAT* solvers with respect to the performance obtained using other problem solving approaches is well-known and well-documented. Thus, by reproducing *SAT*-based experiments we were able to compare the experimental results obtained using our *Regular-SAT* solvers with the best results obtained using other problem solving approaches.

An important part of our experimental investigation was devoted to the definition of suitable encodings for modeling hard combinatorial problems as *SAT* and as *Regular-SAT* instances. For some problems, we observed that minor changes in the encoding can vary remarkably the time needed to solve the problem instances. A challenging question that we would like to investigate in the near future is to deepen our understanding of *SAT* and *Regular-SAT* encodings so that they can be applied more effectively to solve combinatorial problems. For instance, we would like to identify design principles which make an encoding successful; and to understand how encoding components, problem characteristics and good performance for particular satisfiability algorithms are related.

Interestingly, we defined *SAT* and *Regular-SAT* encodings of the round robin problem that gave rise to a dramatic improvement with respect to other approaches; we solved instances of the round robin problem which cannot be solved using solvers based on integer programming and constraint programming. We also defined better *SAT* encodings for the all-interval-series problem. Therefore, this thesis not only contains contributions to *Regular-SAT*, but also to *SAT*.

This experimental investigation allowed us to have a clear picture of the strengths and weaknesses of our generic problem solving approach; helped us in improving our implementations, and in designing better heuristics and algorithms; and provided experimental evidence that, at least for the combinatorial problems studied, our problem solving approach can outperform or compete with state-of-the-art problem solving approaches.

As said in Chapter 4, the *Regular Random 3-SAT* instances of the hard region of the phase transition are also suitable benchmark instances for evaluating and comparing algorithms for *Regular-SAT*. Actually, the experimental results reported in Chapter 4 are part of the experimental investigation conducted in this thesis. We decided, however, to include those results in the chapter of phase transitions, and to devote this chapter to more realistic problems which can be naturally modeled as *SAT* instances and are standard benchmarks in other research communities.

This chapter is organized as follows. In Section 5.2 we focus on the k-colourability problem of graphs, in Section 5.3 we focus on the round robin problem, and in Section 5.4 we focus on the all-interval-series problem. For each problem we provide its definition, its *SAT* and *Regular-SAT* encodings, and report the experimental results obtained and the conclusion than can be drawn from these results. In the description of the round robin problem we follow closely the presentation of (Gomes et al., 1998b), and in the description of the all-interval-series problem we follow closely the presentation of (Hoos, 1998).

5.2 The k-colourability problem of graphs

The k-colourability problem is a well-known combinatorial problem from graph theory: given an undirected graph $G = (V, E)$, where V is the set of vertices and E is the set of edges, determine if there is a colouring such that connected vertices always have different colour; i.e., determine if there is a function $c : V \to \{1, \ldots, k\}$ such that $c(u) \neq c(v)$ for each edge $[u, v] \in E$.

5.2.1 *SAT* encoding of the k-colourability problem

Given an undirected graph $G = (V, E)$, the k-colourability problem of G can be encoded as an instance of *SAT* as follows:

1. The set of propositional variables is

$$\{u_i \mid u \in V \text{ and } 1 \leq i \leq k\},$$

   and its cardinality is $k \cdot |V|$. The intended meaning of u_i is that vertex u is coloured with colour $i \in \{1, \ldots, k\}$,

2. For each vertex $u \in V$, we define the following clauses:

$$\bigvee_{i \in \{1,\ldots,k\}} u_i \wedge \bigwedge_{\substack{i, j \in \{1, \ldots, k\} \\ i \neq j}} (\neg u_i \vee \neg u_j)$$

   The intended meaning of the whole expression is that vertex u is coloured with exactly one colour.

3. For each edge $[u, v] \in E$, we define the following clauses:

$$\bigwedge_{i \in \{1,\ldots,k\}} (\neg u_i \vee \neg v_i)$$

   The intended meaning of this expression is that the adjacent vertices u and v do not have the same colour.

5.2.2 ***Regular-SAT*** encoding of the k-colourability problem

Given an undirected graph $G = (V, E)$, the k-colourability problem of G can be encoded as an instance of *Regular-SAT* as follows:

1. The truth value set N is $\{1, \ldots, k\}$. Each truth value represents a colour.
2. The set of propositional variables is V, and its cardinality is $|V|$.
3. For each edge $[u, v] \in E$, we define k regular clauses:

$$\begin{array}{ll}
(C_1) & \uparrow 2 : u \vee \uparrow 2 : v \\
(C_2) & \downarrow 1 : u \vee \uparrow 3 : u \vee \downarrow 1 : v \vee \uparrow 3 : v \\
\vdots & \qquad\qquad \vdots \\
(C_i) & \downarrow (i-1) : u \vee \uparrow (i+1) : u \vee \downarrow (i-1) : v \vee \uparrow (i+1) : v \\
\vdots & \qquad\qquad \vdots \\
(C_{k-1}) & \downarrow (k-2) : u \vee \uparrow k : u \vee \downarrow (k-2) : v \vee \uparrow k : v \\
(C_k) & \downarrow (k-1) : u \vee \downarrow (k-1) : v
\end{array}$$

The intended meaning of the previous regular clauses is that vertex u and vertex v do not have the same colour. For each $i \in \{1, \ldots, k\}$, the intended meaning of regular clause C_i is that vertex u and vertex v are not both coloured with colour i. Observe that from the definition of interpretation we can ensure that every vertex is coloured with exactly one colour. Also observe that the size of the regular CNF formula obtained is in $\mathcal{O}(|N| \cdot |E|)$, where $|N|$ is the number of colours and $|E|$ is the number of edges. Thus, the k-colourability problem for an undirected graph can be represented in a more concise way if we use regular CNF formulas instead of classical CNF formulas.

test-set	**Parameters**					
	vertices	p	edges	$\frac{\text{edges}}{\text{vertices}}$	flatness	colours
`flat100-3c`	100	0.072	239	2.39	0	3
`flat100-4c`	100	0.124	465	4.65	0	4
`flat150-3c`	150	0.048	360	2.40	0	3
`flat150-4c`	150	0.074	623	4.15	0	4

Table 5.1: Flat graph generator: parameter settings

5.2.3 Experimental results

In this section we describe an experimental investigation conducted for comparing the performance of algorithms for *SAT* on graph colouring instances with the performance of algorithms for *Regular-SAT* on the same instances. Such experiments were performed on Sun Ultra-5 workstations.

We first considered four different test-sets of k-colourable, flat graph colouring instances (Culberson and Luo, 1996); each test-set contained 100 randomly generated instances. Two test-sets (`flat100-3c` and `flat150-3c`) are from the SATLIB. The other two test-sets (`flat100-4c` and `flat150-4c`) were generated with a publicly available flat graph generator;[1] the number of vertices, the number of colours, the edge probability (p), and the flatness parameter used by the generator are shown in Table 5.1. Test-sets have different settings of the number of vertices (100 and 150) and of the number of colours (3 and 4). The settings of the edge probability and the flatness parameter employed allowed us to generate graphs which are, on average, computationally difficult.

It was shown in (Hoos and Stützle, 2000a) that WalkSAT/Basic is a suitable algorithm for solving *SAT* encoded flat graphs. In our experiments

[1]The generator is available from `http://web.cs.ualberta.ca/~joe/Coloring/`, Joe Culberson's Graph Coloring Page. This generator was also employed to generate `flat100-3c` and `flat150-3c`.

		vertices = 100				**vertices = 150**			
		Encoding				**Encoding**			
k		classical		regular		classical		regular	
		mean	CV	mean	CV	mean	CV	mean	CV
3	ω	0.50		0.42		0.50		0.44	
	changes	44945	0.87	8730	0.81	183992	1.22	35696	1.13
	time (sec)	0.24	0.86	0.12	0.82	0.92	1.21	0.52	1.13
4	ω	0.46		0.38		0.44		0.38	
	changes	60880	0.69	14578	0.63	397862	2.31	69121	1.85
	time (sec)	0.45	0.68	0.29	0.62	3.02	2.48	1.48	1.84

Table 5.2: Results of WalkSAT/Basic and Regular-WalkSAT/Basic on flat graph instances

we used WalkSAT/Basic and Regular-WalkSAT/Basic. We used the same basic algorithm for *SAT* and Regular-SAT instances in order to focus on the effects of using different encodings (classical and regular).

For each test-set and for each kind of encoding, Table 5.2 shows the noise setting parameter (ω) employed, the mean number of changes needed to solve an instance, and the mean time needed to solve an instance. Each instance was executed 100 times using an approximately optimal noise parameter, MaxTries was set to 1, and MaxChanges was set to an extremely high value for obtaining a solution in only one try. We also report the coefficient of variation (CV) for the distributions of the mean number of changes and the mean time. The CV of a distribution gives a measure of the relative dispersion of the values of the distribution with respect to its mean value, and it is calculated as the ratio of the standard deviation to the mean.

The results of Table 5.2 show that the mean number of changes and the mean time needed to solve an instance using the *Regular-SAT* encoding is always smaller than using the *SAT* encoding. The mean number of changes is always between 4 and 6 times smaller, and the mean time is always be-

	changes/second		
	classical	regular	$\frac{\text{classical}}{\text{regular}}$
`flat100-3c`	187270	72750	2.57
`flat100-4c`	135288	49526	2.73
`flat150-3c`	199140	68646	2.90
`flat150-4c`	131742	46704	2.82

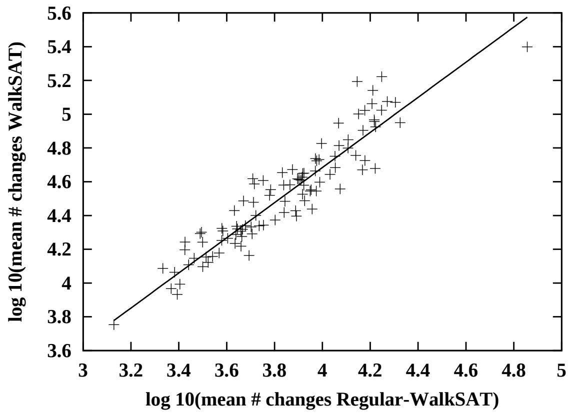
Table 5.3: Comparison of changes per second on flat graph instances

tween 1.5 and 2 times smaller. Moreover, the CV is always smaller with the *Regular-SAT* encoding, and the difference becomes more significant as we increase the number of vertices and the number of colours.

The fact that the difference in the mean time needed to solve the flat graphs is less significant than the number of changes needed is simply a consequence of the fact that the average time needed to perform a change is greater in Regular-WalkSAT/Basic than in WalkSAT/Basic because the number of clauses affected by a simple change is greater. A simple explanation for this difference is the following one. The two encodings produce formulas with a number of clauses very similar, but the number of variables is smaller in the *Regular-SAT* encoding. So, a variable in a *Regular-SAT* instance must appear in more clauses than a variable in the corresponding *SAT* instance.

For comparing the mean time needed to perform changes with WalkSAT/Basic and Regular-WalkSAT/Basic, Table 5.3 shows the mean number of changes per second performed by each algorithm. Observe that the ratio of the mean number of changes per second in WalkSAT/Basic to the mean number of changes per second in Regular-WalkSAT/Basic indicates that the relative speed between them does not vary significantly as the number of vertices and the number of colours increase.

In the following, when we say hardness distribution we mean the distribu-



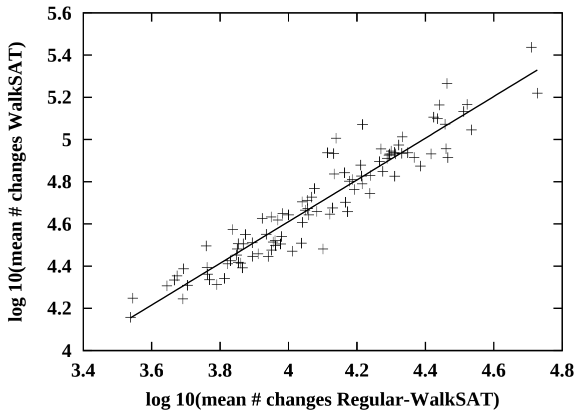


Figure 5.1: Correlation between mean hardness with Regular-WalkSAT/Basic and WalkSAT/Basic for `flat100-3c` (left) and `flat100-4c` (right)

tion of the mean number of changes needed to solve an instance. The results of Table 5.2 indicate that the mean hardness for a test-set is greater with WalkSAT/Basic than with Regular-WalkSAT/Basic, but they tell us nothing about the relation between the hardness of individual instances. To this end, we have performed a correlation analysis for the hardness of the instances of each test-set.

Figure 5.1 shows two log-log scatter plots of the correlation between the mean hardness of solving an instance with Regular-WalkSAT/Basic and of solving the same instance with WalkSAT/Basic. The left plot corresponds to the instances of `flat100-3c`, and the right plot corresponds to the instances of `flat100-4c`. Figure 5.2 shows the same plots but for the test-sets `flat150-3c` and `flat150-4c`. Every point (x, y) in the scatter plot represents, in logarithmic scale, the mean hardness associated with a particular instance of the test-set when this instance is solved with Regular-WalkSAT/Basic (x-axis) and with WalkSAT/Basic (y-axis). The scatter plots indicate a clear positive correlation between the mean hardness; i.e., the mean hardness in WalkSAT/Basic tends to be greater than the mean hardness in Regular-WalkSAT/Basic in each instance of the test-sets

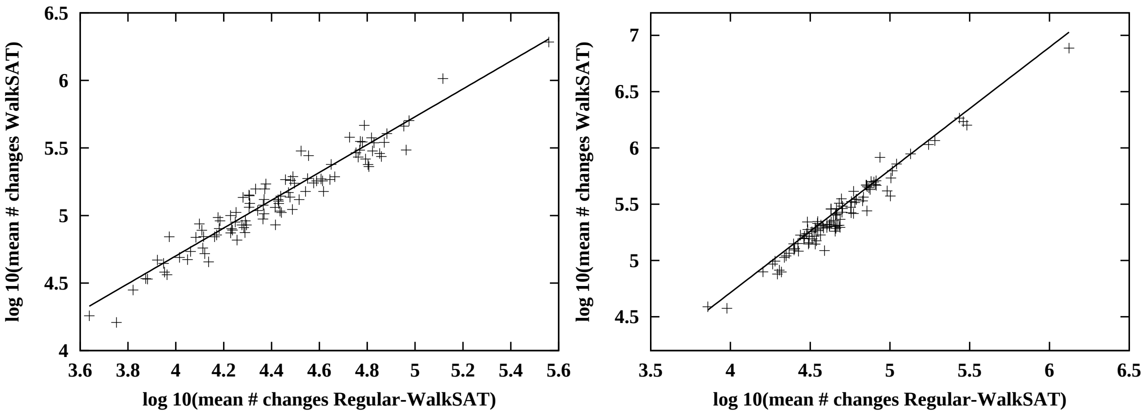


Figure 5.2: Correlation between mean hardness with Regular-WalkSAT/Basic and WalkSAT/Basic for `flat150-3c` (left) and `flat150-4c` (right)

considered. This type of hardness correlation analysis was used to compare *SAT* and CSP encodings in (Hoos, 1999b).

To quantify the correlations suggested by the scatter plots, we also performed least-mean-squares (lms) linear regression analysis of the logarithm of the mean hardness for each test-set. The resulting linear equations $log_{10}(y) = a\log_{10}(x) + b'$ are plotted together with their corresponding scatter plots, and they correspond to power functions of the type $y = b\,x^a$, where $b = 10^{b'}$. We work with a power function model because the parameter a allows us to model possible performance differences between Regular-WalkSAT/Basic and WalkSAT/Basic as the intrinsic hardness increases.[2] Table 5.4 shows the results of the linear regression analysis. This table shows the equations obtained, the adjusted coefficient of determination (R_a^2), and the values of P associated with the statistical tests for the hypothesis $a = 0$ and $b = 1$. The R_a^2 value is defined as

$$R_a^2 = 1 - \frac{n-1}{n-2}\left(1 - \frac{\sum_{i=1}^{n}(\hat{y}_i - \bar{y})^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}\right),$$

[2]By working with the logarithm of the data we can perform the regression analysis of a power function model as a linear regression analysis.

test-set	**Results**			
	Regression equation ($y = b\,x^a$)	R_a^2	P for $a = 0$	P for $b = 1$
`flat100-3c`	$y = 3.34x^{1.04}$	0.873	< 0.0001	< 0.001
`flat100-4c`	$y = 4.55x^{0.98}$	0.895	< 0.0001	< 0.0001
`flat150-3c`	$y = 3.81x^{1.03}$	0.923	< 0.0001	< 0.0001
`flat150-4c`	$y = 2.26x^{1.09}$	0.956	< 0.0001	< 0.002

Table 5.4: Results of hardness regression analysis between Regular-WalkSAT/Basic (x) and WalkSAT/Basic (y) on flat graph instances

where y_i denotes the mean hardness associated with the *ith* instance of the test-set when this instance is solved with WalkSAT/Basic, $\bar{y}$ denotes the mean value of all the y_i values and $\hat{y}_i$ denotes the predicted mean hardness by the linear regression model obtained ($\hat{y}_i = a\log_{10}(x_i) + b'$). R_a^2 varies from 0 to 1 and gives a measure of the level of correlation, under the given model, between the data. Given a particular hypothesis that we want to validate or refute, and a statistic obtained from the empirical data of the experiment, the value of P is the probability of obtaining a value for the statistic equal to or greater than the one observed if the hypothesis is true. Low values of P for the test $a = 0$ indicate rejection of the hypothesis of no correlation, under the model $y = b\,x^a$, between the data; and low values of P for the test $b = 1$ indicate rejection of the hypothesis of no significant effect of the parameter b in the regression model. The values of P in the table indicate that there is a correlation between the data under the given model.

The value of R_a^2 shows that the correlation becomes stronger as we increase the number of vertices and the number of colours of the graphs. We also observe that there is a slight increase in the hardness difference modeled by the equations. This increase is more significant in `flat100-4c` and `flat150-4c`, when increasing the hardness in Regular-WalkSAT/Basic (x

value), due to the difference in the parameter a.

Observe that the hardness analysis that we have performed with the flat graphs is based on experiments where both algorithms —WalkSAT/Basic and Regular-WalkSAT/Basic— were executed with an approximately optimal noise parameter, MaxTries was set to 1, and MaxChanges was set to an extremely high value for obtaining a solution in only one try. Since they find a solution in each try, both algorithms are PAC for the flat graph instances considered.

It could be argued that the best performance of the algorithms could be obtained with other parameter settings. However, it was shown in (Hoos, 1998) that WalkSAT/Basic shows exponential run-length distributions (RLDs) when the flat graph instances are solved with an approximately optimal noise parameter. If a local search algorithm shows exponential RLDs, the probability of finding a solution in a try with t changes is the same as the probability of finding a solution in k tries of t/k changes every try. Thus, it does not make sense to use the algorithm with restarts (MaxTries>1) and finite values of MaxChanges.

The empirical RLD on an instance is obtained by recording the number of changes performed in each jth successful try ($changes(j)$). Given a set of n tries, the RLD in its cumulative form is defined as

$$P(changes \leq t) = \frac{|\{j \mid changes(j) \leq t\}|}{n}$$

By using all the data obtained in the experiments with the flat graphs, we obtained empirical RLDs for WalkSAT/Basic and Regular-WalkSAT/Basic on the flat graph instances. We observed that both WalkSAT/Basic and Regular-WalkSAT/Basic showed exponential RLDs. Here we concentrate on the results with three instances of `flat150-4c`. The three instances selected were the easy, medium and hard instances of the test-set. To identify these instances, we sorted all the instances of the test-set by the Regular-WalkSAT/Basic mean hardness. For obtaining more accurate RLDs, these instances were solved 400 times. The RLDs for WalkSAT/Basic and

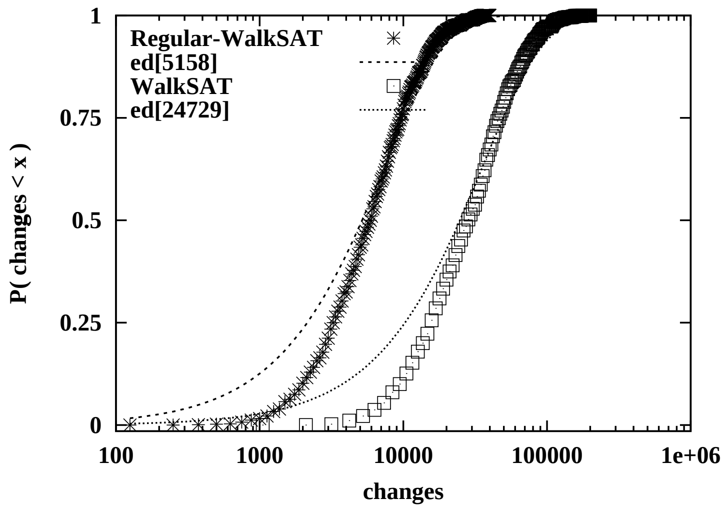


Figure 5.3: RLDs for Regular-WalkSAT/Basic and WalkSAT/Basic on the easy instance of `flat150-4c`

Regular-WalkSAT/Basic were approximated with the cumulative form of an exponential distribution (ED):

$$ed[m](x) = 1 - 2^{-x/m},$$

where m is the median of the distribution. The approximation was derived using the Marquart-Levenberg algorithm. Figures 5.3, 5.4 and 5.5 show the empirical cumulative RLDs for the algorithms on the three instances, as well as the corresponding exponential RLDs obtained. We observe that Regular-WalkSAT/Basic dominates WalkSAT/Basic on the three instances; i.e., the probability of finding a solution with Regular-WalkSAT/Basic in less than t changes is always greater than the probability of finding a solution with WalkSAT/Basic. Thus, for any value of MaxChanges, the success probability for Regular-WalkSAT/Basic is always greater than for WalkSAT/Basic.

For testing the goodness-of-fit of the exponential distributions we used the χ^2 test statistic. Given an empirical RLD and the corresponding exponential distribution $ed[m]$ obtained by the approximation, it generates a statistic —the value χ^2— that is used to measure the correspondence be-

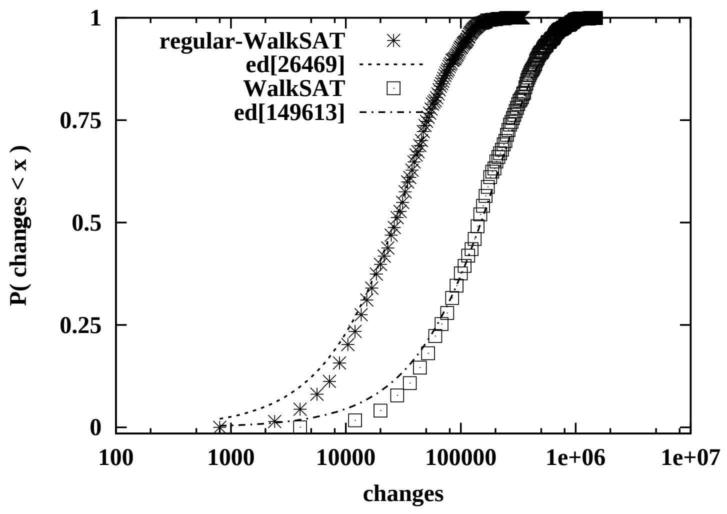


Figure 5.4: RLDs for Regular-WalkSAT/Basic and WalkSAT/Basic on the medium instance of `flat150-4c`

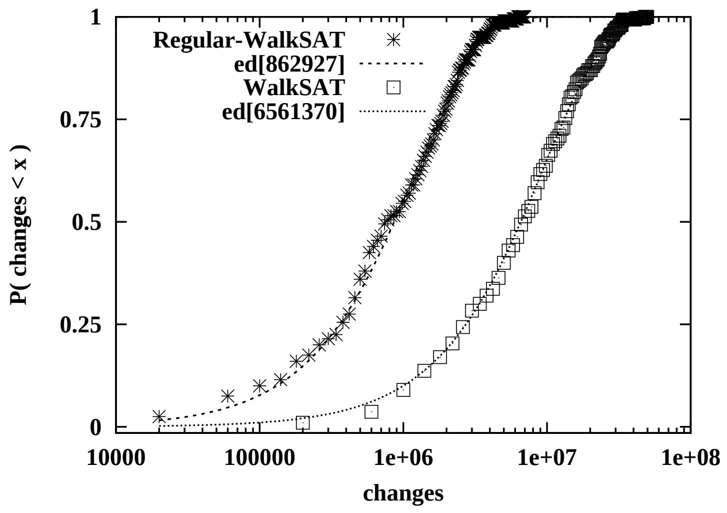


Figure 5.5: RLDs for Regular-WalkSAT/Basic and WalkSAT/Basic on the hard instance of `flat150-4c`

	classical			regular		
	χ^2	d.f.	P	χ^2	d.f.	P
easy	286.49	112	< 0.00001	237.60	99	< 0.00001
median	107.37	95	> 0.19	91.13	86	> 0.33
hard	40.72	58	> 0.95	31.75	38	> 0.75

Table 5.5: Results of the χ^2 test on the easy, median and hard instances of `flat150-4c`

tween the empirical RLD and the predicted exponential distribution. The correspondence is measured by the value of P associated with the value χ^2. The value of P is the probability of obtaining a value for the χ^2 test statistic greater than or equal to the value obtained if the empirical distribution really fits to the distribution $ed[m]$. High values of P indicate a strong correspondence between the two distributions, whereas low values indicate a poor correspondence. Table 5.5 shows the results of the χ^2 test for comparing the RLDs with their corresponding approximations as EDs, for the three instances, with both Regular-WalkSAT/Basic and WalkSAT/Basic. The table also shows, for every χ^2 value, the degrees of freedom (d.f.) of the associated χ^2 distribution for the statistic.

The results of the test show that for both Regular-WalkSAT/Basic and WalkSAT/Basic the level of correspondence of their RLDs with EDs increases with the hardness of the instance. Observe that for the easy instance the value P strongly rejects the fit of the RLD with the ED; for the medium instance the correspondence is very significant; and for the hard instance the value of P is even greater than in the medium one indicating the most strongest correspondence of the RLDs with EDs. This result is consistent with the findings of (Hoos, 1998; Hoos and Stützle, 2000a) where it was shown that the RLDs for local search algorithms for SAT are better approximated with EDs as the hardness of the instance increases. This result indicates that

	vertices	edges	colours
`DSJC125.5.col`	125	4927	17
`DSJC250.5.col`	250	7782	29

Table 5.6: Characteristics of instances `DSJC125.5.col` and `DSJC250.5.col`

this characteristic of local search algorithms for *SAT* is well preserved in their generalizations for *Regular-SAT*, at least for the instances and algorithms considered here. Graphically, we can observe in the figures that the part of the RLD where the algorithms differ with respect to the ED found by the approximation is the part related to the probability of finding solutions with few changes.

The results obtained for flat graph instances clearly show that the *Regular-SAT* encoding outperforms the *SAT* encoding. Let us now see what happens with two well-known graph colouring instances of the DIMACS repository.[3] They were originally used in (Johnson et al., 1991) to compare a number of simulated annealing algorithms (Kirkpatrick et al., 1983) for colouring graphs, and then were used to evaluate local search algorithms for *SAT* (Selman and Kautz, 1993). These graphs, compared to the flat graphs, are very large in terms of the number of edges they have. Table 5.6 shows their number of vertices, number of edges, and the optimum number of colours. We encoded the two instances as *SAT* instances and as *Regular-SAT* instances.

We compared WalkSAT/Basic and Regular-WalkSAT/Basic on the *SAT* and *Regular-SAT* encodings of the DIMACS instances. Each instance was solved 400 times; the noise parameter was approximately optimal, MaxTries was set to 1, and MaxChanges was set to an extremely high value. Table 5.7 shows the mean number of changes and the mean time needed to find a solution for both encodings. As with the flat graphs, both algorithms solved the

[3]ftp://dimacs.rutgers.edu/pub/challenge/graph/benchmarks/colour

		Encoding			
		classical		regular	
		mean	CV	mean	CV
`DSJC125.5.col`	ω	0.13		0.14	
	changes	4827960	0.98	793332	0.89
	time (sec)	470	0.98	161	0.88
`DSJC250.5.col`	ω	0.15		0.186	
	changes	2068166	0.79	513480	0.74
	time (sec)	776	0.81	353	0.76

Table 5.7: Results for DIMACS graph colouring instances

instances in all the runs, indicating that they are PAC for these instances as well. For instance `DSJC125.5.col`, the mean number of changes with Regular-WalkSAT/Basic is about 5 times smaller than with WalkSAT/Basic, and the mean time is about 2.9 times smaller. For instance `DSJC250.5.col`, the mean number of changes with Regular-WalkSAT/Basic is about 4 times smaller than with WalkSAT/Basic, and the mean time is about 2.2 times smaller. Observe that instance `DSJC125.5.col` is harder to solve than instance `DSJC250.5.col` in terms of the number of changes needed but it is easier to solve in terms of the time needed. This is due to the fact that the cost of a change in `DSJC250.5.col` is greater than in `DSJC125.5.col` because `DSJC250.5.col` contains more edges.

To perform a more detailed analysis of the run-time behaviour, we obtained empirical RLDs for WalkSAT/Basic and Regular-WalkSAT/Basic on the two instances. Figure 5.6 shows the RLDs on instance `DSJC125.5.col`, and Figure 5.7 shows the RLDs on instance for `DSJC250.5.col`, as well as their corresponding EDs obtained by the approximation algorithm. As before, Regular-WalkSAT/Basic dominates WalkSAT/Basic.

Table 5.8 shows the mean number of changes per second for the two instances, and the ratio of the mean number of changes per second in the

	changes/second		
	classical	regular	$\frac{\text{classical}}{\text{regular}}$
DSJC125.5.col	10270	4927	2.08
DSJC250.5.col	2665	1454	1.83

Table 5.8: Comparison of changes per second for DIMACS graph colouring instances

	classical			regular		
	χ^2	d.f.	P	χ^2	d.f.	P
DSJC125.5.col	95.85	124	> 0.97	108.86	125	> 0.84
DSJC250.5.col	97.98	80	> 0.08	99.80	82	> 0.08

Table 5.9: Results of the χ^2 test on DIMACS graph colouring instances

algorithm for *SAT* to the mean number of changes per second in the algorithm for *Regular-SAT*. We observe that this ratio does not vary too much between `DSJC125.5.col` and `DSJC250.5.col`.

Table 5.9 shows the results of the χ^2 test. For instance `DSJC125.5.col`, the values of P suggest that the RLD fits strongly to an ED. For instance `DSJC250.5.col`, the values of P suggest that the RLD fits weakly to an ED. This is consistent with the results of the three flat graph instances, where the goodness-of-fit was higher for harder instances. Another common point with the RLDs on the easy and medium flat graph instances is that the difference between the RLD and the approximated ED on instance `DSJC250.5.col` is found in the low part of the RLD. Observe that another factor that indicates that the RLD on `DSJC125.5.col` is more similar to an ED than the RLD on `DSJC250.5.col` is that the CV of the first one is closer to 1 than the CV of the second one, because an ED satisfies that its CV is equal to 1.

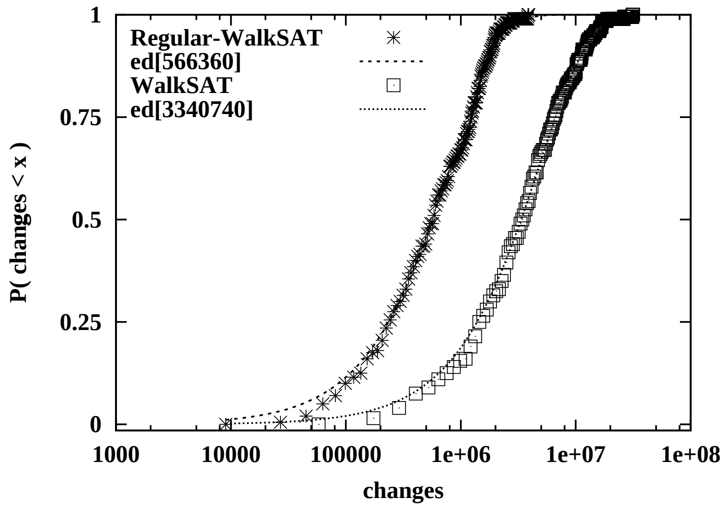


Figure 5.6: RLDs for Regular-WalkSAT/Basic and WalkSAT/Basic on instance `DSJC125.5.col`

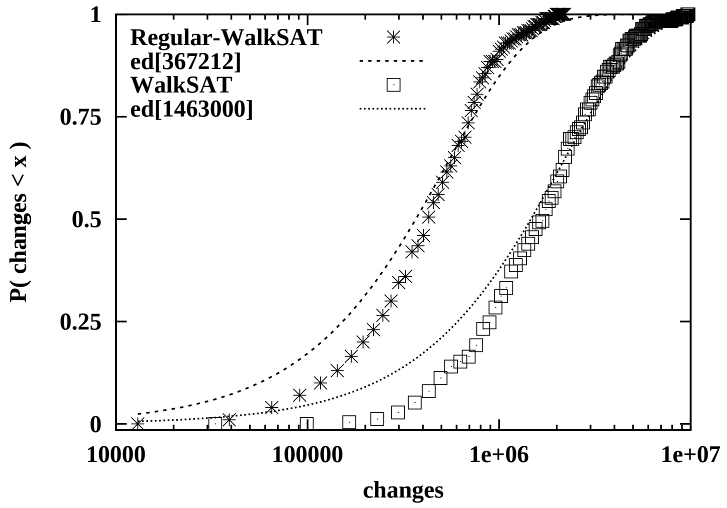


Figure 5.7: RLDs for Regular-WalkSAT/Basic and WalkSAT/Basic on instance `DSJC250.5.col`

		vertices = 100				**vertices = 150**			
		Encoding				**Encoding**			
k		classical		regular		classical		regular	
		mean	CV	mean	CV	mean	CV	mean	CV
3	br. nodes	349	1.05	89	0.88	6182	1.71	597	1.12
	time (sec)	0.70	1.04	0.075	0.90	19	1.57	0.80	1.05
4	br. nodes	133572	1.96	156303	1.97	2457689	1.5	3861096	1.95
	time (sec)	470	1.91	191	1.88	9955	1.3	5920	1.83

Table 5.10: Results of DP and Regular-DP/RH2 on flat graph instances

To our best knowledge, the DIMACS instances have not yet been solved with systematic algorithms for *SAT*. We solved the flat graph instances with DP and Regular-DP/RH2. In this chapter, when we say DP we mean our implementation of Regular-DP/RH2 with $N = \{0, 1\}$; i.e., the Davis-Putnam procedure with the two-sided Jeroslow-Wang branching rule. Table 5.10 shows the mean number of branching nodes per instance and the mean time needed to solve an instance. For 4 colours and 150 vertices, only 10% of instances were solved with DP, and 85% of instances were solved with Regular-DP/RH2; in both cases we used a cutoff of 4 hours, and the results shown correspond to the instances successfully solved. We can conclude that (i) local search algorithms outperform systematic search algorithms on flat graph instances, and (ii) *Regular-SAT* encodings are more well-suited than *SAT* encodings; the differences are impressive for systematic search.

5.3 The round robin problem

In sports scheduling one of the issues is to find a feasible schedule for a sports league that takes into consideration constraints on how the competing teams can be paired, as well as how each team's games are distributed in the entire schedule. Here we consider the timetabling problem for the classic "round robin" schedule: every team must play every other team exactly once. The global nature of the pairing constraints makes this a particularly hard combinatorial search problem.

A game will be scheduled on a certain field at a certain time. This kind of combination will be called a slot. These slots can vary in desirability due to such factors as lateness in the day, the location and the condition of the field, etc. The problem is to schedule the games such that the different fields are assigned to the teams in an equitable manner over the course of the season.

The round robin problem for n teams (`rr`n) is formally defined as follows:

1. There are n teams (n even) and every two teams play against each other exactly once.
2. The season lasts $n - 1$ weeks.
3. Every team plays one game in each week of the season.
4. There are $n/2$ fields and, each week, every field is scheduled for one game.
5. No team plays more than twice in the same field over the course of the season.

The meeting between two teams is called a game and takes place in a slot; i.e., in a particular field in a particular week. Table 5.11 shows a solution for the `rr8` problem; teams are named $1, \ldots, 8$. An `rr`n timetable contains $n(n-1)/2$ slots and slots are filled in with games. A game is represented by a pair of teams (t_1, t_2) with $1 \leq t_1 < t_2 \leq n$. We assume that $t_1 < t_2$ in order to obtain more compact encodings.

The combinatorics of the round robin problem are explosive (McAloon et al., 1997): For an `rr`n league, there are $n/2 \cdot (n-1)$ games (i, j) with $1 \leq i < j \leq n$ to be played. A schedule can be thought of as a permutation of these games. So, for n teams the search space size is

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7
Field 1	$(1,2)$	$(1,3)$	$(5,8)$	$(4,7)$	$(4,8)$	$(2,6)$	$(3,5)$
Field 2	$(3,4)$	$(2,8)$	$(1,4)$	$(6,8)$	$(2,5)$	$(1,7)$	$(6,7)$
Field 3	$(5,6)$	$(4,6)$	$(2,7)$	$(1,5)$	$(3,7)$	$(3,8)$	$(1,8)$
Field 4	$(7,8)$	$(5,7)$	$(3,6)$	$(2,3)$	$(1,6)$	$(4,5)$	$(2,4)$

Table 5.11: An 8-team round robin timetable

$(n/2 \cdot (n-1))!$; i.e., the search space size grows as the factorial of the square of $n/2$.

A related sports scheduling problem that has received increasing interest in the last years is the problem of finding a timetable for the Atlantic Coast Basketball Conference (Nemhauser and Trick, 1998; Walser, 1998).

5.3.1 *SAT* encoding of the round robin problem

The `rr`n problem can be encoded as an instance of *SAT* as follows:

1. The set of propositional variables is

$$\{p_{ij}^{1k} \mid 1 \leq i \leq n/2, 1 \leq j, k \leq n-1\} \cup$$
$$\{p_{ij}^{2k} \mid 1 \leq i \leq n/2, 1 \leq j \leq n-1, 2 \leq k \leq n\},$$

   and its cardinality is $n(n-1)^2$.

   Each slot in the timetable is filled in by a pair of variables $(p_{ij}^{1k_1}, p_{ij}^{2k_2})$. The intended meaning of the pair $(p_{ij}^{1k_1}, p_{ij}^{2k_2})$ is that team k_1 will play against team k_2 in field i in week j.

2. *In each slot, one team plays against another team.* For each slot $(p_{ij}^{1k_1}, p_{ij}^{2k_2})$ $(1 \leq k_1 \leq n-1, 2 \leq k_2 \leq n)$, we define the clauses

$$(p_{ij}^{11} \vee \cdots \vee p_{ij}^{1n-1}) \wedge (p_{ij}^{22} \vee \cdots \vee p_{ij}^{2n})$$

These clauses together with the clauses in (4) ensure that one team plays exactly against another team every week.

3. *In each slot* $(p_{ij}^{1k}, p_{ij}^{2k'})$ *it holds that* $k < k'$. For each two teams k_1, k_2 such that $k_1 \geq k_2$, we define the clause

$$\neg p_{ij}^{1k_1} \vee \neg p_{ij}^{2k_2}$$

4. *Every team plays one game in each week of the season.* For each week j, for each team k, for each two fields i_1, i_2 $(1 \leq i_1, i_2 \leq n/2)$ and for r_1, r_2 $(1 \leq r_1, r_2 \leq 2)$, we define the clause

$$\neg p_{i_1 j}^{r_1 k} \vee \neg p_{i_2 j}^{r_2 k}$$

provided that $p_{i_1 j}^{r_1 k} \neq p_{i_2 j}^{r_2 k}$

In all the steps of the encoding, clauses containing a variable of the form p_{ij}^{21} or p_{ij}^{1n} are not generated.

5. *Every two teams play against each other exactly once.* For each two different slots of the form $(p_{i_1 j_1}^{1k_1}, p_{i_1 j_1}^{2k_2})$ and $(p_{i_2 j_2}^{1k_1}, p_{i_2 j_2}^{2k_2})$ such that $j_1 \neq j_2$ and $k_1 < k_2$, we define the clause

$$\neg p_{i_1 j_1}^{1k_1} \vee \neg p_{i_1 j_1}^{2k_2} \vee \neg p_{i_2 j_2}^{1k_1} \vee \neg p_{i_2 j_2}^{2k_2}$$

The clauses of (5) ensure that every two teams play against each other at most once over the course of the season. Since the total number of slots coincides with the total number of possible games, the above clauses not only ensure that each possible game appears at most in one slot, but exactly once.

6. *No team plays more than twice in the same field over the course of the season.* For each team k, for each field i, for each three different weeks j_1, j_2, j_3 and for each r_1, r_2, r_3 $(1 \leq r_1, r_2, r_3 \leq 2)$, we define the clause

$$\neg p_{ij_1}^{r_1 k} \vee \neg p_{ij_2}^{r_2 k} \vee \neg p_{ij_3}^{r_3 k}$$

The number of clauses of the *SAT* instance obtained with the above encoding for the rr$n$ problem is in $\mathcal{O}(n^6)$. By employing additional variables, it is possible to obtain a *SAT* encoding that produces a formula with a number of clauses which is in $\mathcal{O}(n^4)$. In our expermental investigation we used the encoding that gives rise to a *SAT* instance whose number of clauses is in $\mathcal{O}(n^6)$ because this encoding is more efficient for the satisfiability algorithms used in our experiments.

5.3.2 ***Regular-SAT*** encoding of the round robin problem

The rr$n$ problem can be encoded as an instance of *Regular-SAT* as follows:

1. The truth value set N is $\{1, 2, \ldots, n-1\}$. Each truth value represents either a team from the subset of teams $\{1, 2, \ldots, n-1\}$ or a team from the subset of teams $\{2, \ldots, n\}$.

2. The set of propositional variables is

$$\{p_{ij}^r \mid 1 \leq r \leq 2, 1 \leq i \leq n/2, 1 \leq j \leq n-1\},$$

   and its cardinality is $n(n-1)$.

   Each slot in the timetable is represented by a pair of variables. The pair (p_{ij}^1, p_{ij}^2) refers to the slot corresponding to field i and week j. Since the total number of slots is $n(n-1)/2$, we use $n(n-1)$ variables. Given a satisfying interpretation I, the intended meaning of (p_{ij}^1, p_{ij}^2) is that team $I(p_{ij}^1)$ will play against team $I(p_{ij}^2) + 1$ in field i and week j.

3. *In each slot* (p^1_{ij}, p^2_{ij}) *it holds that* $I(p^1_{ij}) < I(p^2_{ij}) + 1$. For each team $t < n$ and for each slot (p^1_{ij}, p^2_{ij}), we define the regular clause

$$\downarrow (t-1) : p^1_{ij} \vee \uparrow (t+1) : p^1_{ij} \vee \uparrow t : p^2_{ij}$$

   in order to ensure that $I(p^1_{ij}) < I(p^2_{ij}) + 1$. We assume in all the steps that regular literals either of the form $\downarrow 0 : p$ or of the form $\uparrow n : p$ appearing in a regular clause are removed.

4. *Every team plays one game in each week of the season.* For each week j, for each team $t < n$, for each two fields i_1, i_2 $(1 \leq i_1, i_2 \leq n/2)$ and for r_1, r_2 $(1 \leq r_1, r_2 \leq 2)$, we define the clause

$$\downarrow (t_1 - 1) : p^{r_1}_{i_1,j} \vee \uparrow (t_1 + 1) : p^{r_1}_{i_1,j} \vee \downarrow (t_2 - 1) : p^{r_2}_{i_2,j} \vee \uparrow (t_2 + 1) : p^{r_2}_{i_2,j}$$

   where $t_i = t$ if $r_i = 1$ and $t_i = t - 1$ if $r_i = 2$, provided that $p^{r_1}_{i_1,j} \neq p^{r_1}_{i_1,j}$.

5. *Every two teams play against each other exactly once.* For each two different slots $(p^1_{i_1 j_1}, p^2_{i_1 j_1})$ and $(p^1_{i_2 j_2}, p^2_{i_2 j_2})$ such that $j_1 \neq j_2$ and for each possible game (t_1, t_2) such that $t_1 < t_2$, we define the regular clause

$$\begin{array}{l} \downarrow (t_1 - 1) : p^1_{i_1 j_1} \vee \uparrow (t_1 + 1) : p^1_{i_1 j_1} \vee \downarrow (t_2 - 2) : p^2_{i_1 j_1} \vee \uparrow t_2 : p^2_{i_1 j_1} \vee \\ \downarrow (t_1 - 1) : p^1_{i_2 j_2} \vee \uparrow (t_1 + 1) : p^1_{i_2 j_2} \vee \downarrow (t_2 - 2) : p^2_{i_2 j_2} \vee \uparrow t_2 : p^2_{i_2 j_2} \end{array}$$

   in order to ensure that every two teams play against each other exactly once. As with the classical encoding, the above regular clauses not only ensure that each possible game appears at most in one slot, but exactly once.

6. *No team plays more than twice in the same field over the course of the season.* For each team t, for each field i, for each three different weeks j_1, j_2, j_3 and for each r_1, r_2, r_3 $(1 \leq r_1, r_2, r_3 \leq 2)$, we define the clause

$$\begin{array}{l} \downarrow(t_1 - 1) : p^{r_1}_{ij_1} \vee \uparrow(t_1 + 1) : p^{r_1}_{ij_1} \vee \\ \downarrow(t_2 - 1) : p^{r_2}_{ij_2} \vee \uparrow(t_2 + 1) : p^{r_2}_{ij_2} \vee \\ \downarrow(t_3 - 1) : p^{r_3}_{ij_3} \vee \uparrow(t_3 + 1) : p^{r_3}_{ij_3} \end{array}$$

where $t_i = t$ if $r_i = 1$ and $t_i = t - 1$ if $r_i = 2$

The size of the regular CNF formula obtained for the rr$n$ problem is in $\mathcal{O}(n^6)$, although the exact number of clauses is slightly smaller than in the classical encoding. This is because the set of clauses of (2) in the classical encoding is not necessary in the regular encoding, and the number of clauses of (3) is in $\mathcal{O}(n^4)$ in the classical encoding and is in $\mathcal{O}(n^3)$ in the regular encoding.

5.3.3 Experimental results

In this section we report on a series of experiments on the round robin problem performed for comparing the performance of the *SAT* encoding with the performance of the *Regular-SAT* encoding. The *SAT* instances were solved with WalkSAT/G+Tabu and WalkSAT/R-Novelty, and the *Regular-SAT* instances were solved with Regular-WalkSAT/G+Tabu and Regular-WalkSAT/R-Novelty. Such experiments were performed on a Sun Enterprise Ultra-250 Station.

Table 5.12 shows the parameter settings used in the experiments. Instances `rr12`, `rr14` and `rr16` were solved 100 times; the noise parameter was approximately optimal, MaxTries was set to 1, and MaxChanges was set to an extremely high value. The size of the tabu lists ranged from 7 to 8. Instances `rr18` and `rr20` were solved 25 times with the algorithms WalkSAT/G+Tabu and Regular-WalkSAT/G+Tabu using the above parameter settings. For `rr18` and `rr20`, WalkSAT/R-Novelty and

	G+Tabu				**R-Novelty**			
	Encoding				**Encoding**			
	classical		regular		classical		regular	
	tabu	ω	tabu	ω	MaxChanges	ω	MaxChanges	ω
`rr12`	7	0.17	7	0.19	∞	0.09	∞	0.13
`rr14`	8	0.14	8	0.14	∞	0.05	∞	0.112
`rr16`	8	0.128	8	0.128	∞	0.045	∞	0.046
`rr18`	8	0.10	8	0.080	$5 \cdot 10^6$	0.034	$10 \cdot 10^6$	0.023
`rr20`	8	0.07	8	0.054	$9 \cdot 10^6$	0.026	$13 \cdot 10^6$	0.015

Table 5.12: Parameter settings for the round robin instances

Regular-WalkSAT/R-Novelty needed restarts; i.e., they used *finite* values of MaxChanges, and as many tries as necessary.

Table 5.13 shows the mean number of changes and the mean time needed to solve the instances with the *SAT* and the *Regular-SAT* encodings. The results show that the mean number of changes needed to solve an instance is always about 6 times smaller with Regular-WalkSAT/G+Tabu than with WalkSAT/G+Tabu, and about 4 times smaller with Regular-WalkSAT/R-Novelty than with WalkSAT/R-Novelty. The mean time is, however, quite similar in both approaches for all the algorithms, and we do not observe that an algorithm outperforms the others. Thus, the *Regular-SAT*-based approach is comparable in performance with the *SAT*-based approach for the round robin problem.

The two encodings are comparable in performance because the reduction in the mean number of changes is compensated by the decrease in the number of changes per second with the *Regular-SAT* encoding. Table 5.14 shows the mean number of changes per second for the different instances, and the ratio

		G+Tabu Encoding				**R-Novelty Encoding**			
		classical		regular		classical		regular	
		mean	CV	mean	CV	mean	CV	mean	CV
`rr12`	changes	$31 \cdot 10^4$	0.89	49992	0.89	$15 \cdot 10^4$	1.00	19755	1.00
	time (sec)	22	1.00	19	0.89	14	0.99	8	0.89
`rr14`	changes	$12 \cdot 10^5$	0.91	$2 \cdot 10^5$	0.87	$6 \cdot 10^5$	0.98	$1.2 \cdot 10^5$	1.09
	time (sec)	139	0.91	134	0.87	84	0.97	80	1.08
`rr16`	changes	$6 \cdot 10^6$	0.93	$1 \cdot 10^6$	1.14	$6 \cdot 10^6$	1.11	$1 \cdot 10^6$	0.89
	time (sec)	885	0.93	890	1.14	1033	1.11	1068	0.89
`rr18`	changes	$49 \cdot 10^6$	0.99	$6 \cdot 10^6$	0.75	$43 \cdot 10^6$	1.14	$10 \cdot 10^6$	1.10
	time (sec)	9950	0.99	8812	0.73	10248	1.14	11000	1.15
`rr20`	changes	$325 \cdot 10^6$	0.92	$34 \cdot 10^6$	0.79	$326 \cdot 10^6$	1.03	$> 70 \cdot 10^6$	> 1.00
	time (sec)	88961	1.02	73256	0.81	98827	1.03	> 125680	> 1.00

Table 5.13: Results for the round robin instances

	a - changes/second			**b - changes/second**		
	classical	regular	ratio	classical	regular	ratio
`rr12`	13609	2501	5.45	11155	2512	4.44
`rr14`	8453	1503	5.62	7897	1530	5.16
`rr16`	6676	1126	5.93	4955	1026	4.83
`rr18`	4871	693	7.03	4470	856	5.22
`rr20`	3352	464	7.22	3297	560	5.89

Table 5.14: Comparison of changes per second for the round robin instances between WalkSAT/G+Tabu and Regular-WalkSAT/G+Tabu (a) and between WalkSAT/R-Novelty and Regular-WalkSAT/R-Novelty (b)

	classical			regular		
	χ^2	d.f.	P	χ^2	d.f.	P
`rr12`	26.35	23	> 0.28	20.66	26	> 0.75
`rr14`	12.54	30	> 0.99	16.72	25	> 0.89

Table 5.15: Results of the χ^2 test on instances `rr12` and `rr14`

of the mean number of changes per second in the algorithms for *SAT* to the mean number of changes per second in the algorithms for *Regular-SAT*. As in the graph colouring instances, this ratio does not vary too much when the size of the instances is increased, and has always values between 5 and 7 for WalkSAT/G+Tabu and Regular-WalkSAT/G+Tabu, and between 4 and 5 for WalkSAT/R-Novelty and Regular-WalkSAT/R-Novelty.

We selected two instances, `rr12` and `rr14`, for studying in more detail the run-time behaviour of the algorithms Regular-WalkSAT/R-Novelty and WalkSAT/R-Novelty. For obtaining more accurate RLDs, these instances were solved 250 times. Figure 5.8 shows the RLDs on instance

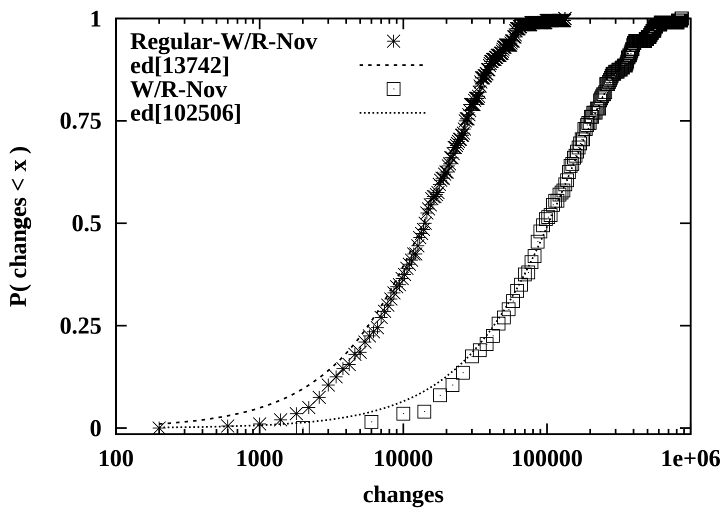


Figure 5.8: RLDs for Regular-WalkSAT/R-Novelty and WalkSAT/R-Novelty on instance `rr12`

`rr12`, and Figure 5.9 shows the RLDs on instance `rr14`. Observe that Regular-WalkSAT/R-Novelty dominates WalkSAT/R-Novelty in both instances. Table 5.15 shows the results of the χ^2 test statistic for the two instances with both algorithms. The values of P indicate that the RLDs fit strongly to their approximated EDs, especially for instance `rr14`.

Regarding systematic algorithms, we were only able to solve instance `rr8` with Satz (Li and Anbulagan, 1997b; Li and Anbulagan, 1997a), the randomized version of Satz (Gomes et al., 1998a), and Regular-DP/RH-2; and we solved instance `rr10` with REL-SAT (Bayardo and Schrag, 1997). We were not able to solve instance `rr12` with systematic algorithms for *SAT* and *Regular-SAT*.

The *scheduling as satisfiability* approach was used to solve the job shop problem in (Crawford and Baker, 1994). To our best knowledge, we first solved the round robin problem using *SAT* and *Regular-SAT* encodings.

To have an idea of the computational difficulty of this problem it is worth mentioning that it was solved for 14 teams using a constraint programming formulation in (McAloon et al., 1997). Then, Gomes et al. (Gomes et al.,

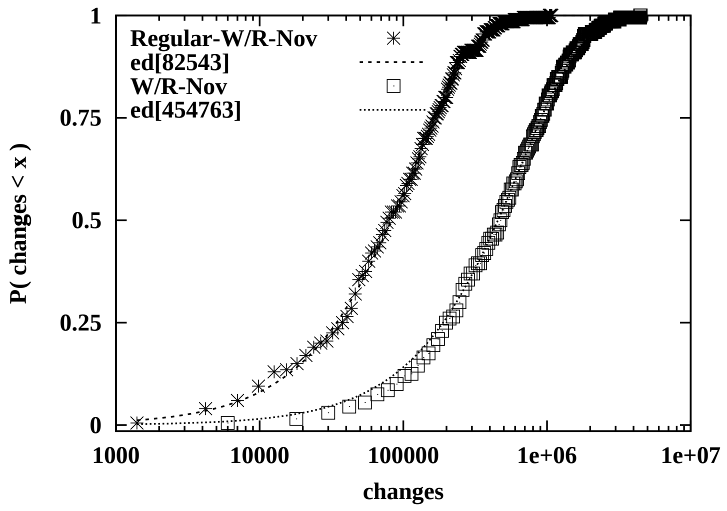


Figure 5.9: RLDs for Regular-WalkSAT/R-Novelty and WalkSAT/R-Novelty on instance `rr14`

1998b) used both an integer programming and a constraint programming formulation. With the former formulation, they were unable to find a solution for 14 teams and they needed 14 hours to find a solution for 12 teams. With the latter formulation and a randomized constraint programming algorithm, they found a solution for 18 teams after 22 hours.

Very recently, we learned that Jean-Charles Régin (personal communication, August 2000) presented results on solving `rr24` using a CSP encoding in (Régin, 1998), and results on solving `rr40` in (Régin, 1999). In (Hamiez and Hao, 2000) there are results for instance `rr40` with a tabu approach, and are also reported Régin's results.

5.4 The all-interval-series problem

The all-interval-series problem is an arithmetic problem first used in (Hoos, 1998) in the context of evaluating stochastic local search algorithms for *SAT*. This problem is inspired by a well-known problem occurring in serial musical composition, which, in its original form, can be stated in the following way:

given the twelve standard pitch-classes ($c, c\#, d, \dots$), represented by numbers $0, 1, \dots, 11$, find a series in which each pitch-class occurs exactly once, and in which the musical intervals between neighbouring notes cover the full set of intervals from the minor second (1 semitone) to the major seventh (11 semitones), i.e., for each of these intervals, there is a pair of neigbhouring pitch-classes in the series, between which this interval appears.

These musical all-interval-series have a certain prominence in serial composition; they can be traced back to Alban Berg and have been extensively studied and used by Ernst Krenek, e.g., in his orchestral piece op.170, "Quaestio temporis".

The problem of finding such series can be easily formulated as an instance of a more general arithmetic problem in $\mathbb{Z}_n$, the set of integer residues modulo n: given a positive integer n, find a vector $s = (s_1, \dots, s_n)$, such that

1. s is a permutation of $\mathbb{Z}_n = \{0, 1, \dots, n-1\}$; and
2. the interval vector $v = (|s_2 - s_1|, |s_3 - s_2|, \dots, |s_n - s_{n-1}|)$ is a permutation of $\mathbb{Z}_n - \{0\} = \{1, 2, \dots, n-1\}$.

A vector v satisfying these conditions is called an all-interval-series of size n; the problem of finding such a series is called the all-interval-series problem of size n and denoted by `ais`n.

Instances of the all-interval-series problem encoded as *SAT* instances are very hard for local search algorithms like GSAT or WalkSAT, but are rather easy for systematic algorithms, such as Satz (Li and Anbulagan, 1997b; Li and Anbulagan, 1997a); at least when considering the *SAT* encoding presented in (Hoos, 1998).

5.4.1 *SAT* encoding of the all-interval-series problem

The all-interval-series problem can be modeled as a CSP in a rather straightforward way. Each element of s and v is represented as a variable; the domains of these variables are $\mathbb{Z}_n$ and $\mathbb{Z}_n - \{0\}$, respectively; and the constraint relations encode the conditions 1 and 2 from the definition given above. From this CSP formulation, instances of the all-interval-series problem can be easily transformed into a *SAT* instance by using essentially the same encoding as for the two previous combinatorial problems, where each assignment of a value to a single CSP variable is represented by a propositional variable and each of the two conditions from the definition above is represented by a set of clauses.

The ais$n$ problem can be encoded as an instance of *SAT* as follows:

1. The set of propositional variables is

   $$\left\{s_i^j \mid i, j \in \mathbb{Z}_n\right\} \cup \left\{v_i^j \mid i, j \in \mathbb{Z}_n - \{0\}\right\}$$

   and its cardinality is $n^2 + (n-1)^2$.

   Each element i of the vector s is filled in by a variable of the form s_i^j, and each element i of the vector v is filled in by a variable v_i^j. The intended meaning of the variable s_i^j is that the element i of the vector s has assigned the value j, and the intended meaning of the variable v_i^j is that the element i of the vector v has assigned the value j.

2. *The vector s is a permutation of $\mathbb{Z}_n$.* For each $i \in \mathbb{Z}_n$ we define the clause

   $$(s_i^0 \vee \cdots \vee s_i^{n-1});$$

   and for each two different values $i_1, i_2 \in \mathbb{Z}_n$ and for each value $j \in \mathbb{Z}_n$, we define the clause

   $$\neg s_{i_1}^j \vee \neg s_{i_2}^j$$

   in order to ensure that the set of variables s_i^j encodes a valid permutation of the set $\mathbb{Z}_n$.

3. *The interval vector v is a permutation of $\mathbb{Z}_n - \{0\}$ and is the interval vector associated with the encoded permutation in the variables s_i^j.*

(a) For each $i \in \mathbb{Z}_n - \{0\}$, we define the clause

$$(v_i^1 \vee \cdots \vee v_i^{n-1});$$

and for each two different values $i_1, i_2 \in \mathbb{Z}_n - \{0\}$, and for each value $j \in \mathbb{Z}_n - \{0\}$, we define the clause

$$\neg v_{i_1}^j \vee \neg v_{i_2}^j$$

in order to ensure that the set of variables v_i^j encodes a valid permutation of the set $\mathbb{Z}_n - \{0\}$.

(b) For each two different values $x, y \in \mathbb{Z}_n$, and for each value $i \in \mathbb{Z}_n - \{0\}$, we define the clause

$$\neg s_{i-1}^x \vee \neg s_i^y \vee v_i^z,$$

where $z = |x-y|$, in order to ensure that the permutation encoded in the variables v_i^j corresponds to the interval vector associated with the permutation encoded in the variables s_i^j.

In the *SAT* encoding of the all-interval-series described in (Hoos, 1998), two additional sets of clauses were used in order to ensure that satisfying interpretations assign exactly one value to each CSP variable. Our *SAT* encoding does not use these two sets because this condition is satisfied without using these sets. Concerning the variables s_i^j, the set of clauses of (2) ensures that the variables s_i^j encode valid permutations of the set $\mathbb{Z}_n$. Observe that a valid permutation has the property that its encoding with the variables s_i^j can never assign two different values to the same variable of the permutation because one of the two values would appear also in another variable of the permutation. The same holds for the variables v_i^j but thanks to the set of clauses of (3-a).

5.4.2 *Regular-SAT* encoding of the all-interval-series problem

The `ais`n problem can be encoded as an instance of *Regular-SAT* as follows:

1. The truth value set N is $\mathbb{Z}_n$. Each truth value represents one of the values that can be assigned to the elements of the vector s or one of the values associated with the interval vector v of s.

2. The set of propositional variables is

$$\{s_i \mid i \in \mathbb{Z}_n\} \cup \{v_i \mid i \in \mathbb{Z}_n - \{0\}\}$$

   and its cardinality is $n + (n-1)$.

   Each element i of the vector s is represented by the variable s_i, and each element i of the interval vector v is represented by the variable v_i. Given a satisfying interpretation I, the intended meaning of the variables s_i and v_i is that the component i of the vector s has assigned the value $I(s_i)$, and that the component i of the interval vector v has assigned the value $I(v_i)$. As the values that can be assigned to elements of the interval vector v are from the set $\mathbb{Z}_n - \{0\}$ but N contains also the value 0, for the variables v_i the truth values 0 and 1 are used indistinctly as the first value of the set $\mathbb{Z}_n - \{0\}$.

3. *The vector s is a permutation of $\mathbb{Z}_n$.* For each two different values $i_1, i_2 \in \mathbb{Z}_n$, and for each value $j \in \mathbb{Z}_n$, we define the regular clause

$$\downarrow(j-1) : s_{i_1} \vee \uparrow(j+1) : s_{i_1} \vee \downarrow(j-1) : s_{i_2} \vee \uparrow(j+1) : s_{i_2}$$

   in order to ensure that the set of variables s_i encodes a valid permutation of the set $\mathbb{Z}_n$. We assume in all the steps that regular literals either of the form $\downarrow\!-1 : p$ or of the form $\uparrow n : p$ appearing in a regular clause are removed.

4. *The interval vector v is a permutation of $\mathbb{Z}_n - \{0\}$ and is the interval vector associated with the encoded permutation in the variables s_i.*

   For each two different values $i_1, i_2 \in \mathbb{Z}_n - \{0\}$, and for each value $j \neq 1 \in \mathbb{Z}_n - \{0\}$, we define the regular clause

   $$\downarrow (j-1) : v_{i_1} \vee \uparrow (j+1) : v_{i_1} \vee \downarrow (j-1) : v_{i_2} \vee \uparrow (j+1) : v_{i_2};$$

   when $j = 1$, we define instead the regular clause

   $$\uparrow 2 : v_{i_1} \vee \uparrow 2 : v_{i_2}$$

   in order to ensure that the set of variables v_i encodes a valid permutation of the set $\mathbb{Z}_n - \{0\}$.

   (a) For each two different values $x, y \in \mathbb{Z}_n$, and for each value $i \in \mathbb{Z}_n - \{0\}$, we define the two regular clauses

   $$\downarrow (x-1) : s_{i-1} \vee \uparrow (x+1) : s_{i-1} \vee \downarrow (y-1) : s_i \vee \uparrow (y+1) : s_i \vee \downarrow z : v_i$$

   $$\downarrow (x-1) : s_{i-1} \vee \uparrow (x+1) : s_{i-1} \vee \downarrow (y-1) : s_i \vee \uparrow (y+1) : s_i \vee \uparrow z : v_i$$

   where $z = |x-y|$, in order to ensure that the permutation encoded in the variables v_i corresponds to the interval vector associated with the permutation encoded in the variables s_i.

The number of clauses in the above encoding is in $\mathcal{O}(n^3)$. Actually, the set of clauses of (4a) has twice the number of clauses of the *SAT* encoding.

5.4.3 Experimental results

In this section we report on a series of experiments on the all-interval-series problem performed for comparing the performance of the *SAT* encoding with the performance of the *Regular-SAT* encoding. The *SAT* instances were solved with WalkSAT/R-Novelty, and the *Regular-SAT* instances were solved with Regular-WalkSAT/R-Novelty.

		Encoding			
		classical		regular	
		mean	CV	mean	CV
`ais12`	ω	0.16		0.32	
	changes	149645	0.99	22368	0.95
	time (sec)	2	0.99	1	0.94
`ais14`	ω	0.10		0.26	
	changes	745899	0.93	81260	1.10
	time (sec)	12	0.93	6	1.08
`ais16`	ω	0.08		0.24	
	changes	4934000	1.04	441556	0.99
	time (sec)	88	1.04	47	0.99
`ais18`	ω	0.06		0.24	
	changes	24129740	0.98	2375661	1.11
	time (sec)	626	0.98	331	1.11

Table 5.16: Results for instances `ais`n

We considered the all-interval-series instances `ais12`, `ais14`, `ais16` and `ais18`. Every instance was solved 100 times; the noise parameter was approximately optimal, MaxTries was set to 1, and MaxChanges was set to an extremely high value. Although both Regular-WalkSAT/R-Novelty and WalkSAT/R-Novelty are not PAC, they always found a solution in every try for all the instances considered. This indicates that these algorithms are PAC for our instances. By contrast, WalkSAT/R-Novelty was shown to be essentially incomplete in (Hoos, 1998) with the encoding defined by Hoos.

Table 5.16 shows the mean number of changes, and the mean time needed to solve the instances with the *SAT* and the *Regular-SAT* encodings. The results show that the mean number of changes and the mean time needed to solve an instance is always smaller when using the *Regular-SAT* encod-

	changes/second		
	classical	regular	$\frac{\text{classical}}{\text{regular}}$
`ais12`	83374	16723	4.97
`ais14`	64680	12671	5.10
`ais16`	56101	9394	5.88
`ais18`	38500	7177	5.36

Table 5.17: Comparison of changes per second for instances `ais`n

	classical			regular		
	χ^2	d.f.	P	χ^2	d.f.	P
`ais12`	14.24	26	> 0.96	18.60	30	> 0.95
`ais14`	18.21	23	> 0.74	17.61	30	> 0.96

Table 5.18: Results of the χ^2 test on instance `ais12` and instance `ais14`

ing than when using the *SAT* encoding. The difference in the number of changes increases from about 7 times smaller in `ais12` to about 10 times smaller in `ais18` and the mean time is always about 2 times smaller with the *Regular-SAT* encoding, similarly to what happens with the flat graphs.

Table 5.17 shows the mean number of changes per second for the different instances, and the ratio of the mean number of changes per second in the algorithm for *SAT* to the mean number of changes per second in the algorithm for *Regular-SAT*. As for the previous problems, this ratio does not vary too much when the size of the instances is increased.

We selected two instances, `ais12` and `ais14`, for studying in more detail the run-time behaviour of the algorithms Regular-WalkSAT/R-Novelty and WalkSAT/R-Novelty. For obtaining more accurate RLDs, these in-

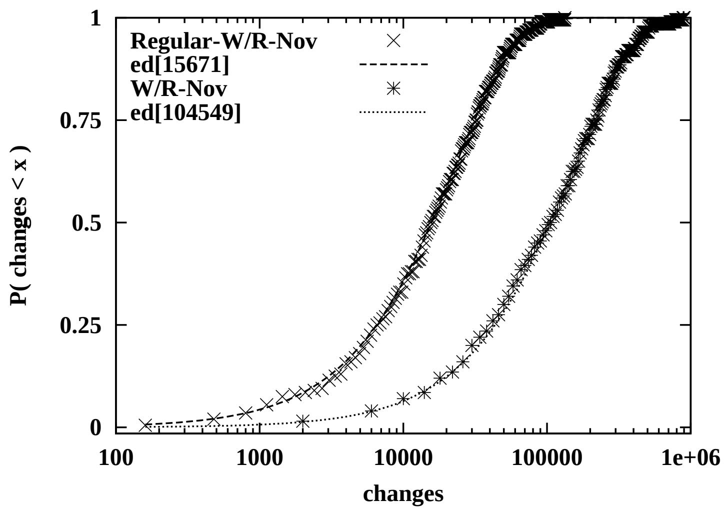


Figure 5.10: RLDs for Regular-WalkSAT/R-Novelty and WalkSAT/R-Novelty on instance `ais12`

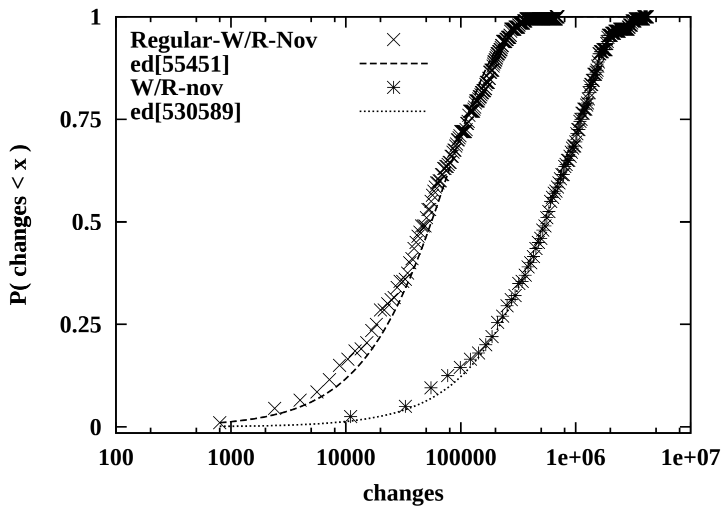


Figure 5.11: RLDs for Regular-WalkSAT/R-Novelty and WalkSAT/R-Novelty on instance `ais14`

stances were solved 250 times. Figure 5.10 shows the RLDs on instance `ais12`, and Figure 5.11 shows the RLDs on instance `ais14`. Observe that Regular-WalkSAT/R-Novelty dominates WalkSAT/R-Novelty in both instances. Table 5.18 shows the results of the χ^2 test statistic for the two instances with both algorithms. The values of P indicate that the RLDs fit strongly to their approximated EDs.

Chapter 6

Conclusions

In the preceding chapters we have been concerned with the design, implementation and analysis of systematic (Regular-DP) and local search (Regular-GSAT and Regular-WalkSAT) algorithms for *Regular-SAT*, and we have defined a new and competitive generic problem solving approach which consists in modeling hard combinatorial problems as *Regular-SAT* instances and then solving the resulting encodings with algorithms for *Regular-SAT*.

We have paid special attention to the definition of suitable data structures for representing formulas, heuristics for efficiently exploring the search space, strategies for escaping from local minima in local search algorithms, and suitable encodings for modeling combinatorial problems as *Regular-SAT* instances. Our implementations have allowed us to put our ideas into practice and, from this experience, we have gained new insights for developing new procedures and improving existing ones.

In Chapter 1, we have already given in detail the main contributions of this thesis. Here, we summarize them briefly:

- Design and implementation of a variant of Regular-DP, equipped with a new branching heuristic.

- Design and implementation of two new families of local search algorithms for *Regular-SAT*: Regular-GSAT and Regular-WalkSAT.

- Construction of the the first benchmark library, formed a representative sample of hard instances, for analysing and comparing systematic and local search algorithms for *Regular-SAT*.
- Experimental evidence of the existence of phase transitions in *Regular Random 3-SAT*, as well as study of the change of the location of the threshold as the cardinality of the truth value set is varied.
- Empirical study of the generic problem solving approach which consists in modeling hard combinatorial problems as *Regular-SAT* instances and then solving the resulting encodings with algorithms for *Regular-SAT*. Within this empirical study, we have provided evidence that, at least for the combinatorial problems studied, our problem solving approach can outperform or compete with state-of-the-art problem solving approaches.

Finally, we would like to point out some future research perspectives:

- Definition and analysis of further branching heuristics and branching rules for Regular-DP, as well as simplification techniques to avoid redundant and useless computations during the exploration of the search space.
- Definition and analysis of new strategies for escaping from local minima in Regular-GSAT and Regular-WalkSAT.
- Design and implementation of a declarative language for modeling combinatorial problems, and design and implementation of translators to produce classical, regular and signed CNF formulas from the problem specification.
- Conducting an experimental investigation to deepen our understanding of *SAT* and *Regular-SAT* encodings so that they can be applied more effectively to solve combinatorial problems. For instance, we would like

to identify design principles which make an encoding successful; and to understand how encoding components, problem characteristics and good performance for particular satisfiability algorithms are related.

- Application of our problem solving approach to optimization problems.
- Finding further application areas (e.g. bioinformatics, planning, and timetabling) for our problem solving approach.
- Creation and maintenance of a web site —similar to SATLIB— with valuable resources for researchers of *Regular-SAT* containing benchmarks, solvers, comparisons, a bibliography, and links to researchers and events. The benchmark library should facilitate the use of the same set of problem instances across different empirical studies and the comparison of experimental results.
- Extension of the results obtained for totally-ordered regular CNF formulas to both partially-ordered regular CNF formulas and signed CNF formulas.

Appendix A

Format of Regular CNF Formulas

The regular satisfiability algorithms described in this thesis read a regular CNF formula from a file and then determine whether the formula is satisfiable. In this section we describe the format that a file containing a regular CNF formula must have. Such a file format is an adaptation of the file format defined by the DIMACS Challenge Committee for classical CNF formulas (DIM, 1993). The DIMACS format is currently accepted by almost all of the best-performing SAT solvers and tools, is simple to parse and generate, reasonably concise and flexible, portable across different platforms, and human-readable (Hoos and Stützle, 2000b). In addition to the information included in the classical setting, we must also include the cardinality of the truth value set and the value of the sign of regular literals.

Without loss of generality, given a regular CNF formula over a particular truth value set N, we will take the set of natural numbers $\{1, 2, \cdots, |N|\}$ as the truth value set of that formula. We can establish an isomorphism between the particular set N and the set $\{1, 2, \cdots, |N|\}$ that preserves the total ordering relation associated with regular literals. For that reason, we specify the cardinality of the truth value set instead of the set itself.

The file has two parts, the preamble and the body. The preamble, which contains information about the formula, is followed by the body, which contains the regular clauses occurring in the formula. We can put any number of carriage returns between the preamble and the body.

The preamble is formed by comment lines and the problem line. Comment lines give information about the formula and are ignored by the program. Each comment line begins with the character `'c'` and ends with a carriage return. The problem line contains the parameters that allow to correctly process the formula: the cardinality of the truth value set ($|N|$), the number of propositional variables (V) and the number of clauses (C). The syntax of the problem line is as follows:

p *format* $|N|$ V C

The letter 'p' indicates that this is the problem line. The field *format* contains a string that identifies the type of logical formula under consideration; in our case, that string must be *regcnf*, which stands for regular CNF formula. This way of identifying the kind of formula follows directly from the recommendations of the DIMACS Challenge Committee.

The body part begins after the problem line and is formed by a sequence of regular clauses. A regular clause is specified by giving the sequence of its regular literals. A regular literal is specified by two integer numbers separated by either spaces or tabulators or newline characters. The first integer specifies the regular sign of the literal. Given a positive regular sign of the form $\uparrow a$, where a is an integer from 1 to $|N|$, we represent this sign by the positive integer a, and given a negative regular sign of the form $\downarrow a$, we represent this sign by the negative integer $-a$. We use the integers from 1 to $|N|$ to denote the elements of the truth value set, having in mind that there is an implicit isomorphism between the elements of the real truth value set and the set $\{1, 2, \cdots, |N|\}$. The second integer specifies the variable of the literal. It must be a positive number between 1 and V because the different variables appearing in the literals of the formula are assumed to be numbered consecutively. We separate the distinct regular literals of the clause by either spaces or tabulators or newline characters and we terminate each clause with the number 0.

Example A.1 *Let* $N = \{0, \frac{1}{3}, \frac{2}{3}, 1\}$ *and let* Γ *be the following regular CNF formula:*

$$\begin{aligned}\Gamma \;=\; & \{\{\downarrow 0 : p_1, \uparrow 1 : p_3, \uparrow 1 : p_4\}, \{\downarrow \tfrac{1}{3} : p_1, \downarrow \tfrac{1}{3} : p_3, \uparrow 1 : p_4\}, \\ & \{\downarrow 0 : p_1, \downarrow \tfrac{1}{3} : p_3, \downarrow \tfrac{1}{3} : p_4\}, \{\uparrow 1 : p_1, \downarrow 0 : p_5\}, \\ & \{\uparrow \tfrac{2}{3} : p_1, \uparrow \tfrac{1}{3} : p_5\}, \{\uparrow 1 : p_3, \downarrow \tfrac{1}{3} : p_4\}\}\end{aligned}$$

The representation of Γ *in a file should contain the following information:*

```
c
c   File for the above regular CNF formula
c

p regcnf 4 5 6

-1 1   4 3  4 4  0
-2 1  -2 3  4 4  0
-1 1  -2 3 -2 4  0
 4 1  -1 5       0
 3 1   2 5       0
 4 3  -2 4       0
```

References

Aspvall, R., Plass, M. and Tarjan, R. (1979). A linear time algorithm for testing the truth of certain quantified boolean formulae. *Information Processing Letters*, 8(3):121–123.

Bayardo, R. J. and Schrag, R. C. (1997). Using CSP look-back techniques to solve real-world SAT instances. In *Proceedings of the 14th National Conference on Artificial Intelligence, AAAI'97, Providence/RI, USA*, pages 203–208. AAAI Press.

Beckert, B., Hähnle, R. and Manyà, F. (1999). Transformations between signed and classical clause logic. In *Proceedings, 29th International Symposium on Multiple-Valued Logics (ISMVL), Freiburg, Germany*, pages 248–255. IEEE Press, Los Alamitos.

Beckert, B., Hähnle, R. and Manyà, F. (2000a). The 2-SAT problem of regular signed CNF formulas. In *Proceedings, 30th International Symposium on Multiple-Valued Logics (ISMVL), Portland/OR, USA*, pages 331–336. IEEE CS Press, Los Alamitos.

Beckert, B., Hähnle, R. and Manyà, F. (2000b). The SAT problem of signed CNF formulas. In Basin, D., D'Agostino, M., Gabbay, D., Matthews, S. and Viganò, L., editors, *Labelled Deduction*, volume 17 of *Applied Logic Series*, pages 61–82. Kluwer, Dordrecht.

Béjar, R. and Manyà, F. (1998). Phase transitions in the regular random

3-SAT problem. In *Proceedings, 1r Congrés Català d'Intel.ligència Artificial, CCIA'98, Tarragona, Spain*, pages 39–45.

Béjar, R. and Manyà, F. (1999a). A comparison of systematic and local search algorithms for regular CNF formulas. In *Proceedings of the 5th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, ECSQARU'99, London, England*, pages 22–31. Springer LNAI 1638.

Béjar, R. and Manyà, F. (1999b). Phase transitions in the regular random 3-SAT problem. In *Proceedings of the 11th International Symposium on Methodologies for Intelligent Systems, ISMIS'99, Warsaw, Poland*, pages 292–300. Springer LNAI 1609.

Béjar, R. and Manyà, F. (1999c). Solving combinatorial problems with regular local search algorithms. In *Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning, LPAR'99, Tbilisi, Republic of Georgia*, pages 33–43. Springer LNAI 1705.

Béjar, R. and Manyà, F. (2000). Solving the round robin problem using propositional logic. In *Proceedings of the 17th National Conference on Artificial Intelligence, AAAI-2000, Austin/TX, USA*, pages 262–266. AAAI Press.

Cheeseman, P., Kanefsky, B. and Taylor, W. M. (1991). Where the really hard problems are. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'91*, pages 331–337. Morgan Kaufmann.

Cook, S. (1971). The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158.

Cook, S. and Mitchell, D. G. (1997). Finding hard instances of the satisfiability problem: A survey. In Du, D., Gu, J. and Pardalos, P., editors, *Satisfiability Problem: Theory and Applications*, volume 35 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science.*

Crawford, J. M. and Baker, A. B. (1994). Experimental results on the application of satisfiability algorithms to scheduling problems. In *Proceedings of the 12th National Conference on Artificial Intelligence, AAAI'94, Seattle/WA, USA*, pages 1092–1097. AAAI Press.

Crawford, J. M. and Auton, L. D. (1996). Experimental results on the crossover point in random 3-SAT. *Artificial Intelligence*, 81:31–57.

Culberson, J. C. and Luo, F. (1996). Exploring the k-colorable landscape with iterated greedy. In Johnson, D. S. and Trick, M. A., editors, *Cliques, Coloring and Satisfiability*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science.*

Davis, M. and Putnam, H. (1960). A computing procedure for quantification theory. *Journal of the ACM*, 7(3):201–215.

Davis, M., Logemann, G. and Loveland, D. (1962). A machine program for theorem-proving. *Communications of the ACM*, 5:394–397.

del Val, A. (2000). On 2-SAT and renamable Horn. In *Proceedings of the 17th National Conference on Artificial Intelligence, AAAI-2000, Austin/TX, USA*, pages 279–284. AAAI Press.

DIMACS Challenge Committee (1993). *Satisfiability Suggested Format.* ftp://dimacs.rutgers.edu.

Dowling, W. and Gallier, J. (1984). Linear-time algorithms for testing the satisfiability of propositional Horn formulæ. *Journal of Logic Programming*, 3:267–284.

Escalada-Imaz, G. (1989a). *Optimisation d'Algorithmes d'Inference Monotone en Logique des Propositions et du Premier Ordre.* PhD thesis, Université Paul Sabatier, Toulouse, France.

Escalada-Imaz, G. (1989b). Un algorithme de complexité quadratique et un algorithme de complexité lineaire pour la 2-satisfiabilité. Technical Report 89378, LAAS, Toulouse.

Escalada-Imaz, G. and Manyà, F. (1994). The satisfiability problem for multiple-valued Horn formulæ. In *Proceedings, International Symposium on Multiple-Valued Logics, ISMVL'94, Boston/MA, USA*, pages 250–256. IEEE Press, Los Alamitos.

Even, S., Itai, A. and Shamir, A. (1976). On the complexity of timetable and multicommodity flow problems. *SIAM J. Computing*, 5:691–703.

Freeman, J. W. (1995). *Improvements to Propositional Satisfiability Search Algorithms*. PhD thesis, Department of Computer and Information Science, University of Pennsylvania, PA, USA.

Ghallab, M. and Escalada-Imaz, G. (1991). A linear control algorithm for a class of rule-based systems. *Journal of Logic Programming*, 11:117–132.

Glover, F. (1989). Tabu search – Part I & II. *ORSA Journal on Computing*, 1/2(3/1):190–206/4–32.

Glover, F. and Laguna, M. (1997). *Tabu Search*. Kluwer Academic Publishers.

Gomes, C. P., Selman, B. and Kautz, H. (1998a). Boosting combinatorial search through randomization. In *Proceedings of the 15th National Conference on Artificial Intelligence, AAAI'98, Madison/WI, USA*, pages 431–437. AAAI Press.

Gomes, C. P., Selman, B., McAloon, K. and Tretkoff, C. (1998b). Randomization in backtrack search: Exploiting heavy-tailed profiles for solving hard scheduling problems. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems, AIPS'98, Pittsburg/PA, USA*. AAAI Press.

Hähnle, R. (1994a). *Automated Deduction in Multiple-Valued Logics*, volume 10 of *International Series of Monographs in Computer Science.* Oxford University Press.

Hähnle, R. (1994b). Short conjunctive normal forms in finitely-valued logics. *Journal of Logic and Computation*, 4(6):905–927.

Hähnle, R. (1996). Exploiting data dependencies in many-valued logics. *Journal of Applied Non-Classical Logics*, 6:49–69.

Hähnle, R. and Escalada-Imaz, G. (1997). Deduction in many-valued logics: A survey. *Mathware and Soft Computing*, 4(2):69–97.

Hähnle, R. (to appear, 2001). Advanced many-valued logics. In Gabbay, D., editor, *Handbook of Philosophical Logic*, volume 2, chapter 5. Kluwer, Dordrecht, second edition.

Hamiez, J. and Hao, J. (2000). Solving the sports league scheduling problem with tabu search. In *Workshop on Local Search for Planning and Scheduling, European Conference on Artificial Intelligence (ECAI-2000), Berlin, Germany.*

Hooker, J. N. and Vinay, V. (1995). Branching rules for satisfiability. *Journal of Automated Reasoning*, 15:359–383.

Hoos, H. H. (1998). *Stochastic Local Search – Methods, Models, Applications.* PhD thesis, Department of Computer Science, Darmstadt University of Technology.

Hoos, H. H. (1999a). On the run-time behaviour of stochastic local search algorithms for SAT. In *Proceedings of the 16th National Conference on Artificial Intelligence, AAAI'99*, pages 661–666. AAAI Press.

Hoos, H. H. (1999b). SAT-encodings, search space structure, and local search performance. In *Proceedings of the International Joint Conference on Ar-*

tificial Intelligence, IJCAI'99, Stockholm, Sweden, pages 296–303. Morgan Kaufmann.

Hoos, H. H. and Stützle, T. (2000a). Local search algorithms for SAT: An empirical evaluation. *Journal of Automated Reasoning*, 24(4):421–481.

Hoos, H. H. and Stützle, T. (2000b). SATLIB: An online resource for research on SAT. In Gent, I., van Maaren, H. and Walsh, T., editors, *SAT2000. Highlights of Satisfiability Research in the Year 2000*, volume 63 of *Frontiers in Artificial Intelligence and Applications*, pages 283–292. IOS Press.

Johnson, D. S., Aragon, C. R., McGeoch, L. A. and Schevon, C. (1991). Optimization by simulated annealing: An experimental evaluation; part II, graph coloring and number partitioning. *Operations Research*, 39(3):378–406.

Kautz, H. A. and Selman, B. (1996). Pushing the envelope: Planning, propositional logic, and stochastic search. In *Proceedings of the 14th National Conference on Artificial Intelligence, AAAI'96, Portland/OR, USA*, pages 1194–1201. AAAI Press.

Kautz, H. A. and Selman, B. (1999). Unifying SAT-based and graph-based planning. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'99, Stockholm, Sweden*, pages 318–325. Morgan Kaufmann.

Kifer, M. and Subrahmanian, V. S. (1992). Theory of generalized annotated logic programming and its applications. *Journal of Logic Programming*, 12:335–367.

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

Kirousis, L. M., Kranakis, E. and Krizanc, D. (1996). Approximating the unsatisfiability threshold of random formulas. In *Proceedings of the 4th An-*

nual European Symposium on Algorithms, ESA'96, Barcelona, Spain, pages 27–38. Springer LNCS 1136.

Kirousis, L. M., Kranakis, E., Krizanc, D. and Stamatiou, Y. C. (1998). Aproximating the unsatisfiability threshold of random formulas. *Random Structures and Algorithms*, 12(3):253–269.

Li, C. M. and Anbulagan, (1997a). Heuristics based on unit propagation for satisfiability problems. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'97, Nagoya, Japan*, pages 366–371. Morgan Kaufmann.

Li, C. M. and Anbulagan, (1997b). Look-ahead versus look-back for satisfiability problems. In *Proceedings of the 3rd International Conference on Principles of Constraint Programming, CP'97, Linz, Austria*, pages 341–355. Springer LNCS 1330.

Li, C. M. (2000). Integrating equivalence reasoning into Davis-Putnam procedure. In *Proceedings of the 17th National Conference on Artificial Intelligence, AAAI-2000, Austin/TX, USA*, pages 291–296. AAAI Press.

Lu, J. J., Murray, N. V. and Rosenthal, E. (1993). Signed formulas and annotated logics. In *Proceedings 23st International Symposium on Multiple-Valued Logic*, pages 48–53. IEEE Computer Society Press, Los Alamitos.

Lu, J. J., Murray, N. V. and Rosenthal, E. (1998). A framework for automated reasoning in multiple-valued logics. *Journal of Automated Reasoning*, 21(1):39–67.

Manyà, F. (1996). *Proof Procedures for Multiple-Valued Propositional Logics.* PhD thesis, Universitat Autònoma de Barcelona. Published in (Manyà, 1999).

Manyà, F., Béjar, R. and Escalada-Imaz, G. (1998). The satisfiability problem in regular CNF-formulas. *Soft Computing: A Fusion of Foundations, Methodologies and Applications*, 2(3):116–123.

Manyà, F. (1999). *Proof Procedures for Multiple-Valued Propositional Logics.* Number 9 in Monografies de l'Institut d'Investigació en Intel.ligència Artificial. IIIA-CSIC, Bellaterra (Barcelona).

Manyà, F. (2000). The 2-SAT problem in signed CNF formulas. *Multiple-Valued Logic. An International Journal*, 5(4):307–325.

Massacci, F. and Marraro, L. (2000). Logical cryptanalysis as a SAT-problem. *Journal of Automated Reasoning*, 24(1/2):165–203.

McAllester, D., Selman, B. and Kautz, H. (1997). Evidence for invariants in local search. In *Proceedings of the 14th National Conference on Artificial Intelligence, AAAI'97, Providence/RI, USA*, pages 321–326. AAAI Press.

McAloon, K., Tretkoff, C. and Wetzel, G. (1997). Sports league scheduling. In *Proceedings of the 1997 ILOG Optimization Suite International Users' Conference, Paris, France.*

Minoux, M. (1988). LTUR: A simplified linear time unit resolution algorithm for Horn formulae and its computer implementation. *Information Processing Letters*, 29:1–12.

Mitchell, D., Selman, B. and Levesque, H. (1992). Hard and easy distributions of SAT problems. In *Proceedings of the 10th National Conference on Artificial Intelligence, AAAI'92, San Jose/CA, USA*, pages 459–465. AAAI Press.

Nemhauser, G. L. and Trick, M. A. (1998). Scheduling a major college basketball conference. *Operations Research*, 46(1):1–8.

Press, W. H., Flannery, B. P., Teukolsky, S. A. and Vetterling, W. T. (1988). *Numerical Recipes in C.* Cambridge University Press.

Régin, J. (1998). Modeling and solving sports league scheduling with constraint programming. In *Proceedings, 1er Congrès de la Societé Française de Recherche Opérationalle et Aide á la Décision, Paris, France.*

Régin, J. (1999). Sports scheduling and constraint programming. In *INFORMS, Cincinnati, Ohio.*

Schuurmans, D. and Southey, F. (2000). Local search characteristics of incomplete SAT procedures. In *Proceedings of the 17th National Conference on Artificial Intelligence, AAAI-2000, Austin/TX, USA*, pages 297–302. AAAI Press.

Scutellà, M. (1990). A note on Dowling and Gallier's algorithm for propositional Horn satisfiability. *Journal of Logic Programming*, 8:265–273.

Selman, B., Levesque, H. and Mitchell, D. (1992). A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference on Artificial Intelligence, AAAI'92, San Jose/CA, USA*, pages 440–446. AAAI Press.

Selman, B. and Kautz, H. A. (1993). Domain-independent extensions of GSAT: Solving large structured satisfiability problems. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'93, Chambery, France*, pages 290–295. Morgan Kaufmann.

Selman, B., Kautz, H. A. and Cohen, B. (1994). Noise strategies for improving local search. In *Proceedings of the 12th National Conference on Artificial Intelligence, AAAI'94, Seattle/WA, USA*, pages 337–343. AAAI Press.

Sofronie-Stokkermans, V. (1998). On translation of finitely-valued logics to classical first-order logic. In *Proceedings, 13th European Conference on Artificial Intelligence, ECAI'98, Brighton, UK*, pages 410–411. John Wiley and Sons.

Sofronie-Stokkermans, V. (2000). Automated theorem proving by resolution for finitely-valued logics based on distributive lattices with operators. *Multiple-Valued Logic. An International Journal.* In Press.

Walser, J. P. (1998). *Domain-Independent Local Search for Linear Integer Optimization.* PhD thesis, Universität des Saarlandes, Saarbrücken, Germany.

Warners, J. P. (1999a). *Nonlinear Approaches to Satisfiability Problems.* PhD thesis, Faculty of Information Technology and Systems, Delft University of Technology, Holland.

Warners, J. P. and van Maaren, H. (1999b). A two phase algorithm for solving a class of hard satisfiability problems. *Operations Research Letters*, 23(3–5):81–88.

Index