Experimental Results on the Satisfiable Core in Random 3SAT

Honglei Zeng[1] and Sheila McIlraith[2]

[1] Department of Computer Science, Knowledge Systems, AI Laboratory, Stanford University, California, USA 94305,
hlzeng@cs.stanford.edu
[2] Department of Computer Science, University of Toronto, Toronto, Canada,
sheila@cs.toronto.edu

Abstract. Given a satisfiable k-CNF SAT instance, a *satisfiable core* is a minimal subset of the k-CNF clauses that preserves all and only the satisfying assignments of the original instance. In this paper, we extend the previous results on satisfiable core, especially on the strong correlation between the hardness of SAT instances and the size of their satisfiable cores. We introduce a measure called the weighted clause-to-variable ratio, which substantially improves on the classic clause-to-variable ratio in explaining the phase transition. We also examine interesting transitions in satisfiable core size of random instances and show that satisfiable core is a powerful concept for studying the constrainedness of instances.

1 Introduction

In this paper, we introduce the notion of satisfiable cores of satisfiable SAT instances, and experimentally analyze their properties with the view to better characterizing the nature of hard and easy random 3SAT instances. Informally, a *satisfiable core* of a satisfiable instance is a minimal subset of clauses from the original SAT instance that preserve all and only its satisfying assignments. Any strict subset of the satisfiable core has satisfying assignment(s) that do not hold in the original SAT instance. We can generate a satisfiable core by removing *redundant clauses* from the SAT instance. We define a redundant clause to be one that may be removed from a CNF instance without altering the satisfying assignments of that instance.

The term satisfiable core is the analogue of the notion of an *unsatisfiable core*, which has been used extensively to study properties of unsatisfiable SAT instances (e.g., [1][2][3]). Nevertheless, little has been done in studying satisfiable cores. The most significant work we are aware of is the redundancy study in random SAT formulas by Boufkhad and Roussel, who in [4] formally define the notions of redundant clauses and irredundant formulas (formulas without redundant clauses). Their two key findings are (1) irredundant formulas are typically much harder to solve than their counterpart redundant formulas; (2) redundancy in random formulas decreases as the number of variables increases.

Satisfiable core and irredundant formula are similar concepts for different purposes: irredundant formula provides a redundancy characterization of a formula, while satisfiable core is defined to study the substructures of SAT instances.

Our interest in extending the results of Boufkhad and Roussel is several-fold. We are interested in better understanding the nature of hard and easy SAT instances and the parameters that affect hardness. We are also interested in predicting the hardness of instances and whether the size of a satisfiable core provides a reasonable prediction. Finally, we are motivated by clause learning [5], which in recent years, has become the driving force in improving the performance of many SAT solvers, such as Siege [5] and zChaff [6]. Clause learning typically works by augmenting the original CNF with implied clauses that are learned from resolution conflicts during the DPLL search procedure. Such implied clauses are intimately related to the notion of satisfiable cores because adding implied clauses to a satisfiable core, like redundant clauses, does not affect the original satisfying assignments.

In Section 2, we introduce satisfiable cores for SAT instances. In Section 3, we propose a definition for weighted clause-to-variable ratio, WCV, basing it on the notion of satisfiable core. WCV substantially improves the classic clause-to-variable (m/n) ratio in explaining the phase transition. In Section 4, we compare WCV to a number of interesting ideas such as backbone and backdoor. We conclude the paper in Section 5 with a summary and discussion of future work. While this paper does not fully explain the phenomena being observed experimentally, it shows, in the notion of satisfiable cores, an important concept in the arsenal of tools used to characterize and generate hard SAT instances.

2 Redundant Clauses

2.1 Definitions

In this paper, we restrict our attention to satisfiable 3SAT instances without loss of generality. We introduce the notion of redundant clause, a convenient term that not only eases our discussion of satisfiable core but also deepens our understanding. Following convention, the variable m denotes the number of clauses in an instance and n denotes the number of variables.

Intuitively, a redundant clause is one that may be removed from a CNF instance without altering the satisfying assignments of that instance. We use the adjective *redundant* because these clauses do not contribute anything further to the defining of the set of solutions.

Definition 1 (Redundant Clause (RC)). *Let Σ be a set of clauses that includes clause c. Let Σ' be Σ with clause c removed. c is a redundant clause relative to Σ iff $\Sigma' \models c$.*

We now define the notion of satisfiable core. Given a set of clauses Σ, a satisfiable core of Σ satisfies three conditions: (1) is a subset of these clauses; (2) has exactly the same satisfying assignments; (3) none of the clauses in a

satisfiable core is redundant (i.e. none of the strict subset of a satisfiable core is still a satisfiable core). Note that the satisfiable core of Σ may not be unique.

Definition 2 (Satisfiable Core). *Given a set of clauses Σ, Σ' is a satisfiable core of Σ iff (i) $\Sigma' \subseteq \Sigma$; (ii) for every truth assignment $\mathcal{M}$ of Σ', if $\mathcal{M} \models \Sigma'$ then $\mathcal{M} \models \Sigma$; and (iii) $\forall c_i \in \Sigma', c_i$ is not a redundant clause relative to Σ'.*

We define the clauses in a satisfiable core to be prime clauses, in contrast to redundant clauses. Given an instance Σ and an associated satisfiable core, Σ', the size of the satisfiable core, or the *number of prime clauses* in the core (NPC) is simply the cardinality of Σ', $|\Sigma'|$. The *number of redundant clauses* (NRC) is defined to be $NRC = m - NPC$. Since a satisfiable core is not unique, a prime clause of one core may be redundant relative to another core. Further, Σ' is a minimal satisfiable core if there is no set Σ'', such that Σ'' is also a satisfiable core relative to Σ and $|\Sigma''| < |\Sigma'|$.

We can use any efficient complete SAT solver to identify redundant clauses. We do so by removing the clause from the instance, conjoining its negation and trying to determine satisfiability. If the resultant formula is unsatisfiable then the clause is entailed by the original formula.

Proposition 1 (Determination of Redundancy). *Given a clause c and an instance Σ, determining whether c is a redundant clause relative to Σ is as difficult as SAT.*

Determination of Redundant Clause:
PROBLEM: Given a set of clauses Σ, $c \in \Sigma$, determine if c is a redundant clause relative to Σ.
ALGORITHM If $\Sigma \backslash c \ \cup \neg c$ is unsatisfiable, then $\Sigma \backslash c \models c$ and c is a redundant clause.

To compute a satisfiable core for a (satisfiable) instance Σ, we randomly choose a clause and test if it is redundant. If it is, we delete it. We test each clause once. The resultant set is a satisfiable core.

Determination of Satisfiable Core
PROBLEM: Given a set of clauses $\Sigma = \{c_i\}, i = 1, \ldots, m$, find a satisfiable core of Σ.
ALGORITHM:
1. $CC \leftarrow \{\}$.
2. While $CC \neq \Sigma$, do
    a. Randomly choose $c_i \in \Sigma$ st $c_i \notin CC$.
    b. If c_i is a redundant clause then $\Sigma \leftarrow \Sigma \backslash c_i$, else $CC \leftarrow CC \cup c_i$.

The resulting Σ is a satisfiable core.

Unfortunately, the determination of a satisfiable core is NP-hard; though there are a number of ways to improve this brute-force algorithm, we are still

severely limited by our computational power. The satisfiable cores we generated are neither unique nor minimal, but nevertheless sufficient for the needs of studying their properties.

2.2 Experiments with Redundant Clauses

In this section we report the degree of redundancy in random and structured instances. The values of NRC seems to be close to normal distribution, shown by our experiment to determine the variance of NRC; therefore, we can use the mean of NRC with some reliability in the subsequent experiments.

We ran experiments over instances with 50, 100, 150, 200, and 250 variables, each with 100 satisfiable test instances from the SATLIB benchmark [7] (uf50-128, uf100-430, uf150-645, uf200-860, and uf250-1065). Each of the 500 test instances had a fixed clause-to-variable ratio 4.3 (m/n=4.3). Instances with more variables (e.g. 300 variables or more) exceeded our computational power as we mentioned above, though we believe that the phenomena we observed will extend to instances with larger numbers of variables. Experiments depicted in Figure 1.a show that the percentage of redundant clauses ($\%RC$) decrease as a function of the number of clauses in the instance, commencing at 40% (50 variables) and reaching to 20% (250 variables). Our observations are consistent with [4] (figure 7) and suggest that the percentage of redundant clauses may decrease to 0 as the number of variables grows to infinity.

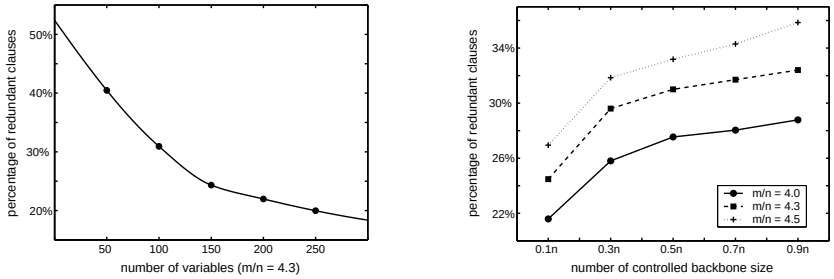

Fig. 1. (a) In random SAT, mean $\%RC$ decreases when n increases. (b) In backbone controlled instances, mean $\%RC$ increases when the controlled backbone size increases and when m/n decreases.

The backbone is the set of literals that are entailed by an instance. Figure 1.b shows the $\%RC$ in structured instances with controlled backbone sizes [7] ($CBS\_k3\_n100\_m403, m429$ and $m449$ instances, $n = 100$, with different m/n). Our results indicate strong correlation between $\%RC$ and the backbone size: $\%RC$ increases when the controlled backbone size increases. In addition, the $\%RC$ increases when m/n decreases.

Our experiments also show that real-world problems (encoded into SAT) have varying degree of redundancy that is largely dependent on the nature of the

problems and the encoding methods. For example, we find that graph coloring problems [7] (flat50-115 instances, $n = 150$, $m/n = 3.6$) have around $30\%RC$ while Blocks World Planning problems [7] (bw_large instances, with different m/n), have around $20\%RC$.

2.3 Kernel SAT instances and Constrainedness

If we fix the number of variables but change m/n, we obtain very different results regarding $\%RC$. Figure 2 plots the mean satisfiable core size against the m/n of the original random 3SAT instance. Each data point represents 100 instances. Although similar results have been shown in [4], we will give a more detailed explanation and then show a new categorization of satisfiable cores.

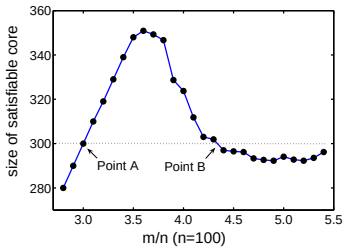

Fig. 2. Mean satisfiable core size vs. m/n of original random instance.

Figure 2 depicts two major transitions. The first one is at $m/n = 3.7$. Observe that the cores of random 3SAT instances with $m/n < 3.7$ are (virtually) the instances themselves. We refer to such cores as *kernel SAT instances*. Also note that the mean satisfiable core size peaks around $m/n = 3.7$. This suggests that (for $n = 100$) 360 clauses may be sufficient to characterize most/all satisfiable 3SAT instances. Further, the mean satisfiable core size decreases starting at $m/n = 3.7$. The second transition point occurs at around $m/n = 4.4$. For random 3SAT instances with $m/n > 4.4$, the size of the satisfiable cores plateaus at around 290-300 clauses, almost never dipping below 290 clauses.

The results lead us to believe that for $n = 100$, satisfiable cores of fewer than 290 clauses can only be kernel SAT instances; satisfiable cores with 360 clauses rarely exist; and 3SAT instances with a satisfiable core size of 290-360 clauses may either be kernel SAT instances (we refer to these as *Type A.*) or be generated via removal of redundant clauses from a larger random SAT instance (we refer to these as *Type B.*) For example, a satisfiable core with 300 clauses could be a kernel SAT instance with 300 clauses (Point A in Figure 2), or a satisfiable core of a random formula with 435 clauses (Point B in Figure 2). Intuitively, a Type A satisfiable core is not critically constrained, whereas a Type B is.

To measure constrainedness, we adopt the *crossover approach* proposed by Crawford and Anton in [8]. The crossover approach adds a few random clauses

(20 clauses when n=100) to a satisfiable formula; if the resulting formula becomes unsatisfiable then the original formula is close to the transition point (between satisfiable and unsatisfiable formulae) and is thus deemed to be *critically constrained*.

We examine problems with satisfiable core size ranging from 290 to 350 clauses where both Type A and B instances exist. We generate 100 instances for each type. Following Crawford and Anton's technique we then added 20 random clauses to each instance and checked whether the instance was satisfiable or unsatisfiable. The results are depicted in Figure 3.a. The probability of a Type A instance becoming unsatisfiable after adding 20 clauses is under 3%, while in sharp contrast, that of a Type B instance is approximately 78% for 290 clauses!

In Figure 3.b, we show that the Type B cores, much more critically constrained than Type A, are also much harder to solve. Notably, the Type B cores are even much harder than their original problems, which suggests the important role of redundant clauses in reducing search cost. As we will explain in the next section, search cost is determined by two factors: the size of a satisfiable core and the number of redundant clauses.

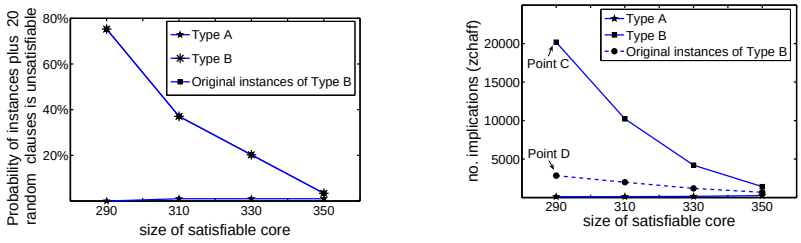

Fig. 3. (a) The graphs of Type B core and its original instances almost coincide. Type A instances are Kernel SAT instances so their original instances are themselves. (b) Search cost in term of number of zchaff implications. The original problems of Type B are much easier to solve than their Type B cores. For example, Point C (290 clauses) is a satisfiable core of Point D (450 clauses).

In [9] Gent and Walsh examined the crossover rate of 100 of the hardest random SAT instances for the DPLL procedure, chosen from 100,000 random instances. The crossover rate was 5% when there were 290 clauses in the instance. Certainly, the Type B satisfiable cores (those generated by removing redundant clauses) are much more constrained than the 100 hardest instances chosen from 100,000 random SAT instances. This leads us to posit that studying the structures of Type A and Type B may be a useful approach to understanding the meaning of constrainedness.

3 A New Measure of Hardness

As we have seen in Figure 3, both NPC and NRC are important factors influencing the search cost. Therefore, we propose a new measure, weighted clause-to-variable ratio (WCV), in an attempt to combine these two factors. We define WCV to be $WCV = (NPC + \alpha * NRC)/n$, where α is a controlling parameter for fitting the correlation, empirically set to 0.20 for random instances.

WCV resembles the definition of m/n ratio, although we replace m by a linear combination of NPC and NRC. WCV may not work well for instances that have very few redundant clauses or none because in that case, $NRC = 0$ and WCV is exactly the same as m/n ratio. Luckily, it is not a significant issue because the instances that we are interested in have abundant redundant clauses.

WCV is negatively correlated with the hardness of instances, in sharp contrast to the m/n ratio. When $m/n \ll 4.3$ or $m/n \gg 4.3$, the instance is easy; when m/n is about 4.3, the instance is hard. In contrast, WCV is monotonically correlated: the higher the WCV the easier the instance is. Figure 4 shows the empirical results. The left Y-axis is the number of implications in zchaff, a measure of the hardness of instance as it takes zchaff more implication steps to solve hard instance. The right Y-axis is the value of WCV, on a *descending* scale. The two curves in Figure 4 strongly coincide, which indicates the strong correlation.

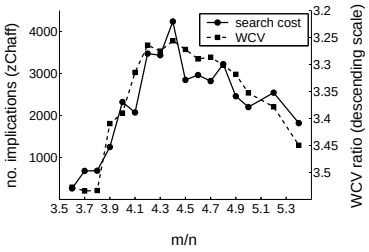

Fig. 4. Correlation WCV ($\alpha = 0.2$) and search cost. n=100.

In the under-constrained region, the NPC portion of m decreases rapidly, as illustrated in Figure 2. While NRC is increasing, it cannot compensate for the loss of NPC because only $\alpha * NRC$ contributes towards the computation of WCV. Consequently, search cost continues to increase while WCV begins to decrease. In the over-constrained region, as m (or m/n) increases, NPC remains stable (as also illustrated in Figure 2). In this region WCV is dominated by the increase in NRC. Therefore, search cost decreases here as WCV increases. In summary, NPC dominates the prediction of behavior in the under-constrained region and NRC dominates in the over-constrained region. Arguably, WCV is not really a predictor, since NPC is even harder to compute than solving the instance. Nevertheless, like the studying of backbones (which are also difficult to compute), we propose an interesting qualitative model to explain the phase

transition phenomenon with the hope that our results may lead to deeper insights into the nature of SAT and SAT solving.

4 Comparison with Other Measures

A number of interesting parameters have been proposed to study phase transition and the hardness of instances, including but not limited to clause-to-variable (m/n) ratio by Mitchell, Selman, Levesque and many others [10], the "constrainedness" measure by Gent and Walsh [11], the number of solutions by Clark et al. [12], backbone size and backbone fragility by Parkes, Singer, and Gent et al. [13] [14], the number of implicates and prime implicates by Schrag and Crawford [15], backdoor by Williams, Gomes and Selman [16]. The hardness of instances are typically measured in terms of search cost, which could be the number of resolution steps in systematic search or the number of variable flipping steps in local search.

Clause-to-variable (m/n) ratio is the simplest measure and the only one that can be easily calculated. Numerous empirical experiments suggest a phase transition happening at the point where the ratio is about 4.3 for random 3-SAT. Nevertheless, while m/n is accurate for random instances, it may not work for structured instances, which in general represent the *real* problems that we care to solve. For example, consider two instances with the same m/n but different backbone size: one with $0.1n$ backbone and the other with $0.9n$ backbone size. We know that typically the second instance is much more difficult to solve than the first instance, despite the fact that they have the same m/n. On the contrary, WCV correctly predicts the hardness of backbone controlled test instances in Figure 5. WCV is a generalization of m/n ratio; it works for both random and most structured clauses. Nevertheless, despite the limitations of m/n ratio, it remains a popular measure in SAT due to its simplicity and effectiveness for random instances.

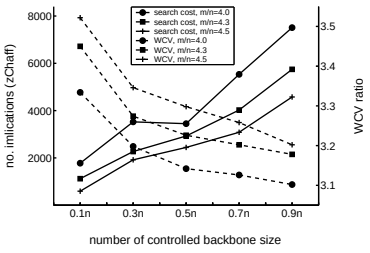

Fig. 5. $n = 100$. WCV ($\alpha = 0.2$) decreases and search cost increases when controlled backbone size increases; when backbone size is fixed, lower WCV means higher search cost.

The *constrainedness* measure [11] and the *number of solutions* [12] are largely overlapping measures because constrainedness is defined in term of the number

of solutions. Gent and Clark et al. show that constrainedness is a good predictor of the search cost problems in the under-constrained region as well as certain CSP problems.

Parkes [13] and Singer et al. [14] all suggest search cost is a result of two competing factors. They agree that the first factor is the number of solutions, or the constrainedness. In the under-constrained region, the number of solutions dominates search cost; the fewer solution an instance has, the higher search cost it takes.

There are different opinions on the second factor, which is the dominating factor in the over-constrained region. Singer et al. shows that backbone fragility (i.e., how persistent the backbone is as clauses are removed) is a good candidate of the second factor. Singer et al. explain that search cost decreases in the over-constrained region because adding clauses increases backbone fragility for large backbone instances and the majority of the instances in that region have large backbones.

Backdoor [16] is another closely related concept to satisfiable core. A (strong) backdoor of an instance is a set of variables which, when they are assigned, gives a simplified instance that may be solved in polynomial time. Backdoor variables are inherently algorithm dependent. Variable ordering technique in backtracking solvers is an example of implicit use of backdoor. Although backdoor and satisfiable core both aim to discover the hidden structure of a problem, identifying backdoor variables and satisfiable cores are very difficult. An interesting question is to find backdoors in a satisfiable core rather than the original instance.

Finally, we contrast the notion of satisfiable core to the notion of prime implicates [15]. Informally, a prime implicate is a clause β entailed by Σ such that there is no β' entailed by Σ such that $\beta' \models \beta$. Like our satisfiable core, prime implicates are often used to provide a minimal or core characterization of a formula Σ. Nevertheless, unlike satisfiable core, prime implicates have no syntactic restrictions. They are only semantically related to Σ and need not be clauses taken from Σ.

The studies of above measures have produced fruitful results, both theoretical and practical. We have sufficient reasons to believe that satisfiable core is another important concept in this family.

5 Summary and Related Work

In this paper we introduced the notions of satisfiable cores and experimentally examined their relationship to the hardness of satisfiable 3SAT instances. We extended previous redundancy results in several ways: first, we gave a more comprehensive evaluation of satisfiable cores and redundant clauses; we introduced two vastly different types of satisfiable cores, namely the kernel instance cores and the generated cores. Our experimental findings provided a new look at critically constrained problems, and their relationship to the hardness of instances. With these observed phenomena in hand, we proposed a new measure based on the size of satisfiable core, *weighted clause to variable ratio*, WCV. Experimental

results indicated that WCV is strongly correlated to the hardness of instances and is a useful qualitative model in explaining the phase transition.

A number of interesting parameters have been proposed to study phase transition and the hardness of instances, including but not limited to clause-to-variable (m/n) ratio by Mitchell, Selman, Levesque and many others [10], the "constrainedness" measure by Gent and Walsh [11], the number of solutions by Clark et al. [12], backbone size and backbone fragility by Parkes, Singer, and Gent et al. [13] [14], the number of implicates and prime implicates by Schrag and Crawford [15], backdoor by Williams, Gomes and Selman [16]. We fully believe satisfiable core is unique and important because it not only preserves original satisfying assignments but also preserves substructures (non-redundant clauses) of the original problem.

In retrospect, the study of backbone leads to backbone guided search and backdoor to backdoor search. We have focused on the characterization of satisfiable core in this paper, though finding methods for using appropriate redundant clauses is still a major challenge and one that has good potential for improving SAT solving. There have already been extensive results on systematic search via clause learning [5] [6] as well as in local search via long range dependencies and the 2-simplify procedure [17], neighborhood resolution [18], clause weighting schemas [19], and implied constraints generation [20]. Nevertheless, Alsinet et al. [21] reported that adding redundant clauses to instances encoded from all-interval-series problems may have a negative performance impact. Further cost-benefit analysis of using redundant clauses is clearly warranted.

In future work, we are very interested in a theoretical framework of satisfiable cores, possibly tracing back to the mathematical root of the structure core in graph theory. Further experiments may be conducted for deciding α in WCV for instances with different structures, and for further understanding the meaning of constrainedness using satisfiable cores.

References

[1] Bruni, R. and Sassano, A.: Restoring Satisfiability or Maintaining Unsatisfiability by finding small Unsatisfiable Subformulae, SAT-2001

[2] Zhang, L. and Malik, S.: Extracting small unsatisfiable cores from unsatisfiable boolean formulas, SAT-2003

[3] Lynce, I. and Marques-Silva, J.: On Computing Minimum Unsatisfiable Cores, SAT-2004

[4] Boufkhad, Y. and Roussel O.: Redundancy in Random SAT Formulas, AAAI-2000

[5] Ryan, L.: Efficient Algorithms for Clause Learning SAT Solvers, Master Thesis, SFU (2004)

[6] Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L. Malik, S.: Chaff: Engineering an Efficient SAT Solver. 39th Design Automation Conference (DAC 2001)

[7] SATLIB benchmark problems, available at http://www.intellektik.informatik.tu-darmstadt.de/SATLIB/benchm.html

[8] Crawford, J. and Anton, L. D.: Experimental Results on the Crossover Point in Satisfiability Problems, AAAI-1993

[9] Gent, I.P. and Walsh, T.: The satisfiability constraint gap. Artificial Intelligence, 81(1–2):59–80 (1996)
[10] Mitchell, D., Selman, B., Levesque, H.: Hard and Easy Distributions of SAT Problems, AAAI-1992
[11] Gent, I., MacIntyre, E., Prosser, P., Walsh, T.: The Constrainedness of Search, AAAI-1999
[12] Clark, D., Frank, J., Gent, I., MacIntyre, E., Tomov, N. and Walsh, T.: Local Search and the Number of Solutions, CP-1996
[13] Parkes, A.: Clustering at the Phase Transition, AAAI-1997
[14] Singer J., Gent I., and Smaill, A.: Backbone Fragility and the Local Search Cost Peak, Journal of Artificial Intelligence Research 12 (2000) 235-270
[15] Schrag, R. and Crawford, J. M.: Implicates and Prime Implicates in Random 3-SAT. Artificial Intelligence 81 199-222 (1996)
[16] Williams, R., Gomes, C., and Selman, B.: Backdoors to typical case complexity, IJCAI-2003.
[17] Wei W. and Selman B.: Accelerating random walks, CP-2002
[18] Fang, H. and Ruml, W.: Complete Local Search for Propositional Satisfiability, AAAI-2004
[19] Cha, B. and Iwama, K.: Adding new clauses for faster local search, AAAI-1996
[20] Colton, S., Drake, L., Frisch, A. M., Miguel I. and Walsh T.: Automatic Generation of Implied Constraints: Initial Progress, Proceedings of the 8th Workshop on Automated Reasoning (2001)
[21] Alsinet, T., Bejar, R., Cabiscol, A., Fernandez, C., and Manya, F.: Minimal and Redundant SAT Encodings for the All-Interval-Series Problem, Proceedings of the 5th Catalonian Conference on AI (CCIA 2002)