Fair Watermarking Techniques

Gang Qu, Jennifer L. Wong, and Miodrag Potkonjak
Computer Science Department, University of California, Los Angeles, CA 90095

Abstract

Many intellectual property protection (IPP) techniques have been proposed. Their primary objectives are providing convincible proof of authorship with least degradation of the quality of the intellectual property (IP), and achieving robustness against attacks. These are also well accepted as the most important criteria to evaluate different IPP techniques. The essence of such techniques is to limit the solution space by embedding signatures as constraints. One key issue that should be addressed but has not been discussed is the **fairness** *of the techniques: what is the quality of the solution subspace for different signatures, that is, how large the solution subspace is (uniqueness), and how difficulty it is to get a solution from such subspace (hardness)? In this paper, we introduce fairness as one of the metrics for good IPP techniques and post the challenge problem of how to design fair watermarking techniques. We claim that all fair techniques have to be instance-oriented and due to the complexity of the problem itself, we propose an approach that utilizes the statistical information of the problem instance. We use the satisfiability (SAT) problem as an example to illustrate how fairness could be achieved. We make the observation that the unfairness of the previous watermarking techniques comes from the global embedding of the signature and propose fair watermarking techniques. We test the uniqueness and hardness on a model with full knowledge of the solution and real life benchmarks as well. The experimental results show fairness can be achieved.*

1 Introduction

Recently, the watermarking-based technique for intellectual property protection (IPP) was proposed [4]. The key observation is that in real life, many CAD problems can be reduced to (often NP-hard) optimization problems where there exist a huge amount of different solutions of the same quality, and this results in different design implementations for the system with the same functionality. The watermarking-based techniques take advantage of this by embedding the designer's signature as additional design constraints and enforcing the final design to satisfy these constraints. This essentially makes the design rather unique and corresponds to the designer's signature which can be used as the proof of authorship.

Qu et al. [7] extend this to protect decision problems (e.g., SAT) by introducing optimization-intensive techniques. In these new methods, only part of the signature is selected and embedded, the selection is made in such a way that both the uniqueness of the signature and the likelihood of satisfying an instance of the SAT problem are simultaneously maximized. This prevents the signature-based constraints from changing a satisfiable problem to unsatisfiable.

The core idea of such IPP techniques is to cut the solution space (e.g., different implementations of a system, truth assignments to a SAT formula, etc.). With a signature embedded as additional constraints, the solution space is divided into two parts based on whether these extra constraints are satisfied or not. A watermarked solution meets both the original and the additional constraints, and this fact is used to show the authorship.

What is fairness and why it matters?
For a given watermarking technique, different signatures will be interpreted as distinct constraints and thus they will have very different impact on cutting the solution space of the initial problem. A watermark scheme is *fair* if it divides the solution space into subspaces of the same quality. The solution subspace's quality is measured by the quality of the solutions (how many, how good, etc.) and the difficulty to find them.

Consider a formula over four variables in the standard CNF format[10]:

$$\mathcal{F} = (v_1 + v_3' + v_4)(v_1 + v_4')(v_2 + v_4)(v_2' + v_3 + v_4') \quad (*)$$

$\mathcal{F}$ is satisfiable and has six truth assignments:

$$\{(0,1,0,0),(1,0,0,1),(1,0,1,1),(1,1,0,0),(1,1,1,0),(1,1,1,1)\}. \quad (**)$$

For example, one of the watermarking methods in [7] is to add new clauses. By adding a subset the following clauses:

$$\{(v_1 + v_3')(v_2 + v_3)(v_1 + v_3')(v_3 + v_4)(v_1' + v_3')(v_3 + v_4)\}$$

only certain truth assignments still satisfy the new formula.

If clauses $(v_1 + v_3')(v_2 + v_3)$ or $(v_1 + v_3')(v_3 + v_4)$ are added then there exist 4 and 5 truth assignments respectively, while only $(1, 0, 0, 1)$ meets the subset $(v_1' + v_3')(v_3 + v_4)$ and there is no truth assignment to satisfy subset $(v_1' + v_3')(v_2 + v_3)(v_3 + v_4)$.

Now it becomes clear why we need fair watermarking methods. In the above example, little or no extra effort is needed to find a solution that satisfies $(v_1 + v_3')(v_2 + v_3)$ or $(v_1 + v_3')(v_3 + v_4)$ in addition to the original formula; however, it is relatively difficult to find the solution that satisfies clauses $(v_1' + v_3')(v_3 + v_4)$; and it becomes impossible to solve the problem when clauses $(v_1' + v_3')(v_2 + v_3)(v_3 + v_4)$ are appended.

In this paper, we present how to develop fair watermarking techniques. In particular, we claim that a fair watermarking technique has to be instance-dependent and we demonstrate this on the SAT problem by modifying the schemes in [7] so that fairness is possible. We use two different approaches to test the fairness: (1) solve the instances with different watermarks and compare the average run time; (2) we develop a technique to create SAT instances that have exactly k truth assignments, then we count how many are still valid for different watermarks. Both tests suggest that the modified watermarking techniques can provide fairness.

In the next section, we survey the previous effort on watermarking-based IPP. Then we outline the general design methodology for fair techniques. We apply this in section 4 to the SAT problem and propose several fair watermarking methods. In section 5, we report the experimental results and finally we conclude by revisiting the role of fairness in the design of watermarking techniques.

2 Related Work

The generic watermarking-based IPP techniques is proposed in [4], where the entire signature is translated and embedded as extra design constraints. The final implementation satisfies both the initial and extra design specifications. The authorship is proved by showing that a randomly chosen solution has little chance to satisfy the additional constraints. Although this approach has been successfully applied to almost all stages of the system design process, it protects the design as one entity and fails to protect only the core parts. It is vulnerable because the signature is interpreted as one huge watermark, it has to meet all the constraints and cannot be used to protect decision problems. The first two problems are solved by putting multiple small watermarks into the crucial parts of the design that need to be protected [5]. For decision problems, the optimization-intensive techniques [7] create constraints from the signature, select part of them to add into the initial problem.

Fairness is one of the fundamental issues whenever there are multiple users who want to share certain common resources. For example, network connections sharing the bandwidth in networking, programs sharing the same processor and memory in operating systems and multimedia applications, a group of participants in cryptosystem sharing a secret. In our case, the solution space of the original problem is the resource to be shared by different users. Each user will provide his/her signature and then receive a share (solution subspace). As we have seen from the previous section, it is important to have fair watermarking schemes. Unfortunately, little attention has been paid on the fairness of watermarking intellectual properties. The optimization-intensive approach [7] is the first (accidental) attempt which tries to give each user a still satisfiable SAT instance.

3 Fairness in Intellectual Property Protection

We use the SAT problem as an example to discuss the fairness in the context of intellectual property protection.

The goal of watermarking is to place marks into the intellectual property for the purpose of identification of authorship. Watermark-ing-based IPP techniques achieve this by constraining the original SAT instance and then showing that a resulting solution satisfies not only the original instance, but also the additional constraints that are related to the signature.

There are two issues we have to consider when designing these techniques. First, we want to have a rather unique solution that corresponds to the signature, where the uniqueness is measured by the ratio of *the number of solutions for the watermarked instance* to *the number of solutions for the initial instance*. To provide a solution with strong proof of the authorship, we want to have as few solutions as possible after watermarking. I.e., we need constraints that result in small solution subspace.

On the other hand, we have to be able to find at least one solution. In general, the smaller the solution space, the more difficult it is to find a solution [2, 8]. From this point of view, we don't want the solution subspace to be too small. Clearly there is a trade-off here, tight constraints give strong proof, but increase the difficulty of solving the problem. In summary, an ideal watermarking technique will create constraints from different signatures which divide the solution space into subspaces such that it is equi-difficult to find a solution in each subspace and the found solution provides a proof of authorship with the same level of strength.

The watermarking assumption [7] says that the SAT instances are watermarkable if their solution space is large enough to accommodate the watermarks. Such instances belong to the *easy-to-solve* category and usually can be solved in polynomial-time, despite the NP-hardness of the SAT problem [2, 3]. However, measuring the fairness is still a challenge. Needless to say, enumerating all the solution to a SAT formula is hard even if the original instance is easy to solve. This makes it (almost) impossible to use the exact ratio of numbers of solutions before and after watermarking (as mentioned above) as a metric for fairness. Furthermore, the difficulties for finding different solutions are not the same in general. For example, for formula (*) $\mathcal{F} = (v_1 + v_3' + v_4)(v_1 + v_4')(v_2 + v_4)(v_2' + v_3 + v_4')$ from the Introduction section, it is much easier to find solutions like $(1, \cdot, \cdot, \cdot)$ than find the solution $(0, 1, 0, 0)$. This suggests us that a weight in terms of difficulty-to-get has to be assigned to each solution to accurately measure the fairness.

In practice, we exploit the statistical information from the SAT instance to approximate the fairness. Consider a naive watermarking scheme: before we solve the problem, we assign true/false to a subset of selected variables and this selection encodes the signature. (This is a special case of the "adding clauses" technique [7] when we restrict all the new clauses to be single-literal-clause.). If we pick variables and assign them values randomly, in one case we may set $v_1 = 0$ while in another case we may have $v_1 = 1$. As we have shown before, this is not fair.

This scheme can be modified to avoid such extreme unfair situations, although absolute fairness cannot be guaranteed. Instead of selecting from all the variables, we restrict ourselves to a subset of "well behaved" variables. For example, for each variable, we count how many times it appears in the complementary form and how many times as uncomplementary. Then we choose variables which have roughly 50% chance to appear in either form and use these variables to embed the signature. In the above formula, since v_1 never appears in the complementary form, we will not select it as a candidate variable and the above unfairness cannot happen again.

Of course, it is not enough to count only the number of occurrence of variables. Both v_2 and v_2' appear only once in the above formula, however they cut the solution space differently: four solutions out of six have $v_2 = 1$ and only two with $v_2 = 0$. This is because v_2 appears in a short clause $(v_2 + v_4)$, comparing to $(v_2' + v_3 + v_4')$, and short clauses are relatively hard to satisfy.

A fair watermarking technique should be capable of providing solution subspaces of the same quality to different signature holders. There are two metrics to measure the solution subspace's quality: uniqueness (which provides the proof of the signature) and hardness (which shows how difficult to find a solution in the subspace). Both metrics can be defined naturally but are difficult to compute. Therefore, we use the statistical information of the problem instance to approximate the fairness. In the next section, we discuss how to make the watermarking techniques for SAT fair.

4 Fair Watermarking Techniques for SAT

Qu et al. [7] present several optimization-intensive techniques to watermark the truth assignments to the SAT problems. Their key observation is that it is not necessary to embed the whole signature, as long as a convincible proof can be provided. Considering that extra constraints may change a satisfiable formula to unsatisfiable, they introduce objective functions which measure the likelihood of a formula to be satisfied. When the signature is translated into constraints, only those that have little impact on the problem's satisfiability are directly embedded. As for the rest of the constraints, they are embedded after modification or simply discarded.

4.1 Adding Clauses

Technique statement

This technique takes a binary message, translates it into a set of clauses over n variables and append them to the original formula. It fetches the first several bits and determines the length l of first new clause, then get the next $\lfloor \log_2 n \rfloor$ bits from the message and select a proper variable to put in the clause, the form (complementary or not) of the variable is decided by the next bit. This procedure continues until all l literals of the first clause are set. Then it starts to build another new clause until the message is completely embedded. The uniqueness of the solution is provided by the fact that it also satisfies the newly added clauses.

Why it is unfair?

As we have experienced earlier, a variable may have different tendency to be true or false. The "adding clauses" technique decides to put a variable into the new clause as complementary or uncomplementary form depending on one bit from the signature, thus for a random message, any variable will have the same chance to be picked and in either complementary or uncomplementary form. For example, in formula (*), it is easy to satisfy clauses with v_1 than those with v_1'. However, v_1 and v_1' have equal chance to be selected by this technique as literal in the new clauses. This causes the unfairness.

How to make it fair?

Since a fair watermarking technique has to be fair to all possible signatures, we want to translate the signatures into constraints that have the same strength on constraining the original problem. This cannot be accomplished if we make our selection from all variables and determine their forms solely from the signature. We propose two modifications, both based on each variable's (or literal's) statistical information in the SAT instance.

(i) Pre-select a subset of variables that have equal tendency to be assigned true or false and restrict the "adding clauses" methods to only these variables, i.e., build new clauses using only the pre–selected variables.

(ii) The second modification treats the SAT problem as a formula over literals (v and v' for each variable v), and create new clauses on a pre-selected subset of literals that have the same probability to be true.

Clearly from above one can see that we cannot determine the subset of variables/literals before the SAT instance is given. In this sense, we say that any fair watermarking technique has to be instance-oriented.

4.2 Deleting Literals

Technique statement

In this method, one of the literals in each clause (except the single-literal clauses) is selected based on the signature and deleted. Thus, the solution to the watermarked formula will not take advantage of these deleted literals, while they may be useful for other solutions to the original formula, and this is the proof of authorship. For example, let

$$\begin{aligned}\mathcal{F} \;=\; & (x_2+x_6'+x_7)(x_1'+x_2'+x_3'+x_4') \\ & (x_1'+x_2+x_5'+x_{10})(x_1'+x_3'+x_7+x_8+x_9') \\ & (x_1+x_5'+x_7) \\ & (x_1'+x_2'+x_3+x_4+x_6+x_7'+x_9+x_{10}')\end{aligned}$$

and we want to embed $1999_2 = 11111001111$. The first clause has length 3 and we pick literal x_6' based on the first $\lfloor \log_2 3 \rfloor$ bit(s), 1, from the message 11111001111. Similarly, we select literals x_4', x_{10}, x_1', x_5' and x_9 from the clauses in $\mathcal{F}$, delete them and get a new formula:

$$\begin{aligned}\mathcal{F}' \;=\; & (x_2+x_7)(x_1'+x_2'+x_3') \\ & (x_1'+x_2+x_5')(x_3'+x_7+x_8+x_9') \\ & (x_1+x_7)(x_1'+x_2'+x_3+x_4+x_6+x_7'+x_{10}')\end{aligned}$$

Why it is unfair?

In one extreme case, if the formula has $v_1(v_1'+v_2)$ in it, deleting literal v_1' does not change the original formula at all, but deleting literal v_2 makes this formula unsatisfiable. The unfairness comes from the unbalanced role that each literal plays in the same clause. The elimination of a complementary literal has little impact on the satisfiability of the formula if that variable will receive a true assignment in most of the solutions. This unfairness is inevitable if we select the literals based on the random signature.

How to make it fair?

There is a simple way to make this scheme fair (Figure 1).

```
Input:   a formula $\mathcal{F}$ over variables $\{x_1, \ldots, x_n\}$,
         and a message $\mathcal{M}$.
Output: a new formula derived from $\mathcal{F}$ with $\mathcal{M}$ embedded.
Algorithm:
// watermark set-up:
  for each literal $l$, calculate its statistical information $SI(l)$;
  for each clause, select two literals $l_{k_0}, l_{k_1}$ that have the
     same $SI(l)$;
// watermark embedding:
  let $\mathcal{F}'$ be an empty formula;
  convert $\mathcal{M}$ to a binary string $\mathcal{S}$;
  for each bit $\mathcal{S}_k$ of $\mathcal{S}$
  { if ( $\mathcal{S}_k$ == 0 )
       delete $l_{k_0}$ from the $k$th clause and append to $\mathcal{F}'$;
    else delete $l_{k_1}$ from the clause and append to $\mathcal{F}'$;
  }
  report formula $\mathcal{F}'$;
```

Figure 1: Pseudo code for watermarking SAT by deleting literals.

Instead of picking from all literals in the clause, we preselect two which have the same importance from each clause. Then we delete one of them to embed either a 0 or a 1.

Fair watermarking techniques are of particular interesting for reallife problems. In most random SAT models [2, 3, 8], the random-ness implicitly suggests a natural fairness among variables/literals. For example, in the $J(n, r, p)$ model [3], a formula consists of n clauses over r variables. A variable v is in the kth clause as an uncomplementary literal with probability p, as a complementary literal with probability p, and will not appear in this clause with probability $1-2p$. For this type of formula, the modified fair watermarking techniques produce relativly little additional fairness over the original optimization-intensive techniques. However, for formulas deriving from real-life problems, there are certain non-random relationships among the variables. Fair watermarking techniques find such relationships, take advantage of them and provide fairness.

5 Testbed and Experimental Results

We implement the proposed fair watermarking techniques for the SAT problem and test their fairness using two different approaches: (i) We create a model where all the SAT instances have a known number of solutions, then we translate different signatures into constraints and check how many solutions can make these additional constraints true. This avoids the difficulty of solving the SAT problems. (ii) We test the fairness on a set of instances from DIMACS SAT benchmark [9] using **WalkSAT** [10] as the solver. In this case, we estimate the fairness by comparing the runtime.

5.1 SAT Instances with Known Solutions

To get a full-knowledge of the solutions to a SAT instance, we develop a SAT model, where each instance has exactly k known solutions. The creation of such a formula consists of three phrases: solution selection; elimination of non-solution assignments; fine tune-up of the formula.

solution selection: Based on the requirement, select k truth assignments over $n \geq \lceil \log_2 k \rceil$ variables, we call these n variables *core variables*. The requirements could be in any form, such as at least three variables have to be true, v_1 and v_7 have to receive the same value, etc.

elimination of non-solution assignments: There are 2^n different assignments to n variables, since we need a formula with exactly k solutions, we have to build the formula in such way to exclude all the rest of the combinations. The basic technique to do this is introducing new clauses. For example, clause $(v_1+v_2+v_3+v_4)$ over four variables eliminates the choice $v_1 = v_2 = v_3 = v_4 = 0$.

fine tune-up of the formula: The purpose of this step is twofold: simplify the formula from the above step by removing the redundancy, and increase the complexity for solving the formula by introducing new variables. The first goal is to express a formula using the fewest number of literals. Although there are a lot research efforts on this issue, the two basic properties being used are: $v+v' = 1, v\cdot v' = 0$. When introducing new variables, it is crucial to check that the new variables will not increase the number of solutions. This requires that the new variables will be either true or false in all the k truth assignments and this is the reason we call the previous n variables "core".

Now we show how to create a formula with exactly 3 solutions. First we select v_1, v_2 as two "core variables" and $(0,0), (0,1), (1,0)$ as three solutions. Then we will add clauses to exclude the other choice $(1,1)$ and one single clause $(v_1'+v_2')$ does this trick. Now we have the formula $\mathcal{F} = (v_1' + v_2')$ which has three solutions exactly as we require. There is no redundancy in $\mathcal{F}$, however, all the solutions can be easily found. We introducing a third variable v_3 by appending clause $(v_1 + v_3')(v_2 + v_3')$ to $\mathcal{F}$, which forces $v_3 = 0$ in all the solutions. With the addition of $(v_3 + v_4)(v_4' + v_5)$ we get two more variables that both have to be true. Finally, to "confuse" the solvers we make more changes and have a formula over five variables (we also reorder the variables):

$$\mathcal{F} = (v_1 + v_2)(v_1' + v_4)(v_2' + v_3)(v_2' + v_5)(v_3' + v_4)$$
$$(v_3' + v_5')(v_1 + v_4' + v_5)(v_1 + v_2' + v_3')$$

It is non-trivial to claim now that $\mathcal{F}$ has exactly three truth assignments: $(1,0,0,1,0), (1,0,1,1,0), (1,0,0,1,1)$.

5.2 Experimental Results

We report the experimental results on watermarking the SAT instances from both our new model and the DIMACS SAT benchmark [9]. Experiments on the former measures exactly the uniqueness of the solution subspace, this can be calculated efficiently because we have complete knowledge of the solution space for formulas created by our model. For the DIMACS benchmark, we solve the instance and use the runtime to estimate the hardness of each solution subspace.

Table 2 presents the results from fairly watermarking formulas with known solutions by deleting literals. We first create a formula with exactly k solutions over n variables, then we generate 100 random messages of the same length and embed them into the original formula. Finally we check how many solutions can meet these additional constraints imposed by the signature.

number of	number of	short message			long message		
variables	solutions	max.	avg.	min.	max.	avg.	min.
50	1,000	17	11	6	7	3	0
	2,000	50	40	31	23	18	12
	10,000	213	200	180	70	61	49
	20,000	337	310	301	116	105	92
	100,000	572	558	533	254	212	197
100	1,000	45	37	21	11	6	0
	2,000	121	92	75	28	20	7
	10,000	503	482	451	216	199	183
	100,000	1034	993	970	508	492	450
	200,000	3772	3755	3721	732	681	670
200	1,000	51	42	19	31	15	10
	10,000	447	439	402	156	134	127
	100,000	2073	2045	2007	715	689	677
	200,000	3416	3409	3371	1287	1273	1250
	1,000,000	21053	20086	19837	17655	17552	17486

Table 2: Watermarking formulas with known solutions by deleting literals. *max., avg., min.* are the maximal, average, and minimum number of solutions after watermarking over the 100 trials.

In the fair "adding clauses" watermarking technique, we create new clauses over a pre-selected variables or literals. To demonstrate the different impacts of different variables to the solution space, we do the following experiments for the ii8*.cnf instances, which are generated from the problem of inferring the logic in an 8-input, 1-output "blackbox". For each instance, we estimate statistically the tendency that a variable (v) will be assigned either true (p_v) or false ($p_{v'}$). Then we sort the variables by $|p_v - p_{v'}|$, the unbalanceness

Instance	r	n	clause length	original	best 5	best 10	best 20	best 30	best 40	best 50
ii8a1.cnf	66	186	$2 \sim 3$	0.29	0.12	0.16	0.11	-	-	-
ii8a2.cnf	180	800	$2 \sim 6$	0.14	0.20	0.18	0.22	12.93	11.77	12.58
ii8a3.cnf	264	1552	$2 \sim 6$	0.15	0.14	0.18	0.19	14.85	13.29	18.41
ii8a4.cnf	396	2798	$2 \sim 6$	0.20	0.23	0.22	0.23	0.24	0.36	0.47
ii8b1.cnf	336	2068	$2 \sim 6$	0.16	0.22	16.28	17.10	17.48	18.46	19.39
ii8b2.cnf	576	4088	$2 \sim 6$	0.24	0.28	22.73	24.21	24.28	26.72	31.05
ii8b3.cnf	816	6108	$2 \sim 6$	0.29	0.33	5.42	36.96	47.32	46.18	48.26
ii8b4.cnf	1068	8214	$2 \sim 6$	0.34	0.41	0.93	47.22	59.81	58.28	55.77
ii8c1.cnf	510	3065	$2 \sim 10$	0.20	0.13	0.20	0.17	10.66	13.65	17.26
ii8c2.cnf	950	6689	$2 \sim 10$	0.28	0.26	0.31	0.36	16.62	24.97	26.88

Table 1: Impact on the solution space by preset values to different variables. Problem instances are from DIMACS, r is the number of variables, n is the number of clauses, original is the time to solve the original instance, best i is the time for solving the instance when i variable values are preset. All run time are in *seconds*.

that a variable will be true and false in the entire solution space. The ideal choice is variable v such that $p_v = p_{v'}$. If we select 3∼5 bad variable and preset values to them, because all the instances have many clauses of length 2, in almost all the cases, these bad selections turn the problem to unsatisfiable. However, if we make good choices, we can select fairly large number of variables, set their values and still be able to find satisfiable assignments. Table 1 reports the results. (instance ii8a1.cnf becomes unsatisfiable when we select 30 variables or more because of the small size of the problem.).

6 Conclusions

We introduce the concept of fairness for watermarking-based IPP techniques. The basic idea is to cut the solution space into small parts of the same quality according to different signatures. The quality of the solution subspace is measured by its uniqueness and hardness, with the former provides proof of authorship and the latter shows how difficult it is to find a solution. We post the challenge problem of how to design fair watermarking techniques. We argue that all fair techniques should be instance-oriented, not depending only on the problem. Due to the complexity of the problems, we propose the approach that utilizes the statistical information of the problem instance.

This is illustrated by the SAT problem. We explain why the previous watermarking techniques [7] fail to provide the fairness and propose modifications to have better fairness. In the new proposed fair watermarking techniques, the signatures are embedded on top of a selected part of the problem instance which statistically gives better fairness. We use different approaches to test the uniqueness and hardness, the two metrics for fairness. For uniqueness, we generate models with known solutions and then compute the size the solution subspace created by different signature. For hardness, we solve the fairly watermarked instance and use the runtime as an estimation. The experimental result suggest that the proposed fair watermarking techniques are capable of providing fairness.

REFERENCES

[1] E. Charbon. *Hierarchical Watermarking in IC Design.* IEEE 1998 Custom Integrated Circuits Conference, pp. 295-298, 1998.

[2] P. Cheeseman, B. Kanefsky, and W.M. Taylor. *Where the Really Hard Problems Are.* Twelveth International Joint Conference on Artificial Intelligence, pp. 331-337, 1991.

[3] J. Franco, and Y.C. Ho. *Probabilistic Performance of A Heuristic for the Satisfiability Problem.* Discrete Applied Mathematics, Vol. 22, pp. 35-51, 1988.

[4] A.B. Kahng, J. Lach, W.H. Magione-Smith, S. Mantik, I.L. Markov, M. Potkonjak, P. Tucker, H. Wang and G. Wolfe. *Watermarking Techniques for Intellectual Property Protection.* 35th Design Automation Conference Proceedings, pp. 776-781, 1998.

[5] J. Lach, W.H. Mangione-Smith, and M. Potkonjak. *Robust FPGA Intellectual Property through Multiple Small Watermarks.* 36th Design Automation Conference Proceedings, pp. 831-836, 1999.

[6] A.L. Oliveira. *Robust Techniques for Watermarking Sequential Circuit Designs.* 36th Design Automation Conference Proceedings, pp. 837-842, 1999.

[7] G. Qu, J.L. Wong, and M. Potkonjak. *Optimization-Intensive Watermarking Techniques for Decision Problems.* 36th Design Automation Conference Proceedings, pp. 33-36, 1999.

[8] B.Selman, H.Kautz, and D.McAllester. *Ten Challenges in Propositional Reasoning and Search.* Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97) pp. 50-54, 1997.

[9] http://dimacs.rutgers.edu/

[10] http://aida.intellektik.informatik.th-darmstadt.de/ hoos/SATLIB/