Optimization-Intensive Watermarking Techniques for Decision Problems

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

Abstract

Recently, a number of watermarking-based intellectual property protection techniques have been proposed. Although they have been applied to different stages in the design process and have a great variety of technical and theoretical features, all of them share two common properties: they all have been applied solely to optimization problems and do not involve any optimization during the watermarking process.

In this paper, we propose the first set of optimization-intensive watermarking techniques for decision problems. In particular, we demonstrate how one can select a subset of superimposed watermarking constraints so that the uniqueness of the signature and the likelihood of satisfying an instance of the satisfiability problem are simultaneously maximized. We have developed three watermarking SAT techniques: adding clauses, deleting literals, push-out and pull-back. Each technique targets different types of signature-induced constraint superimposition on an instance of the SAT problem. In addition to comprehensive experimental validation, we theoretically analyze the potentials and limitations of the proposed watermarking techniques. Furthermore, we analyze the three proposed optimization-intensive watermarking SAT techniques in terms of their suitability for copy detection.

1 Introduction

Watermarking techniques are designed for the purposes of identification and copyright of the owner and legal users of any intellectual property (IP). One major challenge in IP protection is to maintain the correct functionality of the IP. This is not a serious concern for watermarking digital images, audio, video, etc. where the ultimate consumer is human who cannot detect minute errors.

A conceptually new *constraint-based* watermarking technique has been proposed[3] and successfully applied for the protection of IPs that can be properly mapped to an optimization problem. The basic idea is to add extra design constraints and thus cut the solution space of the optimization problem. However, this technique cannot be used directly to watermark decision problems because of the natural difference between optimization and decision problems.

Decision problems, represented by the boolean satisfiability problem (SAT), play the central role in theoretic computer science and find numerous applications in various fields. Because of its discrete nature, SAT appears in many contexts in the field of VLSI CAD, such as automatic pattern generation, logic verification, timing analysis, delay fault testing and channel routing.

In this paper, we propose techniques to fill this gap. The basic idea is to embed a message in an optimal way so that the probability of changing the solution to the decision problem is minimized. We do not need to convert the entire message into additional constraints as long as we can provide convincing proof of authorship.

This research was supported in part by NSF under grant CCB-9734166.

A motivational SAT example

Given a formula of 13 variables in the standard CNF format[8]:

$$\begin{aligned}\mathcal{F} = \; & x_2 \cdot x_5(x_3+x_4)(x_1'+x_3')(x_9'+x_8')(x_3+x_{11}) \\ & (x_{13}'+x_{12}')(x_1+x_2+x_3')(x_6+x_8+x_9) \\ & (x_1+x_3+x_7)(x_4+x_{11}+x_{10})(x_1'+x_2'+x_3') \\ & (x_{12}+x_5'+x_2)(x_7'+x_6+x_4+x_5)(x_2'+x_3+x_5+x_9) \\ & (x_3+x_5+x_7'+x_{12})(x_6+x_4'+x_8+x_{11}+x_{13}')\end{aligned}$$

$\mathcal{F}$ is satisfiable and there exist **256** truth assignments. We encode a message into new clauses using the simple case-insensitive watermark key shown in Figure 1. Each word is interpreted as a new clause of the same length as that word. For example, phrase "*A red dog*" is translated to $x_1(x_9'+x_3+x_2')(x_2'+x_8+x_4)$. After we embed the message "*A red dog is chasing the cat*", the number of satisfying assignments decreases to **12**.

Figure 1: GUI for watermarking SAT.

For a specific solution to the watermarked formula, we have a chance of $\frac{1}{12} \approx 8.3\%$ to get it. But from the original $\mathcal{F}$, this probability is $\frac{1}{256} \approx 0.39\%$. The odd is about 1:21, which is the strength of the watermark. However, if we use the same technique to embed "*A red dog is chasing the bee*", then the new formula becomes *unsatisfiable* and we receive the wrong answer. With our new optimization techniques, this can be avoided by replacing the "blind encoding" with a selective one. Once we detect a to-be-added clause that has high probability of changing the satisfiability of the problem, we may decide to modify it or not to add it at all.

In next section, we review the concept of constraint-based watermarking techniques. Then we propose the optimization-intensive watermarking methodology for protecting decision problems. As an example, we develop three such techniques for the SAT problem. We analyze these optimization techniques and present the experimental results before conclude.

2 Related Work

A watermark is a mark embedded into an object for identification of the owner. Recently, the constraint-based watermarking technique was proposed [3], which we will review in details next. Their core idea is to add extra signature-related constraints to the problem before solving it, and an exact matching between the solution and the

signature shows the proof of authorship. This method is restricted to optimization problems because new constraints may change the answer of a decision problem.

Because of the importance of SAT in both theoretical and applied computer science, many heuristics have been developed[8, 6] and rigorous analysis has been conducted based on well-defined random models[1, 2, 5]. The former gives us tools to solve the problem and the latter provides us theoretical background. Most of the current available SAT solvers fall into three categories: systematic search (e.g., POSIT, NTAB, REL SAT and REL SAT-rand, Satz and Satz-rand.), stochastic local search (e.g. GSAT and WalkSat.) and translation to 0-1 integer programming.

Constraint-Based Watermarking Methodology

In[3], the concept of constraint-based watermarking methodology is introduced for the purpose of IP protection. This generic scheme has been successfully applied at the level of algorithms, behavior, logic synthesis and physical design, as well as in FPGA designs.

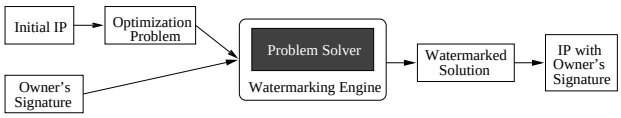


Figure 2: Illustration of the constraint-based watermarking.

Figure 2 illustrates the general approach. We take the initial IP, which is corresponding to the solution of an optimization problem, the owner's signature that needs to be put into the IP and a solver from the optimization problem. Then we build the watermarking engine that takes the optimization problem and the signature as input and returns a solution with the signature embedded. From this solution, we get the watermarked IP.

The key components for this technique are:

(i) A well-defined interpretation that maps the IP to solutions of an optimization problem with known difficult complexity.

(ii) A large solution space for the optimization problem within acceptable degradation of solution's quality. The solution space has to be large enough to accommodate the owner's watermark, and we allow some overhead to acquire this solution space for the watermarked problem.

An effective watermark must provide: high credibility, low overhead, resilience, transparency, perceptual invisibility and part protection ([3, 4]).

3 Optimization-Intensive Constraint-Based Watermarking

The essence of the constraint-based watermarking method is to cut the solution space by adding extra constraints into design process of the original IP. The authorship is proved by showing the probability of a random solution also satisfying all the extra constraints generated from the author's signature. The tighter the extra constraints, the more difficult to solve the optimization problem, and hence the more degradation the quality of solution may suffer. This trade-off between overhead and credibility is analyzed in [4]. For most optimization problems, we are guaranteed the existence of valid solutions despite of their quality. Therefore the only concern of watermarking is to keep the overhead as low as possible.

The decision problems, on the other hand, have only two different solutions: YES or NO. If the answer is YES, often a truth assignment is required. We make the following assumption that corresponds to the "large solution space" requirement for the constraint-based watermarking on optimization problems.

Watermarking Assumption:

The decision problem to be watermarked must have an answer YES and have many different ways to achieve this answer.

Since the watermarked IP has to maintain the correct functionality, the question arises immediately when we try to watermark decision problems by adding extra constraints: *Will the YES/NO answer stay unchanged as we watermark the decision problem?*

It is not difficult to construct counter-examples where we may turn a satisfiable formula to unsatisfiable by adding clauses. To avoid this scenario, we propose the optimization-intensive watermarking, where only a selected part of the signature is embedded. Recall that when we watermark the optimization problem, the proof of authorship relies on the probability of coincidence. We cannot show 100% of authorship even if a perfect matching is found in the IP with the owner's signature unless there is no coincidence. Instead, we prove by reasoning that the probability of coincidence is so small that it is unlikely to happen. So there is no reason to embed the entire signature as long as we can provide a convincible proof of authorship.

For hard decision problems, there is no simple test that tells us which constraint will cut the solution space slightly and which one may completely change the answer to the problem. In the optimization constraint-based watermarking techniques we will present soon, we intend to add a subset of the constraints from the signature into the IP based on statistical information while optimally keeping the YES/NO answer to the original decision problem.

4 Optimization Watermarking Techniques

In this section, we present three watermarking techniques on the satisfiability problem to explain the methodology of optimization constraint-based watermarking for decision problems.

Basic Notations:

$\{x_i : i = 1, 2, \ldots, n\}$ is a set of *boolean variables*, and we denote a variable x_i's *complement* by x_i'.

A *literal* is either a variable or its complement.

A *clause* is a disjunction (logic-OR, denoted by +) of one or more literals. We say a clause is true iff at least one of its literals is assigned value 1.

A *formula* is a conjunction (logic-AND, denoted by $\cdot$ or omitted when there is no ambiguity) of one or more clauses. A formula is satisfiable if there is a truth assignment to the variables, such that all the clauses are true.

For example, the formula $\mathcal{F} = x_1(x_2 + x_3')(x_4 + x_5' + x_6)$ over variables $\{x_1, x_2, x_3, x_4, x_5, x_6\}$ is satisfiable and one truth assignment can be $\{x_1 = 1, x_2 = 1, x_3 = ?, x_4 = ?, x_5 = 0, x_6 = ?\}$, where ? is *don't-care* which means the value of this variable does not affect the satisfiability of the given formula.

4.1 Adding Clauses

Given a set of n boolean variables, we may have 2^n truth assignments, this is the potential solution space of any satisfiability problem over n variables. A satisfiable formula has non-empty solution space while a unsatisfiable formula's solution space is empty. Any clause in a formula is a constraint that will prune the solution space. For instance, a clause $x_1 + x_2$ will eliminate all truth assignments that assign both x_1 and x_2 to be 0 from the solution space.

Input: a formula $\mathcal{F}$ over variables $\{x_1, \ldots, x_n\}$, and a message $\mathcal{M}$.
Output: a new formula $\mathcal{F}'$ derived from $\mathcal{F}$ with $\mathcal{M}$ embedded.

Algorithm:
copy $\mathcal{F}$ to $\mathcal{F}'$;
convert $\mathcal{M}$ to a binary string $\mathcal{S}$;
while ($\mathcal{S}$ is not empty)
{ get the length l of the next clause $\mathcal{C}$ from the first k bits of current $\mathcal{S}$;
construct the clause $\mathcal{C}$ from the next $l\lfloor 1 + \log_2 n \rfloor$ bits of $\mathcal{S}$;
evaluate the objective function Obj at $\mathcal{C}$;
if ($Obj(\mathcal{C}) \geq$ preset_threshold) add $\mathcal{C}$ to $\mathcal{F}'$;
advance $\mathcal{S}$;
}
report formula $\mathcal{F}'$;

Figure 3: Pseudo code for watermarking SAT by adding clauses.

Under the "watermarking assumption", the most straightforward way to embed signatures is to add new clauses into the formula and hence restrict the solution space. The extra clauses will be

generated from the signature and can be used to prove the existence of the signature as shown in Figure 3.

The objective function $Obj()$ takes clauses as input and return a non-negative value, which measures the likelihood that adding this clause will not change the formula to unsatisfiable.

4.2 Deleting Literals

In general, the longer the clause is, the easier it will be satisfied. (A clause with k literals is false iff all these k literals are assigned 0). Based on this observation, we propose the second technique:

Input: a formula $\mathcal{F}$ over variables $\{x_1, \ldots, x_n\}$, and a message $\mathcal{M}$.
Output: a new formula $\mathcal{F}'$ derived from $\mathcal{F}$ with $\mathcal{M}$ embedded.

Algorithm:
convert $\mathcal{M}$ to a binary string $\mathcal{S}$;
while ($\mathcal{S}$ is not empty)
{ get the current clause $\mathcal{C}$ from $\mathcal{F}$;
get the first $\lfloor \log_2 l \rfloor$ bits from $\mathcal{S}$, where l is the length of clause $\mathcal{C}$;
delete the k^{th} literal from $\mathcal{C}$ and get $\mathcal{C}'$, where $k = \lfloor \log_2 l \rfloor$ in binary;
evaluate the objective function Obj at clause $\mathcal{C}'$;
if ($Obj(\mathcal{C}') \geq$ preset_threshold)
append $\mathcal{C}'$ to $\mathcal{F}'$;
else append $\mathcal{C}$ to $\mathcal{F}'$;
advance both $\mathcal{F}$ and $\mathcal{S}$,
}
report formula $\mathcal{F}'$;

Figure 4: Pseudo code for watermarking SAT by deleting literals.

For example, let

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

And we want to embed $1998_2 = 11111001110$. Following Figure 4, we select literals $x_6', x_4', x_{10}, x_1', x_5'$ and x_9 from the clauses in $\mathcal{F}$, delete them and get a new formula:

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

It is clear that the solution space for formula $\mathcal{F}'$ is a proper subset of that for $\mathcal{F}$, so any truth assignment that satisfies $\mathcal{F}'$ also satisfies $\mathcal{F}$. Moreover, the ratio of size of $\mathcal{F}'$'s solution space to that of $\mathcal{F}$'s shows the authorship.

4.3 Push-out and Pull-back

When we keep the formula fixed and introduce new variables, the solution space will increase because these new variables serve as "don't cares" in the formula. This suggests us a variation of the "adding clauses" technique, where we embed a watermark into the same formula but over a larger set of variables, then restrict the solution to the original variable set. Figure 5 shows the idea, in (c) and (d) the shaded area is the solution space for the formula with watermarked clauses.

With the freedom of adding new variables, we can change the "adding clauses" technique in the following way: if we detect a dangerous clause, i.e., one that may make the entire formula unsatisfiable, introduce a new variable. In this way, we have better chance to maintain the satisfiability of the watermarked formula.

5 Analysis of the Optimization Techniques

5.1 The Objective Function

Definition 5.1: Let $\Psi = \{\mathcal{F} : \mathcal{F}$ is a formula over a set of boolean variables$\}$, the objective function $Obj : \Psi \to R^+ \cup \{0\}$ measures the likelihood of a formula can be satisfied and it is defined by:

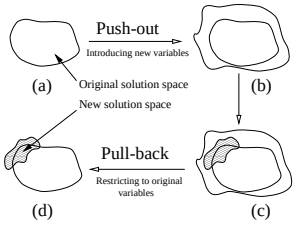
$Obj(\mathcal{F}) = min_{1 \leq i \leq n}\{Obj(\mathcal{C}_i)\}$ for any formula $\mathcal{F} = \mathcal{C}_1 \cdot \ldots \cdot \mathcal{C}_n$.

$Obj(\mathcal{C}) = \Sigma_{i=1}^{n} Obj(l_i)$ for any clause $\mathcal{C} = l_1 + l_2 + \ldots + l_n$.

$Obj(l)$ = the likelihood that literal l is assigned true.

First order objective function

For a literal l, let n_l be the number of clauses that contains l and s_i be the length of the i^{th} such clause. Then we define the first order objective function on l as:



(a) Solution space for the formula over original variables.
(b) Enlarge solution space by introducing new variables.
(c) Prune the solution space by embedding watermark.
(d) Retrieve solution space for the original formula.

Figure 5: Illustration of the push-out and pull-back.

$$Obj_1(l) = \begin{cases} r(l) \cdot f_1(l) & r(l) \notin \{0, \infty\} \\ r(l) & \text{otherwise} \end{cases} \quad (1)$$

where $f_1(l) = \sum_{i=1}^{n_l} \frac{1}{2^{s_i - 1} - 1}$ and

$$r(l) = \begin{cases} f_1(l)/f_1(l') & f_1(l), f_1(l') \notin \{0, \infty\} \\ 0 & f_1(l) = 0 \text{ or } f_1(l') = \infty \\ \infty & f_1(l) = \infty, f_1(l') \neq \infty \\ & \text{or } f_1(l') = 0, f_1(l) \neq 0 \end{cases} \quad (2)$$

Proposition 5.2: The first order objective function satisfies:
(i) $Obj(l) = \infty$ if l's complement l' does not appear in the formula.
(ii) $Obj(l) = \infty$ if l is a single literal clause and l' is not.
(iii) $Obj(l)$ is increasing w.r.t. n_l and decreasing w.r.t. $n_{l'}$.
(iv) $Obj(l)$ is decreasing w.r.t. s_i.

(i) implies that if the formula does not have l', then we should set $l = 1$; (ii) means l has to be true if itself is a clause; (iii) suggests that the more l occurs, the more likely it will be assigned true; and (iv) says the longer is the clause, the less it contributes to the objective function since a long clause is easy to satisfy.

Second order objective function

We define the second order objective function by considering the correlation among literals in the same clause as:

$$f_2(l) = \sum_{i=1}^{n_l} \frac{1}{2^{s_i - 1} - 1 + \Sigma_{j=1, l_j \neq l}^{s_i} Obj_1(l_j)} \quad (3)$$

f_2 estimates the difficulty of determining the satisfiability of a formula. $Obj_1(l)$ is determined by only the occurrence of l, l' and the length of the clause where they appear. In the second order objective function, not only l and l', but also their *neighbors* (the literals in the same clause) are considered. For a given satisfiable formula, the optimization watermarking techniques do not guarantee the watermarked formula still satisfiable, but maximize this probability.

5.2 Limitations of the Optimization Techniques on the Constant-Probability SAT Model

We adopt the model $J(n, r, p)$ for generating random SAT instances. A formula of this type 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 in this clause with probability $1 - 2p$.

Franco and Ho[2] proved that almost all SAT instances are solved in polynomial time if any of the following conditions holds:

$$p < \frac{0.4 \ln n}{r} \quad (4)$$

$$p > \frac{\ln n}{r} \quad (5)$$

$$p = c\frac{\ln n}{r} \text{ and } \lim_{n, r \to \infty} \frac{n^{1-c}}{r^{1-\varepsilon}} < \infty \ \ (0.4 < c < 1, \varepsilon > 0) \quad (6)$$

And almost all randomly generated SAT have no solution if:

$$p = c\frac{\ln n}{r}, \text{ and } \lim_{n, r \to \infty} \frac{n^{1-c}}{r} = \infty \ (\ 0.4 < c < 1\ ) \quad (7)$$

These relationships are shown in Figure 6(a) (redraw from [2]). The regions to the left of curve I (eq. (4)) and above curve III (eq. (7)) consist of instances that are always unsatisfiable. Curve II's right side are instances that almost always satisfiable (eq. (6)). The shaded area is a mixture of satisfiable and unsatisfiable problems. Under the "watermarking assumption", a to-be-watermarked SAT instance belongs to the region right to curve II. After we embed watermarks, the new instance and/or the curves will move. We

want to keep the new instance to the right of (new) curve II and this is the limitation of the optimization techniques.

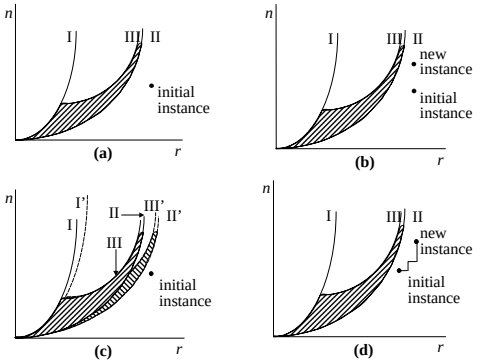


Figure 6: A SAT instance and its watermarked versions.

Adding clauses: Assuming the message is random, the watermarked instance is still random of the same type, except that the number of clauses has increased (Figure 6(b)).

Deleting literals: Deleting literals decreases p, and optimization strategy prevents us from deleting single-literal clauses and eliminating any variable completely. Therefore in the r-n chart (Figure 6(c)), the SAT instance remains unchanged, but the curves have moved towards right due to the decrement of p.

Push-out and pull-back: The newly introduced variables only appear in the added clauses, this makes the instance not random any more. However, this can be approximately viewed by Figure 6(d), where the initial instance is moving along r-axis as we add new variables, then moving up as we append new clauses.

5.3 Copy Detection

Here we outline the approaches to retrieve watermarks embedded by the "adding clauses" where solutions are forced to satisfy extra clauses according to the signature. Let $w_1, w_2, \ldots, w_k$ be the lengths of the k extra clauses and $P_k = \Pi_{i=1}^{k} 1 - (\frac{1}{2})^{w_i}$

Proposition 5.3: A random assignment makes all k clauses true with a probability P_k, and the probability that it satisfies at least $m < k$ clauses is:

$$P_m = P_k\{1 + \sum_{i_1=1}^{k} \frac{2^{-w_i}}{1-2^{-w_i}} + \sum_{i_1 \neq i_2, i_1, i_2=1}^{k} \frac{2^{-(w_{i_1}+w_{i_2})}}{(1-2^{-w_{i_1}})(1-2^{-w_{i_1}})} + \ldots + \sum_{i_j \text{ distinct}, i_1,\ldots,i_{k-m}=1}^{k} \frac{2^{-\Sigma_{j=1}^{k-m} w_{i_j}}}{\Pi_{j=1}^{k-m} 1-2^{-w_{i_j}}}\}. \quad (8)$$

In particular, for 3-SAT, $P_m = (\frac{7}{8})^k \sum_{i=0}^{k-m} \binom{k}{m} (\frac{1}{7})^i$

It is easy to see from the expression of P_m that this probability can be arbitrarily small when both k and m are large enough. Thus, this method provides high credibility of the signature for large instances. In practice, for a given SAT instance, from the limitation of the technique we can determine the maximal constraints we may introduce. Then according to the level of credibility we want to achieve, we can calculate the minimal constraints we have to add to the original problem and then fine tune the objective function.

6 Experimental Results

We have implemented our proposed optimization-intensive watermarking techniques and apply them to instances from DIMACS SAT benchmarks[7]. Due to the spacing restriction we only report the results on 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 embed the same message using regular techniques without optimization and the optimization-intensive ones respectively. The results show that in most instances, much longer messages can be embedded by the new techniques before changing the problem to unsatisfiable. Both the initial and watermarked instances are solved by **WalkSAT**[8]. All instances are solved instantaneously, the run-time overhead is negligible. Among these techniques, the "adding clauses" method has the best performance. Table 1 reports the maximal length of the bit-steam that we can take before turning the problem to unsatisfiable.

Instance	r	n	Ratio	CL	WM	Opt	Improve
ii8a1.cnf	66	186	2.4	$2 \sim 3$	1900	1400	-26.32%
ii8a2.cnf	180	800	2.6	$2 \sim 6$	4900	3100	-36.73%
ii8a3.cnf	264	1552	2.6	$2 \sim 6$	3900	6700	71.80%
ii8a4.cnf	396	2798	2.7	$2 \sim 6$	3900	3900	0
ii8b1.cnf	336	2068	-	$2 \sim 6$	3800	6800	78.95%
ii8b2.cnf	576	4088	2.3	$2 \sim 6$	4900	8900	81.63%
ii8b3.cnf	816	6108	-	$2 \sim 6$	4900	8900	81.63%
ii8b4.cnf	1068	8214	2.2	$2 \sim 6$	3900	11600	197.44%
ii8c1.cnf	510	3065	2.4	$2 \sim 10$	3900	7500	92.31%
ii8c2.cnf	950	6689	2.3	$2 \sim 10$	8900	13000	46.07%
Average Improvement							58.68%
Median Improvement							78.95%

Table 1: Improvement of the optimization-intensive technique over regular watermarking technique. r: number of variables, n: number of clauses, Ratio is measured by literals/clause, CL is the range of clause length, WM and Opt columns are the numbers of bits can be embedded before making the instance unsatisfiable by the regular and optimization-intensive techniques.

As one can see from Table 1, we achieve an average of 58.68% improvement. It is worth mentioning here that in the worst case, we successfully embedded 1400 bits, which corresponds to 63 clauses. Although the probability that a random assignment satisfies 63 additional clauses is not very small, ($< 2.3 \times 10^{-4}$), the chance that these clauses are created from a meaningful message is low.

We also test the proposed methods on random 3-SAT instances, where the literal per clause ratio is fixed at 4.25. These instances are in the range of "hard-to-be-solved"[1], i.e., even all the problems are known satisfiable, it is not expected that many satisfying assignments exist. Therefore, the "watermarking assumption" does not hold. When we try to watermark these problems, very limited message can be embedded (less than 100 bits), and the optimization-intensive techniques do not help that much. (Just imagine an instance very close to curve II in Figure 6(a)).

7 Conclusions

The current watermarking techniques can only be used for the protection of IP which are related to optimization problems. The need for effective methods to protect decision problems is urgent because of their numerous applications in various fields. In this paper, We proposed the first set of optimization-intensive watermarking techniques for decision problems. The basic concept of these techniques is to select a subset of the signature and embed it as the watermark. Theoretically, we showed that this partial signature will provide convincing authorship and an average of 58.68% improvement is achieved in practice when we implement this idea to watermark a set of benchmark SAT instances.

References

[1] 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.

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

[3] 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.

[4] G. Qu, and M. Potkonjak. *Analysis of Watermarking Techniques for Graph Coloring Problem.* IEEE/ACM International Conference on Computer Aided Design Proceedings, 1998.

[5] 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.

[6] J.P.M. Silva, and K.A. Sakallah. *GRASP— A New Search Algorithm for Satisfiability.* Proceedings of ICCAD-96, pp. 220-227, 1996.

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

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