

Hiding Signatures in Graph Coloring Solutions

Gang Qu and Miodrag Potkonjak

Computer Science Department, University of California, Los Angeles, CA 90095, USA
{gangqu,miodrag}@cs.ucla.edu

Abstract. One way to protect a digital product is to embed an author's signature into the object in the form of minute errors. However, this technique cannot be directly applied to protect intellectual properties such as solutions to hard problems which must maintain the correct functionality. This paper proposes a *constraint-based watermarking technique* and lays out a theoretical framework to evaluate this type of technique. Based on this framework, we analyze three watermarking techniques for the graph coloring problem because of its theoretical importance in complexity theory and numerous applications in real life. The first adds extra edges between some pairs of vertices and therefore forces them to be colored by different colors. The second pre-colors a set of well-selected vertices according to the watermark. And the last introduces new vertices and edges to the graph. Since credibility and overhead are the most important criteria for any efficient watermarking technique, we derive formulae which explicitly illustrate the trade-off between high credibility and low overhead. Asymptotically we prove that for almost all random graphs $G_{n,p}$, an arbitrarily high credibility can be achieved by all proposed watermarking techniques with at most 1-color-overhead. Numerical simulation on random graphs shows that a large amount of information can be embedded with very low overhead. We also watermark several sets of graphs generated from real-life benchmarks. Almost all the watermarked graphs, with a huge watermark embedded, can be colored with the known minimal number of colors for the original instances with no run-time overhead.

1 Introduction

Suppose that you have developed a very efficient heuristic to solve a well-known hard problem, for example the graph (vertex) coloring (GC) problem. You test your program on real-life benchmarks and your solutions outperform the best known solutions, then you decide to put your solutions online on a pay-per-use base to recoup the cost of developing your software. Since the solutions you post are heavily used in real-life and soon you will find yourself a target of pirates, who purchase your solutions and then resale them at a lower price. Even if you catch the dishonest buyer, he/she can claim that he/she solves the problem by himself/herself. To protect your intellectual property (IP), you need to prove to a third-party that you are the creator and owner of this solution.

In this paper, we propose a constraint-based watermarking technique for in-

tellectual property protection (IPP). Instead of solving the real problem and posting the answer directly, we build a watermarking engineer which takes the real problem and the owner's signature as input and gives a solution to the initial problem. Inside the watermarking engineer, we translate the signature into a set of additional constraints and add them into the original problem. Therefore, the solution will satisfy both the original and additional constraints. I.e., in this solution, there exist special structures that cannot be easily discovered without the owner's signature. Now the owner can claim his/her authorship by showing the small probability that such structures exist in a random solution without the signature. Since the signature is embedded as extra constraints, there might be some degradation in the quality of the IP. The trade-off between *credibility* (measures for the strength of proof for authorship) and *overhead* (measures for the degradation of quality of the IP) has to be balanced, based on which we build a framework to evaluate watermarking techniques. As an attempt to theoretically analyze watermarking techniques, the primary objective of this paper is to lay out analytical foundations for all watermarking techniques, not only those which we discuss here.

We take the GC problem as an example to illustrate our approach because of its theoretical importance in complexity theory and numerous applications in real life. The GC problem is to color the vertices of a graph with the fewest number of colors such that no vertices connected by an edge receive the same color. A watermarked solution is a coloring scheme from which the embedded information can be retrieved. We propose three watermarking techniques to embed signatures into solutions of the GC problem. For each technique, we explain how signatures can be put into the graph as extra constraints, then we do the asymptotic analysis to answer the questions about credibility and overhead (number of extra colors required in the GC problem). Surprisingly, the result shows that an arbitrarily high credibility can be achieved with at most one color overhead by all proposed watermarking techniques. This is tested by numerical simulation on random graphs. Finally we color several sets of random graphs, graphs from real-life benchmarks and the DIMACS challenge graph. For most instances, the watermarked graphs can be colored with no overhead with the same amount of run-time.

The rest of the paper is organized as follows. In section 2, we survey the related work on watermarking, random graphs and outline the constraint-based watermarking IPP techniques. The framework for evaluating watermarking techniques is then illustrated by analyzing the three watermarking techniques for the graph coloring problem. We report the numerical simulation and experimental results in section 6 and summarize our work in section 7. The proof and pseudo-code are listed in the Appendices.

2 Preliminary

Data watermarking, also known as data hiding, embeds data into digital media for the purpose of identification, annotation, and copyright. Recently, the pro-

liferation of digitized media and the Internet revolution are creating a pressing need for copyright enforcement schemes to protect copyright ownership. Several techniques for data hiding in digital images, audios, videos, texts and multimedia data have been developed [1, 3, 4, 5, 9, 10, 11]. All these techniques take advantage of the limitation of human visual and auditory systems, and simply embed the signature to the digital data by introducing minute errors. The transparency of the signature relies on human’s insensitiveness to these subtle changes.

Watermarking for the purpose of IPP, on the other hand, is more difficult because it has to maintain the correct functionality of the initial IP. A conceptually new method [6], *constraint-based watermarking*, translates the to-be-embedded signature into a set of additional constraints during the design and implementation of IP in order to uniquely encode the signature into the IP. The proof of authorship is shown by arguing the small probability for a random solution to satisfy all these extra constraints. The effectiveness of this generic scheme has been demonstrated at all stages of the design process [6].

A generic watermarking procedure consists of the following components:

- An *optimization problem* with known difficult complexity. By difficult, we mean that either achieving an acceptable solution, or enumerating enough acceptable solutions, is prohibitively costly. The solution space of the optimization problem should be large enough to accommodate a digital watermark.
- A well-defined *interpretation* of the solutions of the optimization problem as intellectual property.
- Existing *algorithms and/or off-the-shelf software* that solve the optimization problem. Typically, the “black box” software model is appropriate, and is moreover compatible with defining the watermarking procedure by composition with pre- and postprocessing stages.
- *Protection requirements* that are largely similar to well-understood protection requirements for currency watermarking.

A non-intrusive watermarking procedure then applies to any given instance of the optimization problem, and can be attached to any specific algorithms solving it. Such a procedure can be described as:

- *A use model or protocols* for the watermarking procedure. In general, each watermarking scheme must be aware of attacks based on design symmetries, renaming, reordering, small perturbations (which may set requirements for the structure of the solution space), etc.
- Algorithmic descriptions of the *pre- and post-processing* steps of the watermarking procedure. Pre- and post processing preserve the algorithms and/or software as a “black box”.
- *Strength and feasibility analyses* showing that the procedure satisfies given protection requirements on a given instance. Strength analysis requires metrics, and structural understanding of the solution space (e.g., “barriers” (with respect to local search) between acceptable solutions). Feasibility analysis requires measures of solution quality, whether a watermarked solution remains well-formed, etc.

- *General robustness analyses*, including discussion of susceptibility to typical attacks, discussion of possible new attacks, performance guarantees (including complexity analysis) and implementation feasibility.

In addition to maintaining the correct functionality of the IP, an effective watermark must satisfy the following properties:

- *high credibility*: The watermark should be readily detectable for the proof of the authorship. The probability of coincidence should be low.
- *low overhead*: The degradation of the software or design by embedding the watermark should be minimized.
- *resilience*: The watermark should be difficult or impossible to remove without the complete knowledge of the software or design.
- *transparency*: The addition of the watermark to software and designs should be transparent so that it can be used for existing design tools.
- *perceptual invisibility*: The watermark must be very difficult to detect. This is related to but not the same as the resilience problem.
- *part protection*: Ideally, a good watermark should be distributed all over the software or design in order to protect all parts of it.

The theory of random graphs was founded by Erdős and Rényi after Erdős had discovered, in the middle of this century, that probabilistic methods were often useful in tackling extremal problems in graph theory. The traditional way of estimating the proportion of graphs having a certain property is to obtain exact but complicated formulae. The new probabilistic approach is to approximate a variety of exact values by appropriate probability distributions and using probabilistic ideas.

Random graphs play a very important role in many fields of computer science. The two most frequently occurring models of random graphs are $\mathcal{G}(n, M)$ and $\mathcal{G}(n, p)$. The first consists of all graphs with n vertices and M edges, the second consists of all graphs with n vertices and the edges are chosen independently with probability p ($0 < p < 1$). We will focus on the second model and use these conventional notations: $G_{n,p}$ for an element of $\mathcal{G}(n, p)$, $q = 1 - p$, $b = \frac{1}{q}$, $\alpha(G_{n,p})$ is the independent number of graph $G_{n,p}$ (i.e., the maximal cardinality of independent sets.), and $\chi(G_{n,p})$ denotes the chromatic number of $G_{n,p}$ (i.e., the minimum number of colors required to color the graph.). For almost all graphs $G_{n,p}$, we have [2]:

$$\begin{aligned} (1) \quad & \alpha(G_{n,p}) = (2 + o(1)) \log_b n \\ (2) \quad & \chi(G_{n,p}) = \left(\frac{1}{2} + o(1)\right) \frac{n}{\log_b n} \end{aligned}$$

The graph (vertex) coloring problem is to label the vertices of a graph with minimal number of colors such that vertices connected by an edge do not receive the same color. This problem is NP-complete and has a lot of applications in various fields¹. Many heuristics have been developed dedicated to it[12]. In the next three sections, we propose techniques for watermarking the solutions to the GC problem, and lay out the theoretical framework for technique evaluation through the analyses of these techniques.

¹For example, Toft stated 75 interesting and easily-formulated graph coloring problems[8].

3 Watermarking Technique #1 — Adding Edges

3.1 Generic Watermarking Procedure for the Graph Coloring Problem

The essence of constraint-based watermarking techniques is to add extra design constraints in order to get a rather unique solution. This is shown explicitly in Figure 1 for the GC problem.

Suppose we have a graph G which is k -colorable, the inner and outer regions in Figure 1 represent the solution spaces of k -color and $(k+1)$ -color solutions to G respectively. We assume that when a k -color solution is required, every solution in the inner region has equal probability being picked. The shaded area is the solution space for the watermarked graph G' , where we impose our signature as additional constraints. Since graph G' inherits all the constraints of graph

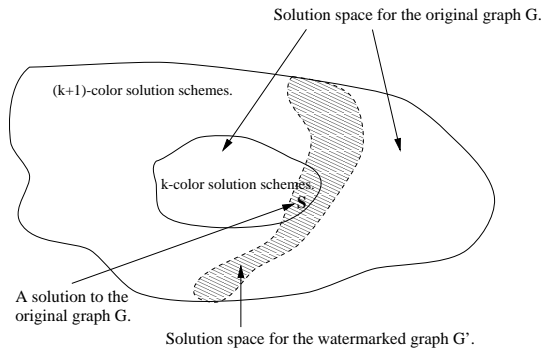


Figure 1: Additional constraints cut the solution space.

G , a solution to G' is also valid for G . However, the solutions to G may violate the new constraints in G' . By coloring graph G' instead of G , we can obtain solutions to G and more important, we force the solutions fall into the shaded area. Denote n_k and n'_k the numbers of k -color solutions for graphs G and G' . The chance to get a particular solution S from the constraints in G is $\frac{1}{n_k}$, which increases to $\frac{1}{n'_k}$ if from the more-constrained graph G' . When n_k is large and $n'_k \ll n_k$, the difference between $\frac{1}{n_k}$ and $\frac{1}{n'_k}$ is significant and becomes a credible evidence for the authorship.

High credibility depends not only on the amount of constraints, but also the “quality” of the constraints. For example, one constraint that cuts the solution space by half is definitely better than 20 constraints each cutting the solution space by less than 1%. Constraints for the GC problem are the edges: vertices connected by an edge have to receive different colors. One type of straightforward watermarks is extra edges. By translating signature as extra edges, we make the original graph more constrained, and some solutions to the original graph will become invalid for the watermarked graph. The solution space eventually shrinks. There are other interpretations of signatures as constraints. However, to have a transparent watermark, we require that the watermarked graph preserve the characteristics (e.g. connectivity, randomness, acyclicity.) of the original graph. In the following sections, we propose three watermarking techniques for the GC problems of random graphs, and investigate the impact of the corresponding watermarks to the solution space.

3.2 Technique Statement

Signature embedding:

Given a graph $G(V, E)$ and a message M to be embedded in G . Let $V = (v_0, v_1, \dots, v_{n-1})$ and we encrypt the message into a binary string $M = m_0 m_1 \dots$ (by stream ciphers, block ciphers, or cryptographic hash functions). Figure 2 shows how M is embedded into the graph G as additional constraints.

Input: a graph $G(V, E)$, a message $M = m_0 m_1 \dots$
Output: new graph with message M embedded
Algorithm: copy $G(V, E)$ to $G'(V, E')$; foreach bit m_i { find the nearest two vertices v_{i_1}, v_{i_2} that are not connected to vertex v_i ; if $m_i = 0$ add edge (v_i, v_{i_1}) to E' else add edge (v_i, v_{i_2}) to E' } report graph $G'(V, E')$;

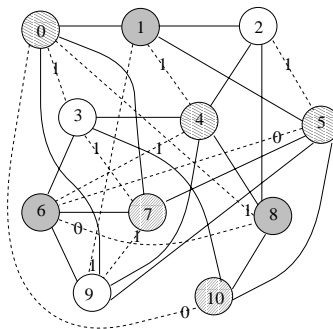


Figure 2: Pseudo code for technique # 1.

Figure 3: An example.

By the nearest two vertices v_{i_1} and v_{i_2} which are not connected to vertex v_i , we mean that $i_2 > i_1 > i \pmod{n}$, the edges $(v_i, v_{i_1}), (v_i, v_{i_2}) \notin E$ and $(v_i, v_j) \in E$ for all $i < j < i_1, i_1 < j < i_2 \pmod{n}$. For example, in Figure 3, vertices 2 and 3 are the nearest two vertices that are not connected to vertex 0. The essence of this technique is to add an extra edge between two vertices, these two vertices have to be colored by different colors which may not be necessary in the original graph G . Figure 3 shows a graph of 11 nodes with solid lines for original edges. The message $1998_{10} = 11111001110_2$ has been embedded by 11 dotted edges, each represents one bit marked on the edge. A 4-color scheme, $\{\{v_0, v_2, v_6\}, \{v_1, v_7, v_{10}\}, \{v_3, v_8, v_9\}, \{v_4, v_5\}\}$, is shown as well.

Signature recovering:

How can we read the watermark from the solution? Given the original graph, we claim that some pairs of vertices will have different colors. For example, in Figure 3, these pairs are $\{v_0, v_3\}, \{v_1, v_4\}, \{v_2, v_5\}, \dots, \{v_{10}, v_0\}$. In the original graph, every such pair of vertices are not directly connected by an edge, so it is not necessary to assign them different colors. However we observe that this happens in the coloring scheme shown in Figure 3. For each such pair $\{v_i, v_j\}$, we can retrieve one bit of information by counting how many nodes in between (i.e., nodes with indices between i and j) are not connected to v_i . If there is none, the hidden bit is 0; if there is only 1, the hidden bit is 1; and if there are more than 1, reverse the order of v_i and v_j . This binary string is the (encrypted) message. In the same manner, it is not difficult to construct many other binary strings, even if the vertices have a standard order and the watermark is embedded in the well-accepted manner. For example, node 0 in Figure 3 has different color from both nodes 2 and 3, which are the nearest two vertices that are not connected to

node 0. So both bits 0 and 1 can be claimed as the hidden bit in this case and one may have a different binary string. However, it will be hard to build one with a piece of meaningful information. In particular, if the original message is encrypted by one-way functions, forging a watermark with the same level of credibility needs to break the one-way functions.

3.3 Technique Analysis

The signature or message can be anything that is capable of identifying authorship. We can transfer it into binary (e.g., in ASCII), encrypt it by stream ciphers or cryptographic hash functions and assume the final bit stream is random. To have a quantitative analysis, we assume that exactly χ colors are required to color the graph $G_{n,p}$, where χ is given by²:

$$(3) \quad \chi(G_{n,p}) = \lceil \frac{n}{2^{\log_b n}} \rceil$$

It follows immediately that after adding k extra edges into the graph $G_{n,p}$ according to the signature, the resulting graph remains random³ with the same number of vertices and a new edge probability:

$$(4) \quad p' = p + \frac{2k}{n(n-1)}$$

So formula (3) for the chromatic number still holds, we denote this number by χ' . The **overhead** is defined to be $\chi' - \chi$, i.e., the number of extra colors required to color the watermarked graph. Intuitively, the more edges we add, the more colors we need to mark the graph. Since the number of colors is one of the most important criteria for the quality of coloring scheme, we want to keep this overhead as low as possible. One question is: how many edges can we add into the graph without introducing a large amount of overhead? Formally speaking: finding $k(n)$, the number of edges can be embedded into an n -vertex random graph, such that $\overline{\lim}_{n \rightarrow \infty} \chi' - \chi < \infty$.

Theorem 3.1

Adding $k(n)$ edges to a random graph $G_{n,p}$, for almost all $G_{n,p}$, $\overline{\lim}_{n \rightarrow \infty} \chi' - \chi = \infty$ iff $k(n) \in \omega(n \log n)$.

Corollary 3.2 (1-color overhead)

Adding $k(n)$ edges to graph $G_{n,p}$, if $\lim_{n \rightarrow \infty} \frac{k(n)}{n \ln n} = l$, then $\overline{\lim}_{n \rightarrow \infty} \chi' - \chi \leq 1 + \frac{l}{1-p}$. In particular, if $k(n) \in o(n \log n)$, for almost all $G_{n,p}$, the overhead is at most 1.

A good watermark should be able to provide any desired level of confidence. I.e., the authorship can be proved with a probability almost 1 when the graph

²We choose expression (3) instead of (2) to simplify the asymptotic analysis, all the results hold if we replace (3) by (2).

³In general, the graph is not random unless k is a multiple of n . The randomness can be maintained by modifying this technique in the following way: in Figure 2, select the first vertex of each pair according to the message M instead of the given order for the vertices. E.g., the first node will be v_l where the binary expression of l : $l_2 = m_0 m_1 \dots m_{\lfloor \log_2 n \rfloor}$. In practice, we restrict k to be multiples of n to keep the randomness.

goes large. Obviously one extra edge cannot bring high credibility. The following theorem answers the question: finding $k(n)$, the number of edges to be embedded into a n -vertex random graph, such that $\mathbf{Prob}[\mathcal{E}] \rightarrow 0$ as $n \rightarrow \infty$, where \mathcal{E} is the event that in a random solution all these $k(n)$ constraints are satisfied.

Theorem 3.3 (arbitrarily high credibility)

Adding $k(n)$ edges to a random graph $G_{n,p}$, let \mathcal{E} be the event that a random solution to the original graph also satisfies all these $k(n)$ extra constraints. Then for almost all $G_{n,p}$, $\lim_{n \rightarrow \infty} \mathbf{Prob}[\mathcal{E}] = 0$ if $k(n) \in \omega\left(\frac{n}{\log n}\right)$.

To summarize the “*adding edges*” technique, we conclude: adding $k(n) \in \omega\left(\frac{n}{\log n}\right) \cap o(n \log n)$ extra edges into graph $G_{n,p}$, as n goes large, arbitrarily high credibility can be achieved with at most 1-color-overhead. More precisely, we define the *watermark potential* (by adding edges) for graph $G_{n,p}$:

$$(5) \quad WP(G_{n,p}) = \chi(G_{n,p}) - \frac{n}{2 \log_b n}$$

This function describes the power of the “*adding edges*” technique on random graphs. We list several properties of this function with respect to n (for p , similar results hold):

- (a) $0 \leq WP(G_{n,p}) < 1$ for all graph $G_{n,p}$.
- (b) **periodic:** χ is a non-decreasing step function and $\frac{n}{2 \log_b n}$ is continuous and increasing. So $WP(G_{n,p})$ behaves periodically for different values of χ ,
- (c) **starting points:** χ increases by 1 at the start of each period $WP(G_{n,p})$ achieves its local maximum.
- (d) **locally decreasing:** In each period, since χ is constant, as n increases, $WP(G_{n,p})$ decreases.
- (e) **increasing period:** When n grows by 1, $\frac{n}{2 \log_b n}$ will increase roughly by $\frac{1}{2 \log_b n}$. Thus, the period is about $2 \log_b n$. (a little larger than $2 \log_b n$ to be more precise, since $b = \frac{1}{1-p}$ also increases.)

4 Watermarking Technique #2 — Selecting MIS

4.1 Technique Statement

A maximal independent set (**MIS**) of a graph is a subset of vertices S such that vertices in S are not connected and vertices not in S are connected to at least one vertex of S . This second technique takes advantage of the fact that vertices in one MIS can all be labeled by a single color.

Signature embedding:

Given a graph $G(V, E)$ and a message M to be embedded in G . We order the vertices set $V = (v_0, v_1, \dots, v_{n-1})$ and encrypt the message into a binary string $M = m_0 m_1 \dots$. The message M is embedded into the graph G as shown in Figure 9 (see the Appendices). The idea is to select one or more MISes

according to M , assign each MIS with one color and then color the rest of the graph. The MIS containing M is constructed in the following way: choose v_i as the first vertex of the MIS, where the binary expression of i coincides with the first $\lfloor \log_2 n \rfloor$ bits of M ; then we cut v_i and its neighbors from the graph since they cannot be in the same MIS as v_i ; we reorder the vertices and select the next vertex of the MIS based on M . When we get a MIS, we color it with one color, remove it from the original graph and start constructing the second MIS if M has not been completely embedded.

A small example of an 11-node graph with the embedded message $1998_{10} = 11111001110_2$ is shown in Figure 4, where we use three colors to color the graph $\{v_1, v_4, v_7, v_{10}\}$, $\{v_0, v_2, v_5, v_6\}$, and $\{v_3, v_8, v_9\}$. From 11 nodes, we choose node 7 to embed the first three bits of M , 111. Then all node 7's neighbors are crossed and the rest nodes are reordered; the node with the new index 3 is picked based on the next two bits 11; after cutting this node's neighbors, we obtain a MIS of the original nodes $\{1, 4, 7, 10\}$ which we mark by one color; reorder the rest 6 nodes and continue the procedure till M is completely embedded.

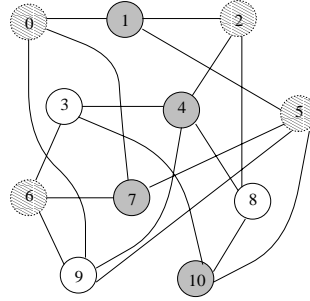


Figure 4: An example.

Signature recovering:

The selected MIS with a particular order of its vertices is the watermark. We can retrieve a binary string from this watermark by reconstructing the MIS in the specific order. For example, in Figure 4, 11111 is the information hidden behind the MIS $\{v_7, v_4, v_1, v_{10}\}$ in that order. The first vertex v_7 is node No. 7 in the original 11-vertex graph, so we have the first three bits $111_2 (= 7_{10})$. After deleting v_7 and its neighbors, there are 7 vertices left. We reorder the vertices and claim the next two bits from the second vertex of the MIS v_4 , which is now node No. 3 in the new graph. From the number 3 we get bits 11. Removing v_4 and its neighbors from the new graph gives us two isolated vertices v_1 and v_{10} , no further information can be hidden and this completes the given MIS. Similarly, the rest of the (encrypted) message 001110 is hidden in the second MIS $\{v_0, v_6, v_5, v_2\}$ in that order.

The uniqueness of the selected MIS determines the credibility. In Figure 4, vertex v_1 may be involved in any of the following MISes: $\{v_1, v_3, v_7, v_8, v_9\}$, $\{v_1, v_4, v_6, v_{10}\}$, $\{v_1, v_4, v_7, v_{10}\}$, $\{v_1, v_6, v_8\}$. The order of the vertices in the MIS also plays a very important role⁴. If we order the MIS $\{v_1, v_4, v_7, v_{10}\}$ by the indices, following the same watermarking scheme, the hidden binary string becomes to 0010101 instead of 11111.

⁴For a given MIS of size k , selecting these k vertices in different orders delivers different messages. However, it is unlikely to get the same MIS from different messages (after encryption).

4.2 Technique Analysis

Our goal is to analyze this technique follow the framework we built in the previous section. In particular, we are interested in finding formulae for overhead and credibility.

First, we claim that after removing one randomly selected MIS, the remaining graph is still random with the same edge probability. One way to generate a random graph $G_{n+1,p}$ is to add one new vertex into a random graph $G_{n,p}$ and add an edge between the new vertex and each of the old vertex in $G_{n,p}$ with probability p . Reversing this procedure says that deleting one vertex from $G_{n,p}$ results in a random graph $G_{n-1,p}$. Since the neighbors of one vertex are also random, it follows that the graph will maintain its randomness after erasing one vertex and all its neighbors.

The first vertex of the MIS can be selected randomly, while the choices for the second vertex are restricted to $(1-p)n = qn$, because all the pn neighbors of the first vertex have been eliminated. In general, only $q^k n$ vertices are left as candidate for the $(k+1)$ th vertex of the MIS. Therefore, we have:

Lemma 4.1

Given random graph $G_{n,p}$, almost all randomly selected MIS is of size $\log_b n$, where $b = \frac{1}{1-p}$.

The strength of the watermark relies on the uniqueness of the MISes we constructed as well as a specific order of the vertices in each MIS. To create a convincing watermark in a large graph, we have to add $\omega(\frac{n}{\log n})$ edges by the first technique. The same goal can be achieved by selecting only one MIS:

Theorem 4.2 (arbitrarily high credibility with 1-color overhead)

Given a random graph $G_{n,p}$, we select one MIS as in Figure 9. Let \mathcal{E} be the event that in a random solution, all vertices in this MIS have the same color and they are in the order as specified by Figure 4. Then $\lim_{n \rightarrow \infty} \mathbf{Prob} [\mathcal{E}] = 0$. Furthermore, this introduces at most 1-color overhead.

By selecting one vertex from an n -vertex graph, we can embed $\lceil \log_2 n \rceil$ bits. From Lemma 4.1, at most $\log_2 \cdot n \log_b n$ bits of information could be embedded into the MIS. To embed long messages, we have to construct more MISes,⁵ which may result in huge overhead.

Theorem 4.3

Given a random graph $G_{n,p}$, if we select $k(n)$ MISes as in Figure 9, assign each MIS one color and color the rest of the graph, then the overhead is at most $k(n)$ and on average at least $\frac{k(n)}{2}$.

⁵Alternatively, we can map long messages to a fixed length message by hash functions. Since hash function is many-to-one, this brings ambiguity which depends on the hash function itself. Such analysis is out of the scope of this paper.

5 Watermarking Technique # 3 – Adding Nodes and Edges

5.1 Technique Statement

Signature embedding:

Given a random graph $G_{n,p}$ and a message M to be embedded. We order the vertices set $V = (v_0, v_1, \dots, v_{n-1})$ and encrypt the message into a binary string $M = m_0 m_1 \dots$ which is then embedded into $G_{n,p}$ as follows: introduce a new node v , take the first $\lfloor \log n \rfloor$ bits from M , find the corresponding vertex v' and connect it to v ; take the next $\lfloor \log n - 1 \rfloor$ bits and locate the next vertex to which v is connected ($n - 1$ since v' has to be excluded); continue till we add np edges starting from v and get a new graph $G_{n+1,p}$; introduce another new node if M has not been completely embedded. We color the new graph, restrict the coloring scheme to the original graph $G_{n,p}$ and we have a solution with message M embedded.

Signature recovering:

This watermark is hard to detect because of the invisibility of the new added nodes and their associated edges. To exhibit the hidden signature in a colored graph, we have to go through the signature embedding procedure again and show that the encrypted signature can be added into the colored graph as edges to the newly inserted vertices without any conflicts. This has to be coupled with a statement of the unlikelihood that this happens for any random message. As we discussed earlier, many different binary strings can be generated in the same way from the same colored graph, but to fake one corresponds to a one-way function with a specific information is not easy.

5.2 Technique Analysis

Suppose k new nodes have been added into the initial graph $G_{n,p}$ to accommodate the message, similar to the previous two techniques, it is clear that the embedded graph is an instance of $G_{n+k,p}$.⁶ This guarantees that randomness of the watermarked graph and hence the validity of the formula (3) which implies an overhead in the amount of $\chi' - \chi = \lceil \frac{n+k}{2 \log_b(n+k)} \rceil - \lceil \frac{n}{2 \log_b n} \rceil$, where $b = \frac{1}{1-p}$.

We have defined the watermark potential for graph $G_{n,p}$ as $WP(G_{n,p}) = \chi(G_{n,p}) - \frac{n}{2 \log_b n}$. A large $WP(G_{n,p})$ means there is still room for adding new nodes and/or edges into $G_{n,p}$ without introducing a new color, especially at the starting point of each period (property (c) of function $WP(G_{n,p})$ in section 3.3). From the step function nature of χ , we have

Theorem 5.1 (1-color overhead)

Given a random graph $G_{n,p}$, we introduce $k(n)$ new vertices and associate edges based on the signature, then for almost all $G_{n,p}$, the overhead is at most 1 if $k(n) \in o(\log n)$.

⁶For the last node, we can add edges randomly or repeat the message to make sure it has $(n + k - 1)p$ neighbors.

A graph of χ colors is essentially a partition of the vertices to χ independent sets. The neighbors of any new vertex can be selected randomly from these χ set. However, to add one new vertex v without bringing a new color, v 's neighbors have to be chosen from at most $\chi - 1$ independent sets. It is not hard to see that when many edges have to be added, it is unlikely that none of these edges ending into a specific independent set.

Theorem 5.2 (arbitrarily high credibility)

We build graph $G_{n+1,p}$ from a given random graph $G_{n,p}$ by introducing one new vertex and np edges. A coloring scheme to the initial $G_{n,p}$ is obtained by coloring $G_{n+1,p}$. Let \mathcal{E} be the event: add a vertex v to the colored graph $G_{n,p}$, connect v to np random vertices, and v does not require a new color. Then $\lim_{n \rightarrow \infty} \mathbf{Prob}[\mathcal{E}] = 0$ for almost all $G_{n,p}$.

6 Simulation and Experimental Results

6.1 Numerical Simulation for Techniques # 1 and # 2

We conduct simulation in the ideal case assuming we know how to color the graph optimally. In the “adding edges” technique, we add extra edges into the original graph corresponding to the signature. Figure 5 shows for graph $G_{n,0.5}$ ($500 \leq n \leq 1000$), the number of edges can be added (y-axis) with 0-overhead ($k_0(n)$, shown as black dots) and 1-color overhead ($k_1(n)$, shown as grey triangles), the curve in between is the difference of $k_0(n)$ and $k_1(n)$. Revisiting the properties of the watermark potential function, we see that $WP(G_{n,p})$ describes correctly the amount of information can be embedded into graph $G_{n,p}$.

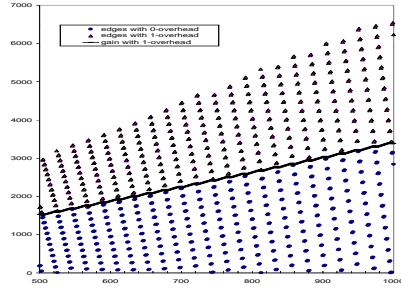


Figure 5: Numerical data for # 1.

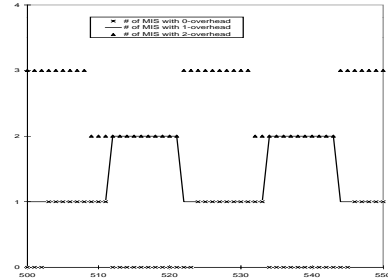


Figure 6: Numerical data for # 2.

In Figure 6, for graph $G_{n,0.5}$ ($500 \leq n \leq 550$), the numbers of MISes (y-axis) that can be constructed within 2-color overhead are given. One observation is that the number of MISes as a function of n for the same number of overhead is piecewise constant. This has been predicted from the proof of Theorem 4.3. Another fact is that when we select one MIS, with 50% probability there will be a 1-color overhead. The reason is that the increment on $\frac{n}{2 \log_b n}$ by selecting one MIS is around $\frac{1}{2}$ and $\mathbf{Prob}[\lceil x - \frac{1}{2} \rceil = \lceil x \rceil] = 0.5$.

6.2 Experimental Results

The main goal of the experiment is to compare the difficulty of coloring the original graphs vs. the watermarked graphs, as well as the quality of the solution. For this purpose, we choose three types of graphs: random graphs $G_{n,p}$, graphs generated from real-life benchmarks, and the DIMACS challenge graph. For each type of graphs, we do the simulation in three steps: (1) color the original graph, (2) apply the watermarking techniques to embed a random message, (3) color the watermarked graph. Each graph is colored 10 times and the average result is reported. All experiments are conducted on 200MHz UltraSparcII and 40 MHz SPARC 4 processors using the algorithm in [7]. The same parameters are used for the original and watermarked graph.

Original $G_{n,0.5}$			Adding n Edges			Adding $2n$ Edges			Selecting 1 MIS		
n	color	best	color	best	mesg	color	best	mesg	color	best	mesg
125	19	19	19	19	125	19	19	250	19	19	42
250	30	30	30.2	30	250	30.2	30	500	30	30	80
500	50.1	50	50.4	50	500	50.6	50	1000	50.2	50	81
1000	85.8	85	86	85	1000	86.8	86	2000	86	85	110

Table 1: Coloring the watermarked $G_{n,0.5}$.

Table 1 shows the results on random graphs $G_{n,0.5}$, and the corresponding watermarked graphs by adding n and $2n$ random edges or by selecting one MIS. The columns labeled *color* are the average numbers of colors on 10 trials for each instance, while the *best* columns are the best solutions from the 10 trials, and the columns *mesg* measure the amount of information (in bits) being embedded in the graph.

Table 2 is the result on dense/sparse random graphs. For dense graphs $G_{n,0.9}$, there is not much space left to add extra edges, so it is expensive to watermark dense graphs by adding edges. On the other hand, the size of MIS for dense graph is relatively small, therefore very limited information can be embedded by selecting MISes. For sparse graphs $G_{n,0.1}$, both techniques perform well.

Original $G_{n,p}$			Add n Edges	Add $2n$ Edges	Select 1 MIS	Select 2 MISes
n	p	color				
125	0.9	46	49.1	52	47	47.3
250	0.9	77.9	79	82	77.2	77
250	0.1	9	9.1	9.8	9	9.8
500	0.1	13.9	14	14.2	14	14.2

Table 2: Coloring other watermarked $G_{n,p}$.

When applying to the on-line challenge graph at the DIMACS site [12], for the graph with 1000 vertices and 249826 edges which implies an edge probability slightly larger than 0.5, we restrict the run-time to 1 hour and get the results from 10 trials shown in Table 3. In the 10 trials for the original graph, we find two 85-color solutions and the average number of colors is 86.1. The second column is the amount of information (in bits) being added into the graph. The last column shows the probability of coincidence, where low coincidence means high credibility. One can see both methods provide high credibility with little degradation of the solution’s quality.

For the technique of “adding vertices and edges”, we start from a random graph $G_{125,0.5}$ and introduce new vertex (and certain number of edges to keep

Edges Added	Information	Colors	Overhead	Coincidence
0	0	86.1	-	-
500	500	85.8	-0.3	2.89e-03
1000	1000	87	0.9	9.55e-06
3000	3000	87	0.9	8.71e-16
MIS Selected	Information	Colors	Overhead	Coincidence
1	110	86.2	0.1	3.52e-17
2	210	86.4	0.3	1.24e-33
3	300	87	0.9	8.51e-50
4	390	87.4	1.3	5.85e-66
5	480	87.4	1.3	2.06e-82
6	590	87.9	1.8	7.24e-99

Table 3: Coloring the watermarked DIMACS benchmark.

the edge probability) one by one till we reach an instance of $G_{549,0.5}$. Then we color each of these 425 graphs 10 times and plot the average number of required colors in Figure 7. The results for the last 50 instances are enlarged as shown in Figure 8

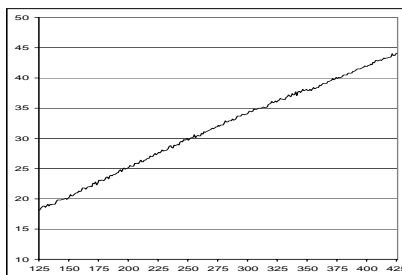


Figure 7: Numerical data for # 1.

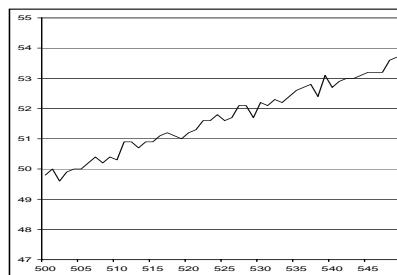


Figure 8: Numerical data for # 2.

Original Instance			Optimal Coloring	Add n Edge		Select 1 MIS		Add 1 Vertex	
Instance	Vertices	Edges		Edges	Colors	Size	Colors	Edge	Colors
fpsol2.i.1.col	496	11654	65	496	65	229	65	47	65
fpsol2.i.2.col	451	8691	30	451	30	90	30	38	30
fpsol2.i.3.col	425	8688	30	425	30	64	30	40	30
inithx.i.1.col	864	18707	54	864	54	347	54	43	54
inithx.i.2.col	645	13979	31	645	31	89	31	43	31
inithx.i.3.col	621	13969	31	621	31	64	31	45	31
multsol.i.1.col	197	3925	49	197	49	61	49	40	49
multsol.i.2.col	188	3885	31	188	32	17	32	41	31
multsol.i.3.col	184	3916	31	184	32	12	32	42	31
multsol.i.4.col	185	3946	31	185	31	12	31	42	31
multsol.i.5.col	186	3973	31	186	31	12	31	42	31
zeroin.i.1.col	211	4100	49	211	49	87	49	39	49
zeroin.i.2.col	211	3541	30	211	30	56	30	33	30
zeroin.i.3.col	206	3540	30	206	30	51	30	34	30

Table 4: Coloring watermarked real-life graphs.

The graph coloring problem has a lot of applications in real life, for example, the register allocation problem, the cache-line coloring problem, wavelength assignment in optical networks, and channel assignment in cellular systems. The instances of GC problems based on register allocation of variables in real codes and the optimal solutions are available at [13]. We watermark these graphs and then color them. The *fpsol2* and *inithx* instances are colored in 1 ~ 3 minutes,

while the others are all colored in less than 0.5 minute. Table 4 reports the details. The first four columns shows the characteristic of the original graph and the known optimal solution; the next two are for technique #1, showing the number of edges (information in bits) being embedded and the overhead; followed by two columns for technique #2, where the *Size* columns are the number of vertices in the selected MISes. The last two columns are for technique #3, where we compute the average edge probability of the original graph and add edges to keep this probability unchanged. Again, in almost all examples, there is no overhead.

7 Conclusion

Most watermarking techniques lack formal analysis and their effectiveness solely relies on the experiments. In this paper, we build the first theoretical framework for analyzing watermarking techniques. In particular, we select two of the most important criteria for any effective watermark, namely high credibility and low overhead, as the basis of our analysis. We propose three watermarking techniques for the graph coloring problem, which are provably capable to provide high credibility with at most 1-color overhead for large graphs. Asymptotic formulae are given on the amount of information that can be embedded into the graph without too much overhead as well as the amount of information that should be embedded to provide high credibility. Numerical data from simulation has been analyzed and confirms our results. Also, we watermark and then color a large range of graphs from random graphs, DIMACS challenge graph to graphs generated from real life problems. With the same amount of run-time as for the original graphs, for almost all instances, we obtain solutions of the same quality with no overhead.

References

- [1] H.Berghel and L.O’Gorman. *Protecting ownership rights through digital watermarking*. IEEE computer, 29(7): 101-103, 1996.
- [2] B.Bollobás. *Random Graphs*. Academic Press, London, 1985.
- [3] L.Boney, A.H.Tewfik, and K.N.Hamdy. *Digital watermark for audio signals*. International Conference on Multimedia Computing and Systems, pp. 473-480, 1996.
- [4] I.J.Cox, J.Kilian, T.Leighton, and T.Shamoon. *A secure, imperceptible yet perceptually salient, spread spectrum watermark for multimedia*. Southcon, pp. 192-197, 1996.
- [5] F. Hartung, and B. Girod, *Digital watermarking of rae and compressed video*. In Proceedings of the SPIE-The Internation Society for Optical Engineering, volume 2952, pages 205-213, 1996.
- [6] 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.
- [7] D.Kirovski and M.Potkonjak. *Efficient Coloring of a Large Spectrum of Graphs*. 35th Design Automation Conference Proceedings, pp. 427-432, 1998.
- [8] R. Nelson, and R.J. Wilson (Editors) *Graph Colourings*. Longman Scientific & Technical, Harlow,Essex, UK 1990.
- [9] C. Podilchuk, W. Zeng, *Perceptual watermarking of still images*. IEEE Workshop on Multimedia Signal Processing, pp. 363-368, 1997.
- [10] M.D.Swanson, B.Zhu, B.Chau, and A.H.Tewfik. *Object-based transparent video watermarking*. IEEE Workshop in Multimedia Signal Processing, pp. 369-374, 1997.
- [11] M.M.Yeung, F.C.Mintzer, G.W.Braudaway, and A.R.Rao. *Digital watermarking for high-quality imaging*. IEEE Workshop on Multimedia Signal Processing, pp. 357-362, 1997.
- [12] <http://dimacs.rutgers.edu/>
- [13] <http://mat.gsia.cmu.edu/COLOR/instances.html>

8 Appendices

8.1 Proof of Theorem 3.1

Theorem 3.1

Adding $k(n)$ edges to a random graph $G_{n,p}$, for almost all $G_{n,p}$, $\overline{\lim}_{n \rightarrow \infty} \chi' - \chi = \infty$ iff $k(n) \in \omega(n \log n)$.

Proof:

In the original graph $G_{n,p}$, let $b = \frac{1}{1-p}$ and $\chi = \lceil \frac{n}{2 \log_b n} \rceil$ as given by (3). After adding $k(n)$ extra edges, the edge probability increases to $p' = p + \frac{2k}{n(n-1)}$, and $b' = \frac{1}{1-p'}$, $\chi' = \lceil \frac{n}{2 \log_{b'} n} \rceil$.

$$\chi' - \chi = \lceil \frac{n}{2 \log_{b'} n} \rceil - \lceil \frac{n}{2 \log_b n} \rceil \geq \frac{n}{2 \log_{b'} n} - \frac{n}{2 \log_b n} - 1 = \log_n \left(\frac{b'}{b} \right)^{n/2} - 1$$

where $\frac{b'}{b} = \frac{1-p}{1-p'} = 1 + \frac{2k/n(n-1)}{(1-p)-2k/n(n-1)}$.

It is clear that $k(n) \in O(n^2)$, and further if $k(n) \in \Theta(n^2)$ then $\chi' - \chi \rightarrow \infty$ as $n \rightarrow \infty$. Therefore,

$$\text{and } \lim_{n \rightarrow \infty} \left(\frac{b'}{b} \right)^{\frac{n}{2}} = \lim_{n \rightarrow \infty} \left(1 + \frac{2}{1-p} \frac{k}{n(n-1)} \right)^{\frac{n}{2}} = \lim_{n \rightarrow \infty} e^{\frac{k}{(1-p)(n-1)}}$$

$$\overline{\lim}_{n \rightarrow \infty} \chi' - \chi \geq \lim_{n \rightarrow \infty} \log_n \left(\frac{b'}{b} \right)^{n/2} - 1 = \lim_{n \rightarrow \infty} \frac{k}{(1-p)(n-1) \ln n} - 1$$

So, if $k(n) \in \omega(n \log n)$, $\overline{\lim}_{n \rightarrow \infty} \chi' - \chi = \infty$.

On the other hand, since

$$\chi' - \chi = \lceil \frac{n}{2 \log_{b'} n} \rceil - \lceil \frac{n}{2 \log_b n} \rceil \leq \frac{n}{2 \log_{b'} n} + 1 - \frac{n}{2 \log_b n} = \log_n \left(\frac{b'}{b} \right)^{n/2} + 1$$

similarly, we can see if $k(n) \in O(n \log n)$, $\chi' - \chi$ will be bounded. \square

8.2 Proof of Theorem 3.3

Theorem 3.3 (arbitrarily high credibility)

Adding $k(n)$ edges to a random graph $G_{n,p}$, let \mathcal{E} be the event that a random solution to the original graph also satisfies all these $k(n)$ extra constraints. Then for almost all $G_{n,p}$, $\lim_{n \rightarrow \infty} \text{Prob}[\mathcal{E}] = 0$ if $k(n) \in \omega\left(\frac{n}{\log n}\right)$.

Proof:

The event \mathcal{E} is probabilistic equivalent to fixing a GC solution, then selecting $k(n)$ pairs of disconnected vertices and each pair do not have the same color. For random graph $G_{n,p}$, each vertex has $p(n-1)$ neighbors, and if the graph is color by $\chi = \lceil \frac{n}{2 \log_b n} \rceil$ colors as given by (3), in average there will be $\lfloor 2 \log_b n \rfloor$ vertices for each color. Hence, when we select two disconnected vertices, the probability that they have different colors is $1 - \frac{\lfloor 2 \log_b n \rfloor - 1}{q(n-1)}$. Assuming that $k(n)$ pairs of vertices are picked independently, then the probability that the vertices in each pair are of different colors is $\left(1 - \frac{\lfloor 2 \log_b n \rfloor - 1}{q(n-1)} \right)^{k(n)}$.

$$\begin{aligned}
\lim_{n \rightarrow \infty} \mathbf{Prob}[\mathcal{E}] &= \lim_{n \rightarrow \infty} \left(1 - \frac{\lfloor 2 \log_b n \rfloor - 1}{q(n-1)}\right)^{k(n)} \leq \lim_{n \rightarrow \infty} \left(1 - \frac{\log_b n}{qn}\right)^{k(n)} \\
&= \lim_{n \rightarrow \infty} \left(1 - \frac{\log_b n}{qn}\right)^{\frac{qn}{\log_b n} \frac{k(n) \log_b n}{qn}} = \lim_{n \rightarrow \infty} e^{-\frac{k(n) \log_b n}{qn}} \\
&= 0 \quad \left(\text{if } k(n) \in \omega\left(\frac{n}{\log n}\right) = \omega(\chi)\right)
\end{aligned}$$

□

8.3 Proof of Theorem 4.2

Theorem 4.2 (arbitrarily high credibility with 1-color overhead)

Given a random graph $G_{n,p}$, we select one MIS as in Figure 9. Let \mathcal{E} be the event that in a random solution, all vertices in this MIS have the same color and they are in the order as specified by Figure 4. Then $\lim_{n \rightarrow \infty} \mathbf{Prob}[\mathcal{E}] = 0$. Furthermore, this introduces at most 1-color overhead.

Proof:

For a random graph $G_{n,p}$, the technique in Figure 4 gives us a MIS of size $\log_b n$ by Lemma 4.1. Given a fixed solution to $G_{n,p}$, event \mathcal{E} has the same probability as: constructing all MISes of size $\log_b n$ with a specific order and one randomly picked MIS has all its $\log_b n$ vertices the same color.

From the Stirling formula:

$$k! = \left(\frac{k}{e}\right)^k \sqrt{2\pi k} e^{\alpha_k}$$

where $\frac{1}{12k+1} < \alpha_k < \frac{1}{12k}$, we have:

$$\begin{aligned}
\mathbf{Prob}[\mathcal{E}] &= \frac{1}{\frac{(nq) \cdot (nq^2) \cdot \dots \cdot (nq^{\log_b n - 1})}{(\log_b n)!}} = \left(\frac{1}{q}\right)^{\frac{\log_b n (\log_b n - 1)}{2}} \left(\frac{1}{n}\right)^{\log_b n - 1} (\log_b n)! \\
&= (b^{\log_b n})^{\frac{\log_b n - 1}{2}} \left(\frac{1}{n}\right)^{\log_b n - 1} (\log_b n)! = n^{\frac{1 - \log_b n}{2}} (\log_b n)! \\
&= n^{\frac{1 - \log_b n}{2}} \left(\frac{\log_b n}{e}\right)^{\log_b n} \sqrt{2\pi \log_b n} e^{\alpha_{\log_b n}} \\
&= \sqrt{2\pi n \log_b n} e^{\alpha_{\log_b n}} \left(\frac{\log_b n}{e\sqrt{n}}\right)^{\log_b n}
\end{aligned}$$

where $\frac{1}{12\log_b n + 1} \rightarrow 0 \quad (n \rightarrow \infty)$
 $< \alpha_{\log_b n} < \frac{1}{12\log_b n}$.

It costs exactly one color for the selected MIS, and coloring the remaining graph requires no more than the number of colors for the original graph. Therefore, this introduces at most one extra color overhead. □

8.4 Proof of Theorem 4.3

Theorem 4.3

Given a random graph $G_{n,p}$, if we select $k(n)$ MISes as in Figure 9, assign each MIS one color and color the rest of the graph, then the overhead is at most $k(n)$ and on average at least $\frac{k(n)}{2}$.

Proof:

The first part is trivial from the fact that χ is non-decreasing in terms of n .

By Lemma 4.1, the MIS is of size $\log_b n$. Assuming the message is random, after we cut this MIS from the original graph $G_{n,p}$, the remaining graph will still be random with $n' = n - \log_b n$ vertices and the same edge probability p . Therefore, from formula (4) in section 3.2, we need $\chi' = \lceil \frac{n'}{2 \log_b n'} \rceil$ colors to color this remaining graph, taking into account one more color for the selected MIS, we use a total of $\chi' + 1$ colors to color the original graph $G_{n,p}$.

$$\begin{aligned} \chi' + 1 - \chi &= \lceil \frac{n'}{2 \log_b n'} \rceil + 1 - \lceil \frac{n}{2 \log_b n} \rceil = \lceil \frac{n - \log_b n}{2 \log_b (n - \log_b n)} \rceil + 1 - \lceil \frac{n}{2 \log_b n} \rceil \\ &= \lceil (\frac{n}{2 \log_b n} - \frac{1}{2}) \log_{(n - \log_b n)} n \rceil + 1 - \lceil \frac{n}{2 \log_b n} \rceil \\ &\geq \lceil \frac{n}{2 \log_b n} - \frac{1}{2} \rceil + 1 - \lceil \frac{n}{2 \log_b n} \rceil \\ \text{since } \log_{(n - \log_b n)} n &> 1 \end{aligned}$$

For a uniformly distributed real number x , $\mathbf{Prob}(\lceil x - \frac{1}{2} \rceil = \lceil x \rceil) = 0.5$. Therefore, when we construct one MIS by Figure 9, we will introduce one extra color overhead with probability at least 50%. And when we construct two MISes, for sure we will introduce at least one-color-overhead since $\lceil \chi - 1 \rceil = \lceil \chi \rceil - 1$. In general, when $k(n)$ MISes are selected, the size of the remaining graph $n' \geq n - k \log_b n$ because the size of MIS decreases with the size of the graph. So

$$\chi' + k(n) - \chi \geq \lceil \frac{n}{2 \log_b n} - \frac{k(n)}{2} \rceil + 1 - \lceil \frac{n}{2 \log_b n} \rceil \sim \frac{k(n)}{2}$$

□

8.5 Pseudo-code for Technique #2

<p>Input: a graph $G(V, E)$, a message $M = m_0 m_1 \dots$</p> <p>Output: new graph with message M embedded</p> <p>Algorithm:</p> <pre> current_graph = original_graph $G(V, E)$; previous_graph = current_graph; MIS = ϕ; do { if (current_graph is empty) { current_graph = previous_graph - MIS; previous_graph = current_graph; report MIS and reset MIS = ϕ; } if (current_graph has more than 2 vertices or is connected) { find the vertex v corresponding to the next $\lceil \log_2 n \rceil$ bits of M, where n is the size of the current graph; cut v and all its neighbors from the current graph; current_graph = current_graph - $\{v$ and its neighbors$\}$; MIS = MIS + v; reorder the vertices in current_graph; advance message M. } else MIS = MIS + current_graph; } while (M is not empty) report the current_graph; </pre>
--

Figure 9: Pseudo code for watermarking technique # 2.