演習 3 (7/9)

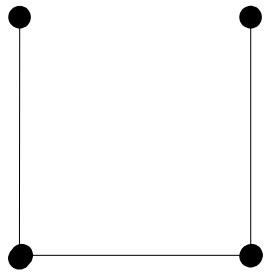
31010 佐藤秀明

概要

- レジスタ割り当てに対する Software Watermarking
  - QP 法
    - グラフ彩色問題に対して情報を埋め込む
    - アルゴリズムの「良さ」に関する定量的な議論
      - シグネチャの信頼性
      - オーバーヘッド
  - QPS 法
    - QP 法の改良版
  - QPS 法のレジスタ割り当てへの応用
  - 評価
    - 長所 / 短所
- 今後の目標

Random Graph

- Random Graph モデル $G_{n, p}$
  - n 個の頂点を持ち、かつそれらの各 2 点間に辺が存在する確率は等しくp であるようなグラフ
- 以下の議論ではこの $G_{n, p}$ を考える



$G_{4, 1/2}$の例

QP 法による埋め込み (1)

- 入力 : グラフ G(V, E) 、 メッセージ $M=m_0m_1...$
- 出力 :M を埋め込んでできたグラフの彩色結果
- アルゴリズム :

1. V の各頂点に順序をつける　$V=(v_0, v_1, ..., v_{n-1})$
2. 各 m_i について
   1)ある頂点 v_j を対応させ、さらに E において v_j に接続されていない最寄の 2 頂点 v_{j0}, v_{j1} を見つける
   2)$m_i=0$ なら辺 (v_j, v_{j0}) を、$m_i=1$ なら辺 (v_j, v_{j1}) を、E に追加
3. G(V, E) を彩色する

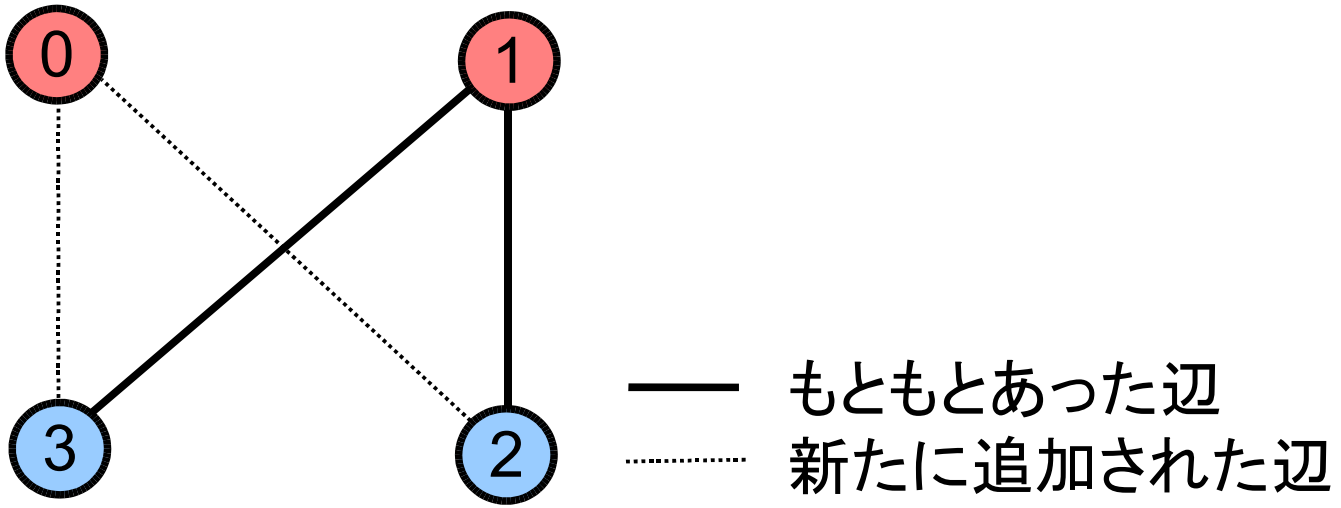
QP 法による埋め込み (2)

- 「ある頂点 v_j に接続されていない最寄の 2 頂点 v_{j0}, v_{j1} 」とは？

  1. $j < j0 < j1 \pmod n$ かつ
  2. (v_j, v_{j0}), (v_j, v_{j1}) の 2 辺は E に存在せず、かつ
  3. $j < k < j0$, $j0 < k < j1 \pmod n$ をみたすようなすべての k について、辺 (v_j, v_k) は E に存在する

QP 法による埋め込み (3)

- 例 :M=”10”



- m_0=1 は (v_j, v_{j0}, v_{j1}) =(0, 1, 2) に対応 … 辺 (0, 2) を追加
- m_0=0 は (v_j, v_{j0}, v_{j1}) =(3, 0, 2) に対応 … 辺 (3, 0) を追加

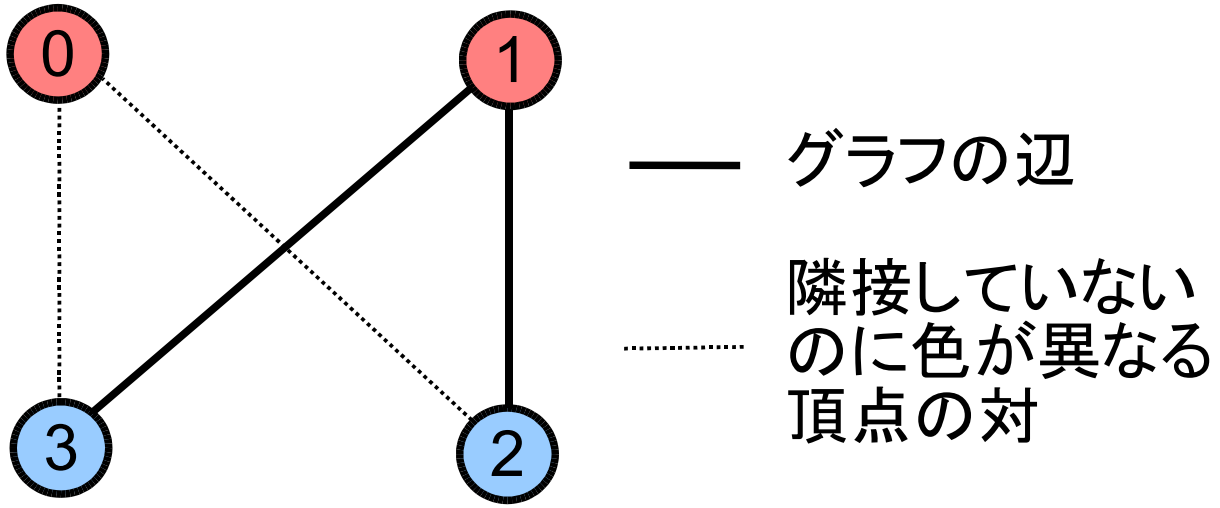
QP 法による情報の復元 (1)

- 入力 : グラフ G(V, E) 、埋め込みの際に得られた彩色結果
- 出力 : メッセージ
- アルゴリズム :

1. 辺で接続されていないのにもかかわらず異なる色に彩色されている 2 頂点 $(v_i, v_j)(j < j \bmod n)$ の組を探す
2. 1. の各対について、$i < k < j \pmod n$ をみたす頂点 v_k で v_i と接続されていないものの数が
   1) 0 個なら "0" を出力し、辺 (v_i, v_j) を追加する
   2) 1 個なら "1" を出力し、辺 (v_i, v_j) を追加する
   3) 2 個以上なら辺 (v_j, v_i) について 2. の操作をやり直す

QP 法による情報の復元 (2)

- 例 (先の例と同じ)



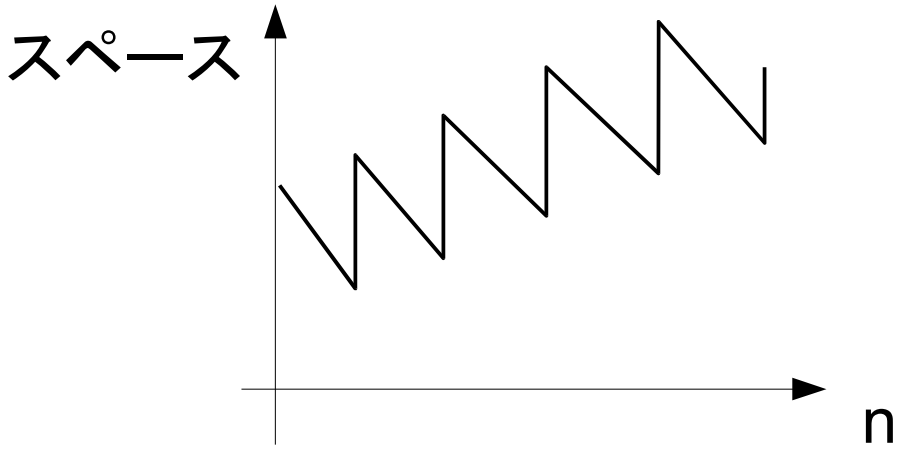
  - (0, 2) の間にあって頂点 0 とつながっていないものは頂点 1 のみだから、これは "1" を表す。
  - 一方、(0, 3) の間には頂点 0 と繋がっていないものが頂点 1 と 2 の 2 個あるから、これは (3, 0) とみなすべき。
  - 辺 (0, 2) を追加後、(3, 0) の関係より "0" を得る。
  - 以上より、復元されたメッセージは "10" である。

アルゴリズムの”良さ”をはかる尺度 (1)

- 信頼性
  - 「もとのグラフに対する彩色解を 1 つとってきたときに、それがメッセージ埋め込みの際に追加された制約 (= 辺) をたまたま満たす確率」は低くなければならない
    - もとのグラフ制約を満たす彩色解の集合 S の中で、メッセージを埋め込んだグラフの彩色解にもなっているようなものの数がたくさんあっては困る (復元できるメッセージが一意に決まらない)
  - QP 法では、追加された制約の数が $\Omega(n/(\log n))$ であれば、前述の確率は n がじゅうぶん大きいときに 0 に収束する。
    - 埋め込むメッセージの量にはある程度の大きさが必要。

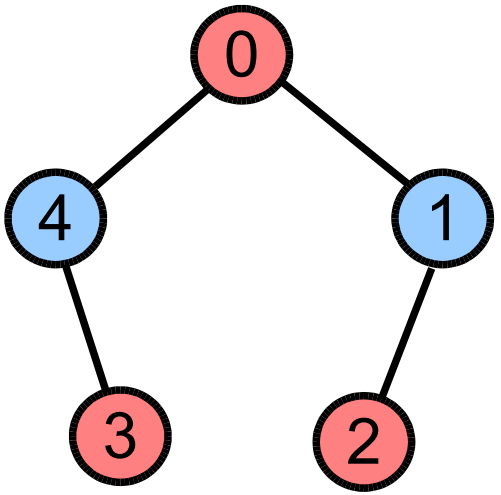
アルゴリズムの”良さ”をはかる尺度 (2)

- オーバーヘッド
  - 新たに追加された制約が、彩色に必要な色数をむやみやたらに大きくしないこと
  - QP 法では、追加された制約の数が O(n (log n)) であれば、色数の増加は高々 1 色に抑えられる。
  - 色数の増加を高々 1 に抑えながら情報を埋め込むことのできるスペースは、以下のようなぎざぎざのグラフで表される。

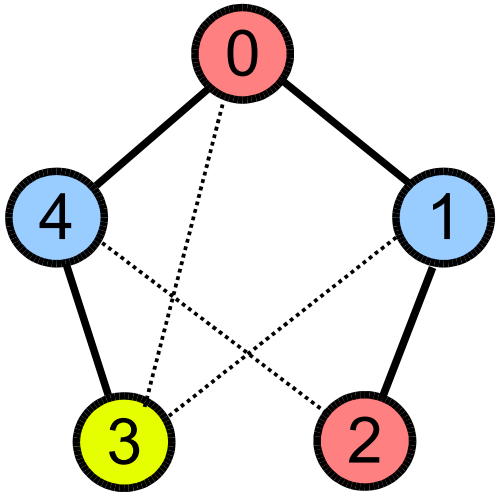


QP 法が失敗する場合

- 例 :



もとのグラフ



埋め込み後のグラフ

  - 埋め込んだメッセージは "101” ((0, 3), (1, 3), (2, 4))
  - しかしこれを復元してできるメッセージは "1001” ((0, 3), (1, 3), (2, 3), (4, 2))
  - 繋がっていない 2 頂点の色が異なるからといって、埋め込みの際に 2 者の間に辺が加えられたとは限らない

Triple

- Triple
  - 3 個の頂点の組で、どの 2 個も接続されていないもの
  - 埋め込みの際に、(v_j, v_{j0}, v_{j1}) の組として他の triples とは孤立した triple を選ぶようにすれば、新たな辺を加えてもそれによる彩色の変化は (v_j, v_{j0}, v_{j1}) 以外の頂点に影響を及ぼさない
- Colored-triple
  - 3 個とも同じ色に彩色された triple
  - (v_j, v_{j0}, v_{j1}) の組に colored-triple を選ぶようにすれば、埋め込み後のグラフで対応する (v_j', v_{j0}', v_{j1}') は、埋め込んだビットによって v_{j0}' または v_{j1}' が異なる色になっている

QPS 法

- 埋め込み
  - (v_j, v_{j0}, v_{j1}) を選ぶ際に、QP 法での条件に対して、新たに「(v_j, v_{j0}, v_{j1}) はどの 1 点も今回はじめて選ばれる colored-triple である」という条件を付け加える
- 復元
  - 埋め込みのときと全く同じ作業をなぞる
  - 埋め込み後のグラフにおいて、元のグラフの頂点 v_{j0} に対応する頂点 v_{j0}' の色が変わっていたら "0" を、v_{j1} に対応する頂点 v_{j1}' の色が変わっていたら "1" を、それぞれ出力する

QPS 法のレジスタ割り当てへの応用

- アルゴリズム
  - 埋め込み
    1. 変数の生存期間から干渉グラフ G を作る
    2. G について最適にレジスタ割り当て (グラフ彩色) する
    3. G にメッセージを表す辺を追加して G' を作る
    4. G' についてレジスタ割り当てをする
    5. 4. を使ってコードを書き下す
  - 復元
    1. コードから干渉グラフ G' を作る
    2. G' をコード中のレジスタ割り当てに沿って彩色する
    3. G' を最適にレジスタ割り当て (グラフ彩色) したものを G とする
    4. G と G' を比較しながらメッセージを復元する

実装の詳細 (論文より)

- Java bytecode をターゲットとして実装
- メッセージの先頭 8 ビットはメッセージの長さを表す
- 各変数には生存期間に基づいて順序をつけておく
- Tampering を難しくするために
  - クラスファイル上のすべてのメソッドに、遍くメッセージを埋め込む
    - 同じフレーズを何回も繰り返すようなメッセージを埋め込んだりする
  - 埋め込み時に使ったキーと同じキーが復元時に要求される
    - 各メソッドを処理する順序を決定する擬似乱数の種として使用

評価 (論文より)(1)

- 信頼性
  - QP 法に比べて制約がより厳しくなった
    - 正しい彩色解を一意に得ることができる
    - 正しいメッセージをエラーなく得ることができる
- 埋め込めるメッセージの量
  - 非常に少ない
    - 互いに孤立した colored-triples の数はそれほど多くない
  - 複雑なメソッドほど埋め込めるメッセージの量は少ない
    - 辺の数が密すぎる
  - インライン化を行うと少しはましになる

評価 (論文より)(2)

- 攻撃への耐性
  - あまりよくない
    - 逆コンパイルして再コンパイルするときれいさっぱり消える
    - 既にメッセージ A を埋め込み済みのコードに別のメッセージ B を重ねて埋め込むと、A は消えることがある
      - 他の Watermarking 手法との併用はできない
    - 意味を保ちながらプログラムを難読化・変換させるとメッセージが消えることがある
  - 変数の生存期間に依存した技術は比較的弱い

評価 (論文より)(3)

- Invisibility
  - メッセージを埋め込んだということ自体は非常に気づかれにくい
    - コードの追加を行わず、変化するのはレジスタ番号のみ
- Part Protection
  - コード全体にメッセージが埋め込まれているので、一部分が破壊されても他の部分から復元できる可能性が高い

評価 (論文より)(4)

- 計算時間
  - メッセージの埋め込み・復元は非常に遅い
    - レジスタ割り当てを何度も何度も行っている
    - コード全体にメッセージを埋め込もうとしている
  - コード自体の実行性能はほとんど変化なし
    - むしろ最適レジスタ割り当てを行ったおかげで前より速くなったりすることも

今後の目標

- Abstract watermarking(予定)
  - Static watermarking とも dynamic watermarking とも違うらしい

References

- Gang Qu and Miodrag Potkonjak, “Hiding signatures in graph coloring solutions”, in *Information Hiding*, pages 348-367, 1999.
- Ginger Myles and Christian Collberg, “Software watermarking through register allocation: Implementation, analysis, and attacks”, in *6th International Conference on Information Security and Cryptology*, 2003.
- Sandmark. http://www.cs.arizona.edu/sandmark/