6J-04

彩色自由度を用いたレジスタ割り当てに基づくソフトウェアウォーターマーキング

秋元　陽祐[†]　　平井　佑樹[‡]　　並木　美太郎[‡]　　金子　敬一[‡]

東京農工大学大学院工学府[†]　　東京農工大学大学院工学研究院[‡]

1. 研究背景と目的

デジタル市場の拡大により，近年デジタルコンテンツの違法コピー被害が広がっている．Business Software Alliance [1]の調査では，2013年の違法コピーによる損失額は全世界で 627 億ドルである．そのため，違法コピー対策への関心が高まり，デジタルコンテンツに対するウォーターマーク(電子透かし)の研究が盛んである．

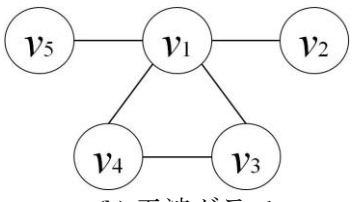
本研究では，ソフトウェアに対してウォーターマークを埋め込む技術のうち，「レジスタ割り当て」に基づく手法に着目する．これはソフトウェアを構成するプログラムの中間コードで実施する，変数のレジスタ割り当てに対して不可視の識別情報を埋め込むという手法である．本研究では既存の手法である PCC (Priority Color Change) アルゴリズム [2] を改良した IPCC (Improved PCC)アルゴリズムを提案する．

2. 先行研究

Chaitin ら[3]はグラフ彩色に基づくレジスタ割り当て手法を提案した．この手法は，プログラムに対する中間コード(図 1(a))に対し，各変数の生存区間を解析し，図 1(b)に示す干渉グラフを作成する．例えば，変数 v_3 の生存区間は変数 v_1 と v_4 の生存区間と重複するため，v_1 を表すノードと変数 v_1 と v_4 を表すノードをリンクで結ぶ．さらに，この干渉グラフに対し，(1) 隣接ノード同士は異なる色で彩色する，(2) できるだけ少ない色数で彩色する，という 2 条件を満たすようにグラフを彩色する．

$v_1 = 0$
$v_2 = 4$
$v_3 = v_1 + v_2$
$v_4 = v_1 + v_1$
$v_5 = v_3 + v_4$
$v_5 = v_5 + v_1$

(a) 中間コード



(b) 干渉グラフ

図 1 レジスタ割り当てのためのグラフ

各レジスタを固有の色で表現すれば，干渉グラフに対する「グラフ彩色問題」を解くことで，変数に対するレジスタの割り当てを決定できる．また，与えられた情報ビット列 $M = m_1m_2\ldots m_n$(シリアル番号などの識別情報など)に応じて彩色を，つまりレジスタ割り当てを変更することで，ウォーターマークを埋め込むことができる．この着想に基づき，QP アルゴリズム[4]や CC アルゴリズム[5]など，様々な手法が提案されている．

3. PCC アルゴリズム

PCC では，まず貪欲な彩色アルゴリズム GC (Graph Coloring)で彩色する．GC では，添字の小さい変数(図 1 の場合は v_1)から前述の 2 条件を満たすように彩色する．図 1 の場合では v_1 から v_5 まで順に，赤，青，青，緑，青のように彩色する．ただし，色にはそれぞれ番号（赤: 0，青: 1，緑: 2）がついており，番号の小さい色から彩色可能性を調べるものとする．次に，$m_i = 1$ なるノード v_i のみ色を変更することで情報を埋め込む．その際，各変数に対し，彩色自由度を計算する．

定義 1 干渉グラフ $G(V = \{v_1, v_2, \ldots, v_n\}, E)$，$G$ の彩色関数 C，埋め込む情報ビット列 $M = m_1m_2\ldots m_n$，および情報ビット $m_i = 1$ なるノード v_iに対し，色の集合$\{C(j)|\ v_j \notin N(v_i)$ または $m_j = 1$ $(1 \le j \ne i \le n)\}$を彩色可能集合，その要素数を彩色自由度と呼ぶ．ただし，$N(v_i)$は v_i の隣接ノード集合を，$C(j)$はノード v_j の色を表す．また，情報ビット m_iをノード v_iに埋め込むものとする．

定義 1 より，例えば，図 1 に示すグラフにメッセージ $M = 11001$ を埋め込む場合，v_1, v_2, v_5の彩色可能集合は，それぞれ，空，{赤, 緑}，{赤, 緑}となる．あるノードの彩色自由度が 1 の場合，そのノードは彩色可能集合の色以外では彩色できない．また，彩色自由度が 0 の場合，そのノードは未使用の色で彩色する必要がある．

PCC では，$m_i = 1$ に対応するノード v_iの色を彩色自由度の小さい(同じ場合は添字の小さい)ものから変更することで，メッセージ M を埋め込む．PCC は，既存アルゴリズムと比較して使用レジスタ数の削減を達成している．

Software Watermarking based on Register Allocation by using Degree of Freedom in Coloring

† Yosuke Akimoto, Graduate School of Engineering, Tokyo University of Agriculture and Technology

‡ Yuki Hirai, Mitaro Namiki, and Keiichi Kaneko, Institute of Engineering, Tokyo University of Agriculture and Technology

4. IPCC アルゴリズム

PCC では，ノード v_i の彩色自由度を計算する際，$m_j = 0$ なる隣接ノードの色集合$\{C(j) | v_j \in N(v_i), m_j = 0\}$を排除し，$m_j = 1$ なる隣接ノードの色集合は排除しない．これは，$m_j = 1$ なるノードはこれから異なる色で彩色されるためである．本研究では，$m_j = 1$ なるノードも考慮したアルゴリズム IPCC を提案することにより，さらなる使用レジスタ数の削減を狙う．

IPCC では，彩色自由度の小さいノードから色を変更にするという PCC の彩色順序に例外を設ける．具体的には変更対象ノードの隣接ノード数および全隣接ノードの色数に着目する．

定義 2　干渉グラフ $G(V = \{v_1, v_2, \ldots, v_n\}, E)$，$G$ の彩色関数 C，埋め込む情報ビット列 $M = m_1m_2\ldots m_n$，および情報ビット $m_i = 1$ なるノード v_i に対し，$|\{C(1), C(2), \ldots, C(n)\}| > |\{C(i)\}| + |\{C(j) | v_j \in N(v_i), m_j = 0\}| + |\{v_j | v_j \in N(v_i), m_j = 1\}|$を満たすものを例外と呼ぶ．

例外となるノードにおいて，現在使用されている色のうち，彩色可能集合に含まれないものの数は，現在使用されている色数を超えない．PCC では，各情報ビットを埋め込む際に，彩色可能集合を再計算する．しかし，例外となるノードの彩色自由度は 0 にならない．そのため，このノードの彩色を後回しにすることができる．図 2 に IPCC の疑似コードを示す．

5. 計算機実験

IPCC と PCC の性能を比較するため，計算機実験を行った．実験では，干渉グラフのノード数 n = 10, 20, 30，リンク数を，完全グラフのリンク数 $(n(n - 1) / 2)$の 0%から 100%まで 10 ポイント刻みで変化させたものを測定対象とした．ノード数とリンク数の各組に対し，ランダムに生成した情報ビットの埋め込みを 100 万回行い，各アルゴリズムの使用レジスタ数の平均値を比較した．

実験結果を図 3 に示す．図 3 は，IPCC が，PCC に対して，多くの場合で，増加レジスタを抑制できることを示している．

6. まとめ

本研究ではレジスタ割り当てに基づくソフトウェアウォーターマーキング手法である IPCC アルゴリズムを提案した．計算機実験の結果，既存の PCC アルゴリズムに対して，多くの場合で，使用レジスタ数を抑制できることを示した．

```
function IPCC(G(V = {v1, v2, ⋯, vn}, E), M = m1m2...mk)
begin
  if n < k then abort("Embedding Failed");
  C := GC(G);
  for j := 1 to k do m̂j := mj, exceptionj := 0;
  while m̂1m̂2...m̂k <> 00...0 begin
    count := 0;
    c := max1≤j≤n{C[j]};
    for j := 1 to n do
      if m̂j = 1 then begin
        Sj := {1, 2, ..., c} \ {C[j]} \ {C[j'] | (vj, vj') ∈ E
              and m̂j' = 0};
        if |{1, 2, ..., c}| > |{C[j]}| + |{C[j'] | (vj, vj') ∈ E
              and mj' = 0}| + |{ vj' | (vj, vj') ∈ E
              and mj' = 1}| then exceptionj := 1
        else count := count + 1
      end;
    i* := argmin1≤i≤k{|Si| | m̂i = 1};
    if ∃i* and exceptioni* = 0 then begin
      if |Si*| = 0 then C[i*] := c + 1
      else C[i*] := min{Si*};
      m̂i* := 0;
      while count > 0 begin
        if |Si*| = 0 then C[i*] := c + 1
        else C[i*] := min{Si*};
        m̂i* := 0;
        count := count - 1
      end
    end
  end;
  return C
end
```

図 2 IPCC アルゴリズム

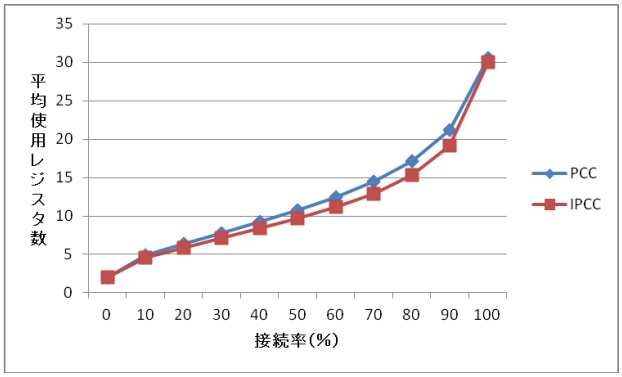


図 3 使用レジスタ数の比較（$n = 30$）

今後の課題は，使用レジスタ数を抑制しつつ，情報ビット列の高速な埋め込みを行うアルゴリズムの提案である．

参考文献

[1] Business Software Alliance, BSA Global Software Survey: The Compliance Gap, 2014.
[2] Y. Akimoto, Y. Hirai, and K. Kaneko: Software Watermarking based on Register Allocation by using Priority, ICT International Student Project Conference 2015, 4B-4, 2015.
[3] G. J. Chaitin, M. A. Auslander, A. K. Chandra et al., Register Allocation via Coloring, Computer Languages, Vol. 6, No. 1, pp. 47-57, 1981.
[4] G. Qu and M. Potkonjak, Analysis of Watermarking Techniques for Graph Coloring Problem, Proc. 1998 IEEE/ACM Int'l Conf. on Computer Aided Design, pp.190-193, 1998.
[5] H. Lee and K. Kaneko, New Approaches for Software Watermarking by Register Allocation, Proc. 9th ACIS Int'l Conf. on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, pp. 63-68, 2008.