社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－SLDM－130（8）
2007／5／11

DAGカバリング問題の下限とそれを用いた厳密アルゴリズムについて

松永 裕介†

† 九州大学大学院システム情報科学研究院
〒819–0395 福岡市西区元岡 744
E-mail: †matsunaga@c.csce.kyushu-u.ac.jp

あらまし セルライブラリ用のテクノロジマッピングやFPGA用のテクノロジマッピングはDAGカバリング問題として定式化できることが知られている．しかし，DAGカバリング問題はNP困難に属することも知られており，現在のところ，効率のよい厳密アルゴリズムは存在しない．本稿では与えられたDAGを木状グラフに変形することで，DAGカバリング問題の下限を求める手法を提案する．さらにこの下限の計算法の応用としてDAGカバリングの近似解法および厳密解法について延べる．ベンチマーク回路を用いた実験では2入力ゲート換算で数百ゲート規模の回路の場合には厳密解が数秒から数千秒で求められることを確認した．
キーワード DAGカバリング，テクノロジマッピング，FPGA

On lower bounds for DAG covering problem and their application to the exact algorithm

Yusuke MATSUNAGA†

† Faculty of Information Science and Electorical Engineering, Graduate School of Kyushu University
744 Motooka, Nishi-ku, Fukuoka, 819–0395 Japan
E-mail: †matsunaga@c.csce.kyushu-u.ac.jp

Abstract Technology mapping with cell libraries or FPGAs can be formulated as DAG covering problem, which is known to be NP-hard. So far, no efficient algorithms exist to solve it exactly. This paper proposes novel methods to compute the lower bound for DAG covering problem by transforming a DAG into a tree graph. A heuristic algorithm as well as an exact algorithm are also described using these lower bounds. The experiments with benchmark circuits show the effectiveness of the proposed methods. For several circuits with handreds gates, the proposed algorithm find the exact minimum solutions within seconds to thousands seconds.
Key words DAG covering, technology mapping, FPGA

1. はじめに

2. Introduction

セルライブラリ用のテクノロジマッピングやFPGA用のテクノロジマッピングはDAGカバリング問題として定式化できることが知られている[1], [2]．しかし，DAGカバリング問題はNP困難であることも知られており，現在のところ，効率よく厳密最小解を求めるアルゴリズムは知られていない．実用的な観点から考えると解の厳密性よりも処理時間が重視されるので，厳密アルゴリズムの開発は必須のものではないが，既存の近似手法の性能評価や新たな近似手法の開発のために厳密アルゴリズムの研究を行うことに意義はあると思われる．

本稿ではDAGカバリング問題に対する下限について考察を行い，効率のよい下限の計算手法を2種類提案する．これらは与えられたDAGを木状グラフに変形し，その木状グラフにおける最適解を求めることで，もとの問題の下限を計算するというものである．さらにこれらの下限の計算手法を用いることでduplication freeカバリング問題が高速に解けることを示す．

DAGカバリング問題に関しては必ずしも変形されたグラフ上で下限値を与える解がもとの問題に対する最適解とはならないが，実験結果によれば，この下限値に基づく解はよい近似解であることが示されている．また，この下限と下限に基づく近似解を利用して分枝限定法に基づく厳密アルゴリズムを開発を行い，数百ゲート規模の回路なら場合によっては数秒から数千秒で厳密最小解が求められことを確認した．

以降，第2章で基本的な定義や定理を示し，第3章で提案する下限の計算手法の説明を行う．その後，第4章で下限の計算手法と既存のヒューリスティックの関係に付いて延べる．第5章で実験結果を示し，その考察を述べた後，第6章でまとめを行う．

3. 基本的な定義

サブジェクトグラフ G とはカバリング問題の対象となっているDAG(directed acyclic graph：非巡回有向グラフ)であり，ノード $v \in V_G$ および枝 $e \in E_G$ を持つ．ノード v の**ファンイン (fan-ins)** はノード v の入力元となっているノードの集合であり $FI(v)$ で表される．ノード v の**ファンアウト (fan-outs))** とはノード v が出力しているノードの集合であり $FO(v)$ で表される．ノード v とノード $u \in FO(v)$ を結ぶ枝 $e(v,u)$ をノード v の**ファンアウトの枝 (fan-out edge)** と呼ぶ．ノード v の全てのファンアウトの枝の集合を $FE(v)$ で表す．**外部入力 (primary inputs)** とはファンインを持たないノードの集合で PI で表される．**外部出力 (primary outputs)** とはファンアウトを持たないノードの集合で PO で表される．ノード v の**推移的ファンイン (transitive fan-ins)** とは次式で定義されるノードの集合であり，$TFI(v)$ で表される．

$$TFI(v) = FI(v) \cup \bigcup_{u \in FI(v)} TFI(u)$$

ノード v の**推移的ファンインサブグラフ (transitive fan-in subgraph)** とは $TFI(v)$ から誘導される部分グラフのことで $G(v)$ で表される．

ノード v の推移的ファンインサブグラフ $G(v)$ に対する極小セパレータをノード v の**カット (cut)** と呼ぶ．つまり，PI の要素から v へいたる全ての経路が必ずカット c の要素を含んでおり(セパレータ)，また，各々のカットは他のカットを含まんでいない(極小性)．カット $c = \{u_1, u_2, \ldots\}$ の要素から v へいたる経路上に含まれるノードから誘導される部分グラフを**コーン (cone)** とよび，$m(v,c)$ で表す．コーン m に対して，ノード v を**根のノード (root node)** と呼び，$v = R(m)$ と表す．同様に c の要素 $u_1, u_2, \ldots$ を**葉のノード (leaf node)** と呼び，$\{u_1, u_2, \ldots\} = L(m)$ と表す．根のノードはコーンに含まれるが葉のノードはコーンに含まれないものとみなす．コーンの要素を $V(m)$ と表す．

サブジェクトグラフ $G = (V, E)$ 上のノード v とコーンの集合 $M = \{m_1, m_2, \ldots\}$ が与えられたとき，ノード v が**実現可能 (feasible)** であるとは，以下の様に定義される．

- v が外部入力の場合は常に実現可能である．
- v が外部入力でない場合には次の2つの条件が成り立つとき，実現可能である．

（1） v を根とするコーン $m \in M$ が存在する．

（2） コーン m のすべての葉のノード u_i が実現可能である．

サブジェクトグラフ G の全ての外部出力がコーン集合 M によって実現可能となるとき，コーン集合 M をサブジェクトグラフ G の**実現可能解 (feasible solution)** と呼ぶ．以降，簡単のため，「解」と略すことにする．

広義の **DAG カバリング (DAG covering)** 問題とはサブジェクトグラフ G と G 上のコーン集合 M^* が与えられたとき，ある尺度で最適となる解 $M \subseteqq M^*$ を求めることである．一般には，コーン m に対してコスト関数 $C(m)$ を用意しておいて，$\sum_{m \in M} C(m)$ が最小となるような解 M を求める問題を DAG カバリングと呼ぶことが多い．以降，本稿では特に断らない限り，コストが最小になる解を求める問題を DAG カバリングと呼ぶことにする．コスト関数は任意であるが，本稿では以下の2つの性質を満たすものと仮定する．

コスト関数の性質

- 2つのコーン m_a と m_b の間にノード集合としての包含関係 $m_a \subseteqq m_b$ が成り立つときには $C(m_a) \leqq C(m_b)$ が成り立つ．
- 2つのコーン m_a と m_b がグラフとして同形の場合には，$C(m_a) = C(m_b)$ が成り立つ．

また，コーン集合 $M = \{m_1, m_2, \ldots\}$ に対してコスト $C(M)$ を次の様に定義する．

$$C(M) = \sum_{m_i \in M} C(m_i)$$

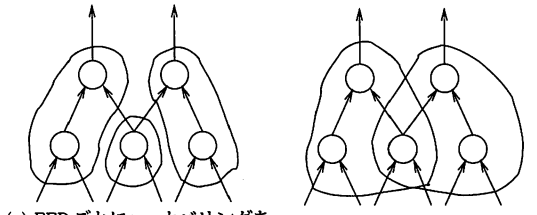
テクノロジマッピングにおいてマッピング対象の回路をサブジェクトグラフとみなし，ライブラリのセルもしくはFPGAの論理ブロックで実現可能な部分グラフをコーンとみなし，コスト関数をセルや論理ブロックの面積としておくことで，面積最小のマッピング結果を得る問題を DAG カバリング問題として定式化できる [1]〜[4]．セルライブラリのマッピングの場合，元となるコーン集合はセルの論理と一致する部分グラフであり，FPGA のマッピングの場合，葉のサイズが論理ブロックの入力数以下のコーン全てとなる．

しかし，DAG カバリング問題は NP 困難であることが知られており [2]，厳密最小解を効率よく求めるアルゴリズムは知られていない．一方，サブジェクトグラフが木状グラフの場合，カバリング問題は tree カバリングアルゴリズムで効率よく解けることが知られている [1], [2]．厳密には木状グラフとは全てのノードのファンアウト数が1以下のグラフを指すが，ここでは外部入力以外のノードのファンアウト数が1以下のグラフを木状グラフと呼んでいる(注1)．

DAG カバリングを近似的に解く手法として，与えられた DAG を木状の部分グラフに分割し，その各々の部分グラフに対して tree カバリングを適用するというものが提案されている．以降，この木状の部分グラフを **FFR(fanout free region)** と呼ぶことにする．FFR ごとに tree カバリングを適用した結果をつなぎ合わせた物は元の DAG カバリングの解であることは自明であるが，個々の tree カバリングの最適解からもとの DAG カバリングの最適解が得られる保証はない．図1のよう

(注1)：そのようなグラフを leaf-DAG と呼ぶ場合もある．

に FFR を跨いだコーンを用いた場合には (b) に示すように 2 つのコーンでカバーすることができるが，FFR に分割してしまった場合には 3 つの FFR にひとつずつのコーンが必要となる．FPGA の様に一定の入力数以下のコーンが同一のコストとなるような問題の場合には，(a) よりも (b) のほうがよい解となる．



(a) FFRごとにtreeカバリングをおこなった結果　(b) DAGカバリングの結果

図 1　tree カバリングと DAG カバリング

4. DAG カバリング問題の下限

DAG カバリング問題の効率的な厳密解法やヒューリスティックを考える上で重要なものとして下限値の計算が挙げられる．もしも，厳密最小解に近い下限値を計算することができれば分枝限定法に基づく探索アルゴリズムで探索すべき解空間を大幅に削減することができる．しかし，過去のテクノロジマッピング関連の論文において DAG カバリング問題の下限に関して論じられたものは皆無であると言える．これは，DAG カバリングを厳密に解くことの実用上の意義があまりないと考えられていることに起因すると思われるが，シャープな下限値を求めておくことはヒューリスティックの性能評価にも用いることができるため，実用上も有益であると思われる．

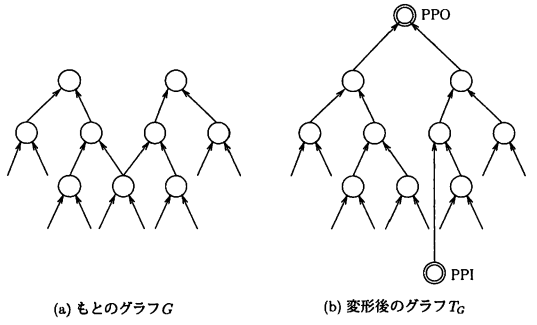
ここでは，2 種類の下限値の計算方法を提案する．この 2 つの手法とも，問題を木状グラフに対する tree カバリングに変換し，それを解くことで下限値を得るというものである．tree カバリングはグラフサイズおよびコーン集合のサイズに比例した手間で処理を行うことができるため非常に効率がよい．

4.1 枝の削除による木状グラフへの変形にもとづく下限値

前述の様にただ単に DAG を FFR ごとに分割し，その各々に tree カバリングを適用するだけでは役に立つ情報は得られないので，少し工夫した変形を行う．具体的には与えられたサブジェクトグラフ G に対して次のような操作を施して新たなグラフ T_G を生成する．

木状グラフへの変形処理 (1)

- G の外部出力が 2 つ以上あるときは，疑似外部出力ノード PPO を追加し，全ての外部出力から PPO へ枝を張る．元の外部出力を葉とし，PPO を根とするコーンのコストは常に 0 としておく．
- 2 つ以上のファンアウトを持つノード v に対して，ただ一つのファンアウトのみを残して，他の枝を削除する．削除された枝の代わりに疑似入力ノード PPI からの枝を追加する．



(a) もとのグラフ G　(b) 変形後のグラフ T_G

図 2　グラフの変形処理 (1)

図 2 にこの変形処理の例を示す．

このようにして得られたグラフ T_G は外部入力 (および疑似外部入力) 以外の全てのノードのファンアウトが 1 以下であるので，最適なカバリング結果は tree カバリングアルゴリズムを用いて解くことができる．さらに重要なことは，T_G に対する最適解は G の最適カバリングの下限になっているということである．

[**定理** 1]　T_G の最適カバリングのコスト値は，G の最適カバリングのコスト値と等しいかより小さい．

証明:　G の解を M とすると以下の処理にしたがって M から T_G の解 M' を作ることができる．

- M に外部出力を葉として PPO を根とするコーンを追加する．このコーンのコストは 0 である．
- M の要素のうち，T_G で存在しない枝 (削除された枝) を含むコーン m を削除し，該当部分を PPI からの枝に置き換えたコーン m' を追加する．コスト関数の条件として $C(m') \leqq C(m)$ が常に成り立っている．

この時，明らかに $C(M') \leqq C(M)$ である．よって G に対する最適解 $\tilde{M}$ が与えられたとき対応する T_G の解 $\tilde{M}'$ が存在して，$C(\tilde{M}') \leqq C(\tilde{M})$ となる．T_G の最適解のコストは必ず $C(\tilde{M}')$ と等しいかより小さいので，T_G の最適解のコストは G の最適解のコストと等しいかより小さいことが示された．　□

この下限値の計算方法は非常にシンプルであり実装も容易である．問題は，複数のファンアウトのうち，どの枝を残すかということである．実際，このファンアウトの枝の選びかたによって T_G の形が変わり，結果として下限値が異なる場合がある．ただし，最適解の値を最大化するようなファンアウトの選び方を求める問題は単純なアルゴリズムで求めることはできないと予想され，また，下限値の計算に計算複雑度の高いアルゴリズムを用いることはそもそも無意味であると思われるので，現在はランダムに枝を選択する処理を複数回繰り返し，そのなかの最大値を記録することで下限値としている．

4.2 部分グラフの複製による木状グラフへの変形に基づく下限値

DAG に対して，2 つ以上のファンアウトを持つノードを根とする部分グラフを複製することで外部入力以外のノードのファ

ンアウト数を 1 以下にすることができる．この変形された木状グラフに tree カバリングを適用し最適解を計算したところでその値と元のグラフの最適カバリングの値の間に関連はない．変形されたグラフに対する最適解では複製された部分グラフに関するコストが重複して数えられているからである．そこで，複製された部分グラフに関してはそのコストを複製された個数に応じて割ることで全体のコストを元のグラフのコストと同程度にする補正方法が考えられる．具体的には以下の処理で木状グラフ WT_G への変形およびコスト関数 WC の計算を行う．

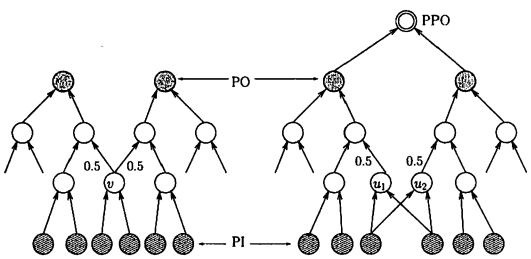
木状グラフへの変形処理 (2)

- G の外部出力が 2 つ以上あるときは，疑似外部出力ノード PPO を追加し，全ての外部出力から PPO へ枝を張る．元の外部出力を葉とし，PPO を根とするコーンのコストは常に 0 としておく．
- 各ノード v に対して，そのファンアウトの枝 e に重み $W(e)$ を付ける．ただし，重み $W(e)$ は以下の条件を満たすように割り当てる．

$$\sum_{e \in FE(v)} W(e) = 1$$

- 2 つ以上のファンアウトを持つノード v に対して，v の推移的ファンインサブグラフの複製を行う．この処理を 2 つ以上のファンアウトを持つノードがなくなるまで繰り返す．結果のグラフを WT_G とする．
- WT_G 上の各ノード u から PPO へ至る経路を $P = (p_1 = u, p_2, p_3, \ldots, p_n = PPO)$ で表すとき (経路はただ一つのみ存在する)，経路の重み $W(P) = \times_{i=1}^{n-1} W(e(p_i, p_{i+1}))$ を計算し，それをノード u の重み $W(u)$ とする．
- WT_G 上のノード u を根とするコーン m のコストを $WC(m) = C(l) \times W(u)$ とする．ただし，l は m に対応する G 上のコーンである．

図 3 にこの変形処理の例を示す．



(a) もとのグラフ G　　(b) 変形後のグラフ WT_G

図 3　グラフの変形処理 (2)

例えば図 3(a) 中のノード v の 2 つのファンアウトの枝の重みをそれぞれ 0.5 と 0.5 だとすると，同図 (b) のノード u_1 と u_2 をカバーするコーンのコストはそれぞれ 0.5 倍されることになる．

直感的にはわかりにくいが，WC をコスト関数とした WT_G の最適カバリングの値ももとの G の最適カバリングに対する下限となっている．

[**補題** 1]　G 上のノード v に対応する WT_G 上のノードを $\{u_1, u_2, \ldots\}$ とすると，$\sum W(u_i) = 1$ である．

証明:

□

[**定理** 2]　WC をコスト関数とした WT_G の最適カバリングのコスト値は C をコスト関数とした G の最適カバリングのコスト値と等しいかより小さい．

証明:　G に対するカバリングの解 M の各々の要素 $m \in M$ から対応する WT_G 上のコーンを求めることができる．ただし，G 上のノード v が WT_G 上では複数のノードに対応している場合があるのでそのようなコーンが複数存在する場合がある．そこで，m に対応する WT_G 上のコーンを $\{m'_1, m'_2, \ldots\}$ と表すことにする．このような対応付けを行った結果のコーン集合を M' とすると，M' が WT_G の解となっていること実現可能性の定義からは自明である．また，m の根のノードを v として，v に対応する WT_G 上のノード集合を $\{u_1, u_2, \ldots\}$ すると，Lemma 1 より $\sum W(u_i) = 1$ が成り立っている．一方，$WC(m'_i) = C(m) \times W(u_i)$ より

$$\begin{aligned} \sum WC(m'_i) &= \sum C(m) \times W(u_i) \\ &= C(m) \times \sum W(u_i) \\ &= C(m) \end{aligned}$$

が成り立つ．これより $WC(M') = C(M)$ となる．つまり，G に対するカバリングの解 M から作られた WT_G のカバリングの解 M' のコストは M のコストと等しいことが示される．明らかに，G に対する最適カバリングの解 $\tilde{M}$ のコストと同じ値を持つ WT_G に対するカバリングの解が存在することになるので，WC をコスト関数として WT_G の最適カバリングのコスト値は C をコスト関数とした G の最適カバリングのコスト値と等しいかより小さいことが示された．　□

こちらのグラフ変形も単純ではあるが，複製される部分グラフ中に 2 つ以上のファンアウトをもつノードがあれば複製処理を再帰的に繰り返す必要があるため，変形後のグラフのノード数は最悪，もとのグラフのノード数の指数乗に比例する可能性がある．そのため，この変形をそのまま適用することは効率的でない．実際には，複製された部分グラフを別々に扱う必要はなく，また，$WC(v)$ を明示的に容易する必要もない．同形のグラフに対するカバリング結果は同一なので使いまわすことが可能である．そこで，実際には元のグラフ G 上の枝に前述の重み $W(e)$ を付加した重み付きサブジェクトグラフに対して図 4 のアルゴリズムを適用することで下限の計算を行うことができる．

アルゴリズムの全体の流れは tree カバリングとまったく同じである．唯一異なっているのが 6 行目の c の計算で，通常の tree カバリングの場合なら c に u のコストをそのまま足し合わ

```
weighted_tree_covering(SubjectGraph G, Set of Cones M*) {
1:  foreach v ∈ V_G from PI to PO {
2:      ĉ ← ∞
3:      foreach m ∈ M* whose root is v {
4:          c ← C(m)
5:          foreach u ∈ L(m) {
6:              c = c + CostOf(u) × W(u, v)
            }
7:          if ĉ > c {
8:              ĉ ← c
            }
        }
9:      CostOf(v) ← ĉ
    }
10: c_all ← Σ_{v∈PO} CostOf(v)
11: return c_all
}
```

図 4　下限値計算のアルゴリズム

せるのに対して，こちらは $W(u,v)$ という重みを掛けている．$W(u,v)$ は u から v へいたる経路 (複数の経路があり得る) 上の枝の重みを掛け合わせたものである．そのコストが最終的には外部出力まで伝搬してゆくが，その場合には u から外部出力へいたる経路上の枝の重みを全て掛け合わせたものの和を計算することになり，前述の変形処理 (2) で定義したコストの計算方法と一致する値を実際には部分グラフの複製なしに求めることができる．

このアルゴリズムの計算複雑度は比較的容易に求めることができる．図 4 には 3 重の for ループがあるが，結局，4 行目から 8 行目までの処理は全てのコーンに対してただ一回だけ実行されることがわかる．5 行めと 6 行目の for ループは各々のコーンの葉の数だけ回ることになるが，テクノロジマッピングの場合，コーンの入力数はある定数値 K 以下であると仮定することができるため，この部分のループの反復回数は $O(1)$ となる．$W(u,v)$ の計算複雑度はコーン m のサイズに比例するが，コーンのサイズもある定数 L 以下であると仮定すれば(注2)5 行目と 6 行目の処理の計算複雑度は $O(1)$ となり，全体の計算複雑度は $O(|M^*|)$ となる．つまり，tree カバリングと同程度の計算の手間でこの下限値の計算が行えることがわかる．

4.1 節で説明した下限の計算方法と本 4.2 節で説明した下限の計算方法は一見，異なるもののように見えるが，実は前節の方法は本節の方法の特殊なケースと見なすことができる．具体的には，本節の方法で，2 つ以上のファンアウトを持つノードに対して，ファンアウトの枝に重みを割り当てる際に，ただひとつの枝の重みを 1 にして残りの枝の重みを 0 にした場合の結果が前節の方法の結果と完全に一致する．そのような意味からもこの重みづきサブジェクトグラフを用いた下限の計算方法は DAG カバリングの下限に対する汎用性のある枠組であると言うことができる．前節の場合と同様に，枝の重みをどのように割り当てるかによって計算される下限値は異なることが予想されるが，現在のところあまり検討を行っていないので，実験ではファンアウト数に応じて均等に重みを分散したものを用いている．

(注2)：コーンの葉の数が K 以下なので，意味のある論理回路であればコーンのサイズは K の関数 (ただし指数) となるので定数と見なせる．

5. 下限の計算と既存手法

本節では提案した下限の計算方法と既存手法との関係について論ずる．

5.1 Duplication Free カバリングとの関係

Duplication Free カバリング (DF カバリング) とはカバリング結果のコーンが互いに重ならないようなカバリングを指す [5]〜[7]．このうち文献 [6] は近似手法であり，文献 [5], [7] の手法は厳密解法となっている．サブジェクトグラフ G 上のコーン m に対して，葉のノード u から外部出力へいたるいかなる経路も根のノード v を通る時，そのようなコーンを Duplication Free コーン (DF コーン) と呼ぶ．以降，DF コーンの全集合を M^*_{DF} で表すものとする．明らかに，DF カバリングの解は DF コーンのみから構成されることがわかるので M^*_{DF} を用いて tree カバリングと同様の処理を行えば最適な DF カバリングを得ることができるように思われるが，実際には DAG カバリングと同様に部分グラフのコストを重複して数えることがあるため，単純なアルゴリズムでは最適解を求めることができない．今回提案した 2 つの下限計算法は，この種の問題をグラフ変形によって回避しているので，部分グラフのコストを重複して数えることはない，ただし，単純に考えると提案手法は下限値を計算しているのであって，それが可能解でなければ意味のないことになってしまう．幸いなことに，DF カバリング問題に関する限り，提案手法で計算される下限値は厳密に DF カバリングの最適解と一致する．また，変形されたグラフ上での最適解からもとの DF カバリングの最適解を簡単に生成することが可能である．計算複雑度は $O(|M^*_{DF}|)$ であり，これは現在知られているアルゴリズムの中でもっとも効率のよいアルゴリズムとなっている．詳細は省略するが，文献 [5] と文献 [7] のアルゴリズムは，まったく異なったアプローチをとっているが計算複雑度は $O(|M^*_{DF}| \times |V_G|)$ となっている．

5.2 ファンアウト数を考慮した DAG カバリングヒューリスティック

論文では明示的にかかれていないが UC Berkeley で開発された論理合成プログラム SIS のテクノロジマッパは DAG カバリングのヒューリスティックとして，tree カバリングの様に入力側からコストを計算していく際に，ノード v のコストをノード v のファンアウト数で割る処理を行っている．同様なヒューリスティックは DAOmap [8] でも用いられている．

これらのヒューリスティックの基本的なアイデアは 4.2 節で述べた重み付きサブジェクトグラフに対するカバリングと似たものであるが，これらの手法はノード v そのもののファンアウト数のみに着目しており，コーン内部のファンアウトを無視している．例えば，図 1 に対するカバリングの際に，同図 (a) のようにコーンの切れ目がちょうど FFR の切れ目と等しい場合にはこのヒューリスティックはうまく働くが，同図 (b) のよう

にコーン内部に2つ以上のファンアウトを含んでいる場合にはまったく考慮できないという欠点がある．

さらに，これらはただのヒューリスティックであり，その妥当性に対する理論的な考察はあまりなされていなかった．実際には前述のように，ファンアウト数に基づく重み付けを行ったサブジェクトグラフ上での最適カバリングはもとの問題の下限となるので，その下限値を与えるようなコーンを選択するヒューリスティックに理論的な根拠を加えることができる．

6. 実験および考察

本稿で提案した下限の計算法および，それを利用したヒューリスティックおよび厳密解法をFPGAのテクノロジマッピング問題に適用した実験を以下のようにおこなった．

（1） MCNCのベンチマーク回路を2入力ノードからなるネットワークに分解する．

（2） 分解されたネットワークに対して，4入力以下のコーンを全て列挙する．

（3） ネットワークをサブジェクトグラフ G，列挙されたコーン集合を M^* としてDAGカバリング問題を解く．ただし，全てのコーン m に対して $C(m) = 1$ とする．

表1に実験結果を示す．表中の「LB1(1)」および「LB1(10)」

表1 実験結果

回路名	LB1		LB2	DF	近似解	厳密解	
	1回	10回				コスト	計算時間
9symml	75	76	75	77	77	77	0.02
C1355	71	73	74	266	74	74	0.08
C1908	153	152	163	186	185	167	3320.98
C2670	189	198	204	243	221	211	>3600
C3540	347	360	381	466	443	407	>3600
C432	70	71	73	88	74	73	0.02
C499	71	74	74	82	74	74	0.01
C5315	516	522	535	588	592	580	>3600
C6288	466	437	481	1424	603	546	>3600
C7552	601	597	646	801	769	745	>3600
C880	104	107	109	136	125	114	13.39
apex6	231	233	235	269	247	244	1543.13
apex7	93	98	98	110	106	103	0.60
b9	49	49	49	53	49	49	0.01
des	2069	2090	2138	2267	2178	2139	6.47
f51m	103	103	103	103	103	103	0.01
rot	431	435	434	443	456	442	75.43
too_large	5092	5095	5095	5100	5098	5098	1884.26
vda	469	471	481	486	491	484	29.66
Total	11200	11241	11448	13188	11965	11730	

は4.1節で述べた計算法による下限値で，LB1(1)は一回のみ計測をおこなったもの，LB1(10)はランダムに残す枝を変えて下限を求める処理を10回繰り返した時の最大値を示している．「LB2」は4.2節で述べた計算法による下限値である．

「DF」はLB1(1)を用いたDFカバリングの結果，「近似解」はLB2を用いたDAGカバリングの近似解の結果，「厳密解」は厳密解のコストおよび計算時間を示している(単位は秒)．厳密解はグラフ中の2つ以上ファンアウトを持つノードに対して，そのノードでグラフを分割するかそのノードを複製するかの二者択一に基づく分枝限定法にLB2の下限とLB2を用いた近似解を組み合わせたものである．使用計算機はPentium-4 3.40GHz (メモリ1GB)である．厳密解以外の計算時間は大きな回路でも1秒程度なので省略した．なお，厳密解は1時間で処理を打ち切り，その時点で求まっている最小値を解としている．

結果は回路によってまちまちであるが，LB2の下限値と厳密解が一致する場合もいくつか見られ，また，LB2の下限に基づく近似解が厳密解に一致する場合もあり，特にLB2の下限の計算方法が効果的であることがわかる．LB1に関してはLB2の極端な重み割り当てと見なせることから予想できたように概ねLB2よりも低い値が出ている．ただし，いくつかの例では10回試行したLB1のほうがより高い値を出している場合もあるのでより詳細な解析が必要と思われる．参考までにDFカバリングの結果を載せたがDFカバリングはDAGカバリングの近似解としては不適切であると言える．

7. おわりに

本稿ではサブジェクトグラフを変形した木状グラフ状で最適解を計算することによって下限の計算を行う手法を提案した．また，この下限値に対応する解がよい近似解となっていることを実験的に確認した．これらの下限および近似解は分枝限定法に基づく厳密アルゴリズムが効果的な枝刈りを行うのに役に立っていると思われるが，現在の実装では分割および複製を行うノードの選択が単純なものなので，より効率的な厳密アルゴリズムの検討を進める必要がある．また，遅延(段数)を考慮したテクノロジマッピングへの応用が今後の課題である．

文　献

[1] K. Keutzer: "DAGON: Technology Binding and Local Optimization by DAG Matching", Proceedings of 24th Design Automation Conference, pp. 341–347 (1987).

[2] R. Rudell: "Logic Synthesis for VLSI Design", PhD thesis, U. C. Berkeley (1989).

[3] K. C. Chen, J. Cong, Y. Ding, A. B. Kahng and P. Trajmar: "DAG-map: Graph-based FPGA technology mapping for delay optimization", IEEE Design and Test of Computer, pp. 7–20 (1992).

[4] J. Cong and Y. Ding: "FlowMap: An optimal technology mapping algorithm for delay optimization in lookup-table based FPGA design", IEEE Transactions on Computer-Aided Design, **13**, 1, pp. 1–12 (1994).

[5] J. Cong and Y. Ding: "On area/depth trade-off in LUT-based FPGA technology mapping", the 30th ACM/IEEE Design Automation Conference, pp. 213–218 (1993).

[6] C.-C. Kao and Y.-T. Lai: "Area-minimal algorithm for LUT-based FPGA technology mapping with duplication-free restriction", ASP-DAC 2004, pp. 719–724 (2004).

[7] 松永："テクノロジマッピングにおけるDAG被覆アルゴリズムについて"，信学技報，第VLD2004-2巻 (2004).

[8] D. Chen and J. Cong: "DAOmap: A depth-optimal area optimization mapping algorithm for FPGA designs", Proceedings of IEEE International Conference on Computer Aided Design, pp. 752–759 (2004).