設 計 自 動 化 89−16
(1998. 9. 22)

大規模論理回路分割に関する一手法

上土井 陽子† 若林 真一‡ 吉田 典可†

† 広島市立大学情報科学部
〒731-3194 広島市安佐南区大塚東三丁目 4-1

‡ 広島大学工学部
〒739-8527 東広島市鏡山一丁目 4 番 1 号

あらまし 論理回路の分割問題は一般にNP困難であることが知られている．現在までにいくつかのヒューリスティックアルゴリズムが開発され，節点の縮約を段階的に行う論理回路分割手法が大規模なサイズの入力に対して有効であることが示されている．節点の縮約による手法では複雑な変換操作や多くの記憶領域を必要とするため，さらなる大規模な論理回路において有効であるかどうかは明らかでない．本稿では，著者らがすでに提案している組み立て法に基づく高速ハイパーグラフヒューリスティックアルゴリズムを改良した手法を提案する．また，提案高速組み立て法を遺伝的アルゴリズムに組み込んだ手法について考察する．大規模なベンチマークを入力としたシミュレーション実験により提案手法と従来手法の性能を比較する．

キーワード 論理回路分割，ハイパーグラフ分割問題，組み立て法，遺伝的アルゴリズム

A Fast Constructive Method for Hypergraph Partitioning

Yoko Kamidoi† Shin'ichi Wakabayashi‡ Noriyoshi Yoshida†

† Faculty of Information Sciences, Hiroshima City University
3-4-1, Ohzuka-Higashi, Asaminami-ku, Hiroshima 731-3194 JAPAN

‡ Faculty of Engineering, Hiroshima University
4-1, Kagamiyama 1 chome, Higashi-Hiroshima 739-8527 JAPAN

Abstract The circuit partitioning problem is known as NP-hard. Recent work has shown the promise of multilevel approaches for partitioning large hypergraphs. Multilevel partitioning methods are besed on the iterative improvement FM method. Multilevel partitioning recursively clusters the instance until its size becomes smaller than a given threshold, then unclusters the instance while applying a partitioning refinement algorithm. These methods need large computational space and long computation time for partitioning large hypergraphs. In this paper, we present a fast constructive method for partitioning hypergraphs. We have developed a hybrid algorithm of a genetic algorithm and the proposed constructive method. We evaluate the performance both in terms of the size of hyperedge cut on the bi-partition as well as run time on benchmark suites.

key words VLSI circuit partitioning, hypergraph partitioning, constructive method, genetic algorithm

1 まえがき

近年，論理回路の大規模化に伴いレイアウト設計において論理回路分割が重要な問題となっている．最近の例では数十万セルからなる論理回路の分割問題を扱う必要性がある [4]．この分割問題はハイパーグラフの分割問題として定式化される．ハイパーグラフ2分割問題は入力ハイパーグラフの節点集合を節点の重みの和がほぼ等しい2つの節点部分集合に分割して出力する．ここで，2つの節点集合間をまたがるハイパー枝の数の最小化が目的となる．

一般的にハイパーグラフ2分割問題はNP困難である [7]．従って，多くのヒューリスティックアルゴリズムが開発されている．現在までに開発されている論理回路分割手法は反復改良法と組み立て法の2つの手法に大きく分類される．反復改良法はある初期2分割が与えられたときにカットされるハイパー枝の数を減らすように節点の移動，交換を繰り返し行うことにより改良する方法である．組み立て法は入力としてハイパーグラフが与えられたとき，簡単な手続きにより2分割を構成する手法である．一般に組み立て法は高速であるが，解の質が悪いため反復改良法の初期解を構成するために適用される．

反復改良法の代表的な手法として，Kernighan-Linによる手法 (KL法)[13] やFiduccia-Mattheysesによる手法 (FM法)[6] が提案されている．FM法とKL法はその実現の容易さ，大規模でないハイパーグラフに対しては効率よく良質な2分割を出力することなどから一般的に用いられている．しかしながら，FM法やKL法は節点の移動，交換に基づいているため，大規模な入力ハイパーグラフに対して良質な2分割を求めることが困難であると考えられている [11]．さらに，同じゲインを持つ節点が複数ある場合，その中の優先順位が決定できない．また，ゲインの値に反映されないカットをまたがるハイパー枝が多く存在するため，大局的な改良が望めない．従って，FM法，KL法に対する多くの拡張が開発された．

Krishnamurthy[14] は同じゲインを持つ節点の優先順位を決定するために先読み (Look Ahead) 機構をFM法に導入した．Dutt と Deng[5] はカットのどちらかに2節点以上の節点を持つハイパー枝の新しいゲインの計算方法を提案した．

近年，multilevel partitioning techniques [11] [10] [2] [3] なる新しい分野が開発された．これらの手法では段階的にハイパーグラフの節点集合を縮約していき，サイズの小さくなった縮約グラフに対してランダムな初期分割からFM法などの反復改良法を適用し2分割を求める．縮約を段階的に元に戻し，それに伴い2分割も対応する2分割に変換され，FM法等により段階的に改良する手法である．Karypis ら [11][10] は良質なハイパー枝の縮約方法を開発した．ランダムネスと強力な縮約方法を組み合わせることにより大規模な入力ハイパーグラフに対して高速で有効な手法 hMetis[12] が開発された．しかしながら，節点を縮約する場合，その縮約グラフを得ること，復元を行うことは計算時間のかかる複雑な手続きを伴う．また，節点次数が増加するため，密結合なハイパーグラフに対しては計算時間が増大する手法での縮約の概念の導入は困難である．FM法に対してはKarypisらの縮約方法が有効であったが，他の手法に対しても有効であるか，より大規模なハイパーグラフが入力となった場合にも実用的な計算時間を持つかは不明である．

本稿では大規模なハイパーグラフの分割問題を節点を縮約しないで直接扱う組み立て法を提案する．提案手法では著者らが先に提案している組み立て法 [9] に節点の探索順序を決定する高速な機構を組み込む．また，提案した組み立て手法を用いて遺伝的アルゴリズムに基づくヒューリスティック手法を考察する．これら2つの提案手法に対して，MCNC (Microelectronics Center of North Carolina) のベンチマークとISPD98 (International Symposium on Physical Design 98) のベンチマークを入力としたシミュレーション実験を行った結果を示す．

2 準備

2.1 ネットワークのモデル化

論理回路は論理セルとそれらを接続するネットから成り立っている．各セルを節点に，ネットをハイパー枝で対応させることにより，論理回路はハイパーグラ

フにモデル化できる．また，節点の重みをセルのサイズとして表現できる．以下にハイパーグラフに関する定義を与える．

定義 1 ハイパーグラフ $H = (V, E)$ は，節点集合 $V = \{ v_1, v_2, \ldots, v_m \}$ とハイパー枝の集合 $E = \{ e_1, e_2, \ldots, e_n \}$ から構成される．ここで，ハイパー枝 e_i $(1 \leq i \leq n)$ は空でない節点集合 V の部分集合で表わされる．すなわち $e_i \subseteq V$ である．全てのハイパー枝 $e \in E$ に対し，$|e| = 2$ であるとき H をグラフともいう．H の各節点 $v \in V$ に対し，重みを表わす関数 $size(v)$ が定義されているなら，H を重み付きハイパーグラフと呼ぶ． □

定義 2 ハイパー枝集合 $C \subset E$ において，ハイパーグラフ H から C を取り除くことにより V を互いに素な空でない 2 つ以上の節点集合に分割するとき，C を H のカットと呼ぶ．また，$|C|$ をカット数とする． □

2.2 ハイパーグラフ分割問題

本研究で扱うハイパーグラフ 2 分割問題を以下に定式化する．

[ハイパーグラフ 2 分割問題]

入力：重み付きハイパーグラフ $H = (V, E)$, 自然数 k

出力：制約条件を満たし，カット数を最小とする H の 2 分割 (V_1, V_2)

制約条件：$\{|\sum_{u \in V_1} size(u) - \sum_{v \in V_2} size(v)|\} / W \leq max\{max\{size(w) | w \in V\}/W, 0.1\}$

ただし，$W = \sum_{u \in V} size(u)$ とする． □

ハイパーグラフ 2 分割問題は NP 困難 [7] であることが知られている．$VLSI$ の設計においては短い期間で開発を行うことが要求されるので，この問題を解くには高速なヒューリスティックアルゴリズムの開発が必要である．

3 高速組み立て法

本節では，著者らが先に提案した高速ヒューリスティック手法 [9] を改良した幅優先探索に基づくハイパーグラフ 2 分割手法 WHB を提案する．提案手法では高速化を目的に大規模な入力ハイパーグラフに対して縮約などのグラフの変形操作を行わない．また，グラフの疎結合な性質などをそのまま保つ．

提案手法の概要を以下に示す．提案手法ではまずハイパー枝集合 E の 2 分割を求める．ランダムに 2 つのハイパー枝を選び，それらの 2 つの種ハイパー枝 seed1, seed2 から幅優先探索に基づく操作で各ハイパー枝をバランスが取れるよう取り込む．2 つの種の探索が同じ節点でぶつかったら，ぶつかった節点からの探索は止まる．全てのハイパー枝が 2 つのどちらかの領域に取り込まれたらハイパーグラフの探索を終了する．この段階で得られるハイパー枝の分割 (E_1, E_2) を概略 2 分割と呼ぶ．E_1 または E_2 に属しているハイパー枝のみに隣接している節点は分割の seed1 または seed2 側のどちらかに属するかが決定される．

次に 2 つの領域の境界部分の節点をどちらに取り込むかを決定する．境界領域のネット間の関係を二部グラフで表現して，その独立節点集合を求めることで節点の分割を詳細に決定する [9]．以上の手続きにより最終的な節点の 2 分割を得る．

概略 2 分割の質によって，出力される 2 分割の質は大きく左右される．最終的な 2 分割を求める段階では概略 2 分割の境界となるハイパー枝集合からカットとなるハイパー枝集合を選ぶ．従って，あまりにも悪い概略 2 分割から良質な最終的な 2 分割を求めることは望めない．本研究では大規模なサイズの入力に対して，高速に良質な概略 2 分割を求めるための幅優先探索のいくつかの変形を考察した．各変更は計算時間にほとんど影響を与えることなしに実現できる．

3.1 Hash に基づく幅優先探索の変形

一般に幅優先探索は FIFO (First In First Out) に従って探索する．しかしながら，大規模なサイズの入力に対して FIFO 順序の探索を進めたなら，幅がどんどん広くなりハイパー枝数に比例した数のハイパー枝数を境界に持つような概略 2 分割を得てしまう．一般に，ハイパー枝数に比例しない小さな 2 分割を持つような入力に対しても FIFO 順序では悪い概

略2分割を求めてしまい，良質な解を見つけることが困難となる．そこで，FIFO順序を何らかのpriorityによって順序づけられたHash順に幅優先探索を変形することについて考察した．Hashの1つの項目からの探索はFIFO順序に探索するものとした．以下では，各ネット節点のpriorityとして設定する値について考察する．

(a) 乱数によるpriorityの設定

FIFO順序を崩すために乱数によりそのpriorityを決定する．Hashのバケットの容量が大きいほどランダム探索となるが，一度探索されたハイパー枝からしか探索が進まないのでseed1側とseed2側の探索が入り交じったりすることはない．緩やかな探索幅の増加を行いたいという動機から導かれた方法である．

(b) ハイパー枝の端子数によるpriorityの設定

各ハイパー枝につながっている端子数によってpriorityを決定する．大規模な論理回路には端子数の大きいハイパー枝が数多く存在する．このようなハイパー枝は最終的にはカットされ易いが，概略2分割段階で多くの端子を持つハイパー枝をカットしないでFIFO順で先に探索されてしまうと探索の幅が急激に増加する傾向にある．その傾向がそのまま概略2分割に受け継がれて，悪い結果を導く．そこで，端子数の少ないハイパー枝から探索し易くすることで，幅を増加させないままで探索を進めていくことを目的とする．端子数の小さいハイパー枝に高いpriorityを設定して，端子数の多いハイパー枝は少ないハイパー枝が全て探索された後で，つまり，探索が十分に進んだ後で探索されるようにする．

(c) 既に取り込まれているハイパー枝と共有する節点の数によるpriorityの設定

既に取り込まれているハイパー枝と共通した節点に隣接する回数が多いものほど，探索において高いpriorityを設定する．共通する節点数が多いほど，それまでに取り込まれているハイパー枝と連結度が高いと見なす．適切な探索順序により，次に取り込まれる節点が少なくなるため急激な探索の幅の増加を押さえることを目的とする．

3.2 実験結果

3.1で述べた3つの幅優先探索の変形を実験的に考察するため，SPARCserver 1000E (60MHz,128MBメモリ) 上でシミュレーション実験を行った．入力データとして表1に示すMCNC (Microelectronics Center of North Carolina) のベンチマークデータとC. Alpert (http://vlsicad.cs.ucla.edu/~cheese/benchmarks.html) によって与えられている18個のISPD98のベンチマークデータを用いた．

アルゴリズムWHBでは2つの種節点をランダムに選んだ．アルゴリズムWHBの4つの異なるハイパー枝探索方法に対して10個の異なるランダムな入力に対して適用し，その最小カット数，平均カット数，1回の試行の平均のCPU時間を表2に示す．ここで，3.1の(a), (b), (c)のそれぞれの探索方法に対する実験結果をrand, net, multipleの行で示す．

表2により多くのデータで幅優先探索の変形により解の質が異なることが分かる．特に，golem3ではハイパー枝の端子数により優先順序を決定することによってFIFOの解を平均83%改良している．また，FIFO以外の探索方法のうちのどれかがFIFOの解よりも小さいカットサイズの平均解を出力している．golem3以外のデータに対してはmultipleの探索方法がほとんどのデータに対して他の手法より良質な解を出力している．計算時間に関してはほぼ節点のサイズの比例した小さな計算時間を持っていることが分かり，より大規模な入力ハイパーグラフに対しても実用的な計算時間を持つことが予測される．

4 遺伝的ハイパーグラフ分割アルゴリズム

3節で提案した高速ハイパーグラフ分割アルゴリズムWHBで得られた解をさらに改良することを目的として，本節では遺伝的アルゴリズムの概念とアルゴリズムWHBを組み合わせたヒューリスティック手法

表 1: ベンチマークデータ

Data	No. of vertices	No. of hyperedges	max size	min size	ave. size
primarySC1	752	1159	180000	4500	31941
primarySC2	2907	3971	180000	4500	17736
industry2	12142	13419	29120	3328	8569
industry3	15059	21966	59392	14336	23967
golem3	100312	144949	8640	90	309
ibm01	12752	14111	269568	0	331
ibm02	19601	19584	960960	0	431
ibm03	23136	27401	1058624	0	425
ibm04	27507	31970	851329	0	337
ibm05	29347	28446	320	0	152
ibm06	32498	34826	1163032	0	263
ibm07	45926	48117	562560	0	257
ibm08	51309	50513	1627648	0	262
ibm09	53395	60902	950752	0	328
ibm10	69429	75196	2281344	0	684
ibm11	70558	81454	950752	0	300
ibm12	71076	77240	2376192	0	520
ibm13	84199	99666	1058624	0	297
ibm14	147605	152772	571520	0	194
ibm15	161570	186608	4017600	0	226
ibm16	183484	190048	995328	0	286
ibm17	185495	189581	397824	0	227
ibm18	210613	201920	324864	0	159

表 2: 探索方法の異なるアルゴリズム WHB の実験結果

	FIFO			rand			net			multiple		
Data	Cut size		CPU	Cut size		CPU	Cut size		CPU	Cut size		CPU
	Min.	Ave.	time[s]	Min.	Ave.	time[s]	Min.	Ave.	time[s]	Min.	Ave.	time[s]
golem3	10877	11352	5.8	8979	9463	6.9	1484	1734	3.6	10499	10948	5.5
ibm01	664	764	1.2	340	595	0.9	608	735	0.8	636	742	0.7
ibm02	505	1258	2.5	542	1075	2.1	779	1933	2.3	896	1345	2.1
ibm03	2417	3205	4.8	2552	2773	4.2	2747	3188	3.7	2113	2678	4.1
ibm04	1343	1856	4.3	1132	1817	3.4	1378	1494	3.8	1029	1224	4.1
ibm05	3272	4575	5.0	3933	5140	5.1	4061	5026	4.4	2882	3909	4.7
ibm06	2348	3100	4.6	2834	4210	5.7	2899	3706	5.3	2012	2474	4.4
ibm07	2273	3783	5.7	2219	4439	5.8	1845	2225	4.4	1551	2643	5.3
ibm08	4112	5350	18	5021	5370	13	4094	4809	21	4290	5477	14
ibm09	1894	4718	7.5	1709	3201	6.6	2485	4083	6.5	1428	2129	7.3
ibm10	4879	5392	9.7	3358	4247	8.8	3359	5003	9.0	3066	3891	8.3
ibm11	3771	4552	8.0	3993	4717	8.4	3344	4366	7.3	2099	2388	8.5
ibm12	5453	8361	15	5430	6980	14	5512	6717	14	4444	5502	15
ibm13	5068	7314	14	3233	4668	21	5432	7132	23	2110	2527	26
ibm14	9730	13222	18	6833	7981	15	7225	10453	18	4245	8602	27
ibm15	9645	14750	29	8387	13330	27	6928	9154	22	5891	7161	27
ibm16	8312	12858	24	4905	8841	20	4721	7454	18	3570	6826	20
ibm17	9354	14732	26	6843	10060	21	7318	8495	17	4949	8863	27
ibm18	12932	17693	28	4640	8686	18	8015	11075	17	4326	5942	20

GWHB を考察する．アルゴリズム GWHB の詳細は文献 [9] に記述している．

4.1 遺伝的アルゴリズムの概要

一般的に遺伝的アルゴリズムは遺伝子の進化の概念を組み合わせ最適化問題に応用した確率的アルゴリズムとして知られている [8]．遺伝的アルゴリズムの特徴はコード化された複数の解を保持し，前世代の2個以上の解のコードを交配することで新しい解を生成することである．これは遺伝子の交配プロセスの概念に基づいている．様々な最適化問題を扱うため，遺伝的アルゴリズムでは解のコード化を行い，各個別問題固有の情報とは独立に2個以上の解をランダムに交配する．しかしながら，コード化を用いることによりサイズの大きな問題に対しては解の収束に長大な時間が必要であるなどの問題点があった．これに対し，アルゴリズム的な交配手法を元の遺伝的アルゴリズムに導入する手法のものでは，コード化による遺伝的アルゴリズムと比較すると高速に解が収束し，また一般のヒューリスティック手法による解より良質な解を得られることが示されている．ハイパーグラフ分割問題に対しては Alpert ら [2] らによって遺伝的アルゴリズムの概念と Multilevel partitioning technique を組み合わせた手法が提案されている．この手法では高速に従来手法と同程度な解を求めることが示されている．

本稿では，著者らが先に提案したハイパーグラフ2分割問題に対する遺伝的アルゴリズム [9] をさらに大規模な入力を対象とするため拡張した．文献 [9] と同様，本手法は主に初期化，交配，淘汰，反復の4ステップにより構成されている．初期化ステップでは $2p$ 個の異なる解をアルゴリズム WHB を適用して求める．求めた解集合の中からカット数の小さい p 個の解を選び，初期解集合を構成する．この初期化ステップはアルゴリズム WHB が出力する解の質に大きなばらつきがある場合，計算時間および解の質を改良する．交配プロセルでは前世代の2個の解を組み合わせ，2個の解の共通性質を受け継いだ新しい解をアルゴリズム WHB を適用して求める．淘汰のステップでは交配プロセスで求めた p 個以上の解から目的関数の値の良い解を $(p-m)$ 個，残りの解より m 個ランダムに選び，次世代を構成する．ここでは m 個の良質でない解を含めることにより，一般の遺伝的アルゴリズムにおける突然変異と同様に，局所解に陥ることを回避するようにしている．

ハイパーグラフ2分割問題に対する遺伝的アルゴリズム GWHB を以下に示す．

[アルゴリズム GWHB]

1. パラメータ p, R, m を入力する．
2. アルゴリズム WHB を適用し，$2p$ 個の初期解を求め，そのうちカット数の小さい p 個を選び解集合 Pを構成する．
3. 解集合 Pのランダムな $1.5p$ 個の異なる非順序対に対しアルゴリズム WHB を適用し解を交配する．
4. 3. で求められた $1.5p$ 個の解より $(p-m)$ 個までの目的関数の良い解と残った解からランダムに m 個選ぶ．選ばれた p 個の解により集合 Pを更新し，次世代を構成する．
5. 3. と 4. を R回繰り返し，得られた最良解を出力し終了．

4.2 解の交配

アルゴリズム GWHB における解の交配方法について説明する．まず，2つの解 $S_x = (X, \overline{X})$，$S_y = (Y, \overline{Y})$ の交配を考える．$X \cap Y, X \cap \overline{Y}, \overline{X} \cap Y, \overline{X} \cap \overline{Y}$ で示される4個の節点部分集合を求める．次に，最も大きな2個の部分集合を求め，これらを S_1, S_2で表わす．この2つの節点集合を縮約し，2つの seed としてアルゴリズム WHB を適用し新しい許容解を求める．この交配プロセスから双方の親の性質を受け継いだ解を求めようとする．交配プロセスで用いられるアルゴリズム WHB での節点の探索方法てしてはネットの端子数による priority の設定方法を用いる．

5 実験結果

アルゴリズム GWHB を Sun Microsystems 社の SPARCserver1000E (60MHz) 上で C 言語を用いて

実現し，先に述べたベンチマークデータに対してシミュレーション実験を行った．

アルゴリズム GWHB の実験結果は人口数 9, 繰り返し回数 10, 悪い解の残存数 2 とした場合の実験結果を示している．golem3 に対してはアルゴリズム WHB, GWHB ともにハイパー枝の探索方法として net を適用した．golem3 以外のデータに対してはアルゴリズム WHB, GWHB ともに探索方法として multiple を適用した．アルゴリズム WHB,GWHB の実験結果は 10 回の試行の中で最も小さいカットサイズと平均計算時間を表している．hMetis[11] に関する実験結果は hMetis バイナリ [12] を SPARCsever1000E 上で適用して求めた．hMetis では 20 個の異なるランダムな初期解から改良し，その中の最良解だけを V-cycle で改良するオプションを選択した (Nruns=20, UBfactor=5, Rtype=1, Vcycle=1, Reconst=0)．上記のオプションの hMetis を 10 回試行し，得られた最良解と 1 回の試行の平均計算時間を表 3に示す．FM 法 [6], CLIP[3] の従来手法に関する実験結果は文献 [4] から引用した値である．FM 法，CLIP に関するカット数はランダムな初期解から始めた 100 回の試行の中の最良解を示し，計算時間は 1 回の試行の平均計算時間である．ここで，FM, CLIP に関する計算時間は IBM RS6000 s/595 (135MHz) によるものである．IBM RS6000 は hMetis の計算時間の比較より，SPARCserver 1000E の約 4 倍の計算速度を持つと予測される．

表 3よりアルゴリズム GWHB は hMetis よりも約 2 倍以上高速な計算時間で，解を求めていることが分かる．ibm03, ibm06, ibm08, ibm12 に対して hMetis よりも GWHB はかなり悪い解を求めていることが分かる．今後，これらのデータに対する新たな探索方法の考察が必要である．他のデータに対しては GWHB は hMetis とほぼ同程度の解を求めている．FM 法, CLIP に対しては多くのデータに対して，解の質が改良できていることが分かる．また，大規模なデータに対する計算時間はほぼ同程度であることが分かる．

表 3から，ほとんどのデータに対してアルゴリズム WHB により求まった解が良質な場合，アルゴリズム GWHB でも良質な解が求まっている．特に golem3 では，アルゴリズム WHB で他の従来手法と変わらない解が得られている．しかし，先に挙げたアルゴリズム GWHB で良質な解を出力できなかったデータに対しては，アルゴリズム WHB の解が良質の場合でも GWHB での解の改良の度合いが小さかった．今後，アルゴリズム GWHB で有効な探索方法の考察，もしくはアルゴリズム GWHB の手続きの変更が必要である．

6 あとがき

本稿では節点の縮約によらない組み立て法に基づく高速ハイパーグラフ 2 分割手法を提案した．また，提案手法を遺伝的アルゴリズムの概念と組み合わせたヒューリスティック手法について考察した．提案手法を実現し，大規模なベンチマークに対するシミュレーション実験を行った．実験の結果，提案した組み立て法は大規模なハイパーグラフに対しても数十秒程度と高速であり，遺伝的アルゴリズムの概念を組み合わせることにより従来手法と同程度の解を実用的な計算時間で求めることが分かった．また，計算時間の増加率もほぼ節点のサイズの増加率と同じであることが分かり，さらに大規模なハイパーグラフに対しても実用的な計算時間を持つことが予測される．今後の課題としては組み立て法におけるさらに良質な解を得ることができる節点の探索順序の決定方法の開発が挙げられる．

謝辞

実験を行うにあたりご協力頂いた広島市立大学学部生 田中 浩一君に感謝する．なお，本研究の成果の一部は平成 10 年度文部省科学研究費補助金奨励研究 (A)10780201 によるものである．

参考文献

[1] C. Alpert *and* A. Kahng: "A general framework for vertex orderings, with applications to netlist clustering," Proc. 1994 IEEE/ACM International Conf. on Computer Aided Design, pp.63-67 (1994).

[2] C. Alpert, L. Hagen *and* A. Kahng: "A hybrid multilevel/genetic approach for circuit partitioning," Technical report, Computer Science Department, UCLA, Los Angeles, CA, pp. 1-6 (1996).

表 3: 提案アルゴリズムと従来手法との比較

Data	Cut size					CPU Time (s)				
	hMetis	FM	CLIP	WHB	GWHB	hMetis	FM	CLIP	WHB	GWHB
primarySC1	49			74	50	6			0.06	4
primarySC2	145			263	145	10			0.6	14
industry2	168			374	221	67			1.3	50
industry3	241			467	224	97			1.2	62
golem3	1424			1484	1345	1603			3.6	335
ibm01	215	270	246	636	238	74	4.5	5.5	0.7	73
ibm02	264	313	439	896	262	303	3.9	7.3	2.1	123
ibm03	673	1624	1915	2113	1139	403	0.3	27	4.1	277
ibm04	441	554	488	1029	396	491	14	11	4.1	190
ibm05	1713	1874	2146	2882	1786	658	24	27	4.6	198
ibm06	367	1479	1303	2012	833	565	36	76	4.4	263
ibm07	741	870	748	1551	812	1133	24	35	5.3	248
ibm08	1153	1411	2176	4290	1497	1256	28	84	14	635
ibm09	521	750	527	1428	563	1098	32	31	7.3	408
ibm10	756	982	971	3066	1098	1777	37	58	8.3	700
ibm11	692	1319	977	2099	967	1600	49	56	8.5	529
ibm12	1975	2306	2717	4444	2571	2554	49	36	15	1024
ibm13	837	1196	1023	2110	1031	2072	48	69	26	817
ibm14	1525	3015	2426	4245	2381	5440	143	157	27	1032
ibm15	1789	7197	5292	5891	2403	6404	25	794	27	1156
ibm16	1680	2173	2314	3570	1900	7657	13	24	20	975
ibm17	2247	2818	3634	4949	2434	10955	185	227	27	1024
ibm18	1522	1664	3043	4326	1634	10023	217	363	20	1042

[3] C. Alpert, J. Huang *and* A. Kahng: "Multilevel circuit partitioning," Proc. 34th IEEE Design Automation Conf., pp. 530-533 (1997).

[4] C. Alpert:"The ISPD98 circuit benchmark suite," Proc. ISPD98, pp. 80–85 (1998).

[5] C. Dutt *and* W. Deng: "VLSI circuit partitioning by cluster-removal using iterative improvement techniques," Proc. IEEE/ACM International Conf. on Computer Aided Design, pp. 194-200 (1996).

[6] C. M. Fiduccia *and* R. M. Mattheyses: "A linear time heuristic for improving network partitions," Proc. 19th IEEE Design Automation Conf., pp.175-181 (1982).

[7] M. R. Garey *and* D. S. Johnson: *Computers and Intractability, A Guide to the Theory of NP-Completeness*, W. H. Freeman and Company, New York (1979).

[8] D. E. Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley (1989).

[9] Y. Kamidoi, S. Wakabayashi *and* N. Yoshida: "An efficient GA hybrid for hypergraph bisection with application to VLSI placement," Proc. IEEE Asia-Pacific Conf. on Circuits and Systems, pp. 396-401 (1992).

[10] G. Karypis, R. Aggarwal, V. Kumar *and* S. Shekhar: "Multilevel hypergraph partitioning: applications in VLSI domain," Proc. 34th IEEE Design Automation Conf., pp. 526-529 (1997).

[11] G. Karypis, R. Aggarwal, V. Kumar *and* S. Shekhar: "Multilevel hypergraph partitioning: applications in VLSI domain," Technical report, Computer Science, University of Minnesota, Minneapolis, pp.1-22 (April, 1997).

[12] G. Karypis *and* V. Kumar : "hMETIS: a hypergraph partitioning package Version 1.5," Available on the WWW at *http : //www.cs.umn.edu/ ~karypis* (1998).

[13] B. W. Kernighan *and* S. Lin: "An efficient heuristic procedure for partitioning graphs," The Bell System Technical Journal, 49(2), pp. 294–307 (1970).

[14] B. Krishnamurthy: "An improved min-cut algorithm for partitioning VLSI networks," IEEE Trans. on Computers, C-33 pp. 438–446 (1984).

[15] B. M. Riess, K. Doll *and* F. M. Johannes: "Partitioning very larger circuits using analytical placement techniques," Proc. ACM/IEEE Design Automation Conf., pp. 646-651 (1994).