様式 C-19

科学研究費補助金研究成果報告書

平成２１年　６月１２日現在

研究種目：基盤研究（C）
研究期間：2006　～　2008
課題番号：１８５００１２２
研究課題名（和文）　VLSI フロアプランのためのゲーム論的配置手法の開発

研究課題名（英文）　Game-theoretic layout algorithm for the VLSI floor plan design

研究代表者
徳増　眞司　（TOKUMASU SHINJI）
神奈川工科大学・情報学部・教授
研究者番号：００２７８０２９

研究成果の概要：

板金の板取やVLSIフロアプランなどの工学的な配置問題を，ポリオミノ箱詰め問題としてモデル化し，ゲーム論的なアプローチに基づいて最適化する新しいアルゴリズムの構築を行った．提案法は，ポリオミノの位相的特徴量を評価尺度として配置専有面積を最小化する点に特徴がある．更に，ピース間の配置制約を考慮した，より現実的なモデルへの拡張を実現し，結果として，本来のアルゴリズムを大幅に変更することなく，また，効率を損なうことなく有効な結果が得られることを示した．

交付額

（金額単位：円）

	直接経費	間接経費	合　計
２００６年度	1,000,000	0	1,000,000
２００７年度	500,000	150,000	650,000
２００８年度	500,000	150,000	650,000
年度			
年度			
総　計	2,000,000	300,000	2,300,000

研究分野：総合領域
科研費の分科・細目：情報学　知能情報学
キーワード：
VLSI，フロアプラン，ポリオミノ，箱詰め問題，ジグソーパズル，最適配置アルゴリズム，ゲーム理論

１．研究開始当初の背景

VLSI 設計における素子（モジュール）配置に関わるフロアプランの問題は，オペレーションズリサーチの観点から見ると，最適化に関わる一種の配置問題であり，俗に「板取問題」と呼ばれる．この種の配置問題に関しては，その解法の提案や応用に関する取組みが数多くなされているが，決め手となる万能薬的な処方箋があるわけではなく，個別の問題に対して解決が図られるのが一般的である．

集積回路の開発においては，その高密度化を実現するために長方形やＬ字状の角形素子群を如何に小さいスペースに配置するか，という問題が扱われる．このタイプの問題については，これまでも種々の解法が提案されているが，配置順序対表現(sequence pair)または領域のブロックスライシングに関わるポーランド法の表現を用いて，素子群の位置関係を表現し，アニーリング手法を基に，その位置関係と各素子位置にゆらぎを与え，最適化を図る手法などが典型的である．さらに，

集積回路設計における素子配置の問題は，これと対になる素子間配線の問題と強くリンクしており，上記の手法においても，アニーリングの際の評価指標として，配線上の制約も考慮するのが一般的である．

２．研究の目的

本研究では，集積回路設計における素子配置と類似した数学パズルの問題として，ポリオミノ箱詰め問題に着目する．すなわち，集積回路設計における素子配置問題を直接対象とする替わりに，これを理想化した典型的な数理モデルとしてポリオミノ箱詰め問題を取り上げ，理論的な考察を試みた．

研究代表者らは，このような見地からポリオミノとその箱詰め問題に対して考察を加えた結果，ポリオミノの位相的特徴量を尺度として将棋やチェスにおけるチェックメイト手順を想定した，ゲーム論的手法を用いた解法が有効であることを発見した．本研究の目的は，研究代表者らが提案するポリオミノ箱詰めアルゴリズムの有効性を検証するとともに，これを実際の集積回路設計における素子配置問題に適用する方法を提示することである．さらに，ピース間配置制約をピース相互の位置関係に対するペナルティコストとして取り扱い，先のアルゴリズムにコスト評価に基づくピース配置順序づけのステップを加えて拡張することによって，この制約を考慮したポリオミノ箱詰め問題の解法が容易に得られることを示すことである．

３．研究の方法

本研究課題では，まず (1) 標準的な平面ポリオミノ箱詰め問題(PPPP)の解法を検討し，その上で (2) これを現実の素子配置問題に適用するための修正法(APLP)を提示し，その検討を行った．さらに，(3) ピース間配置を考慮した拡張アルゴリズム(ePPPP)について検討を行った．以下に，各ステップで提案するアルゴリズムの概要について述べる：

(1) PPPP の解法

① 問題の定式化

PPPP (Planar Polyomino Packing Problem) とは，「あらかじめ，標準的なピースである 1 個以上の D 型ポリオミノと，これとは別に，領土と呼ばれる 1 個の D 型領域が与えられ，解が存在するという前提の下でこの領域内部を与えられたピースで埋めつくす問題」である．ここで，D 型ポリオミノとは，「単体ポリオミノと呼ぶ，単位寸法の正方形を 1 個以上単体複体的に接合して得られる角形形状のポリオミノ」を指す．D 型とはディジタル型の意味である．なお，ゲーム論的展開を図るために，与えられたピースと領土を総称して盤面と呼ぶことにする（図 1）．

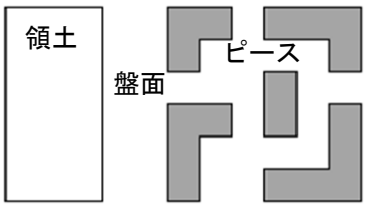
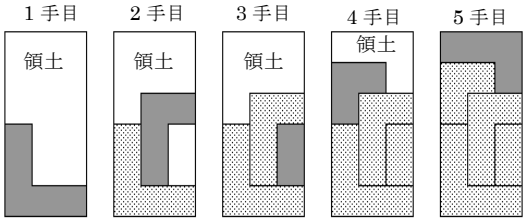


図 1　盤面の定義



Puzzle-A

図 2　配置手順の例

本研究では，PPPP の解とは“与えられた角形ポリオミノのピースを一定のルールの基で領土上に順次配置するピース選択の手順である”と解釈して，ゲーム論的なアプローチに基づく解法の提案を行った．ここで，一定のルールとは，提示された領土の左下隅で且つ既に置かれた駒に単体複体的に接する位置に次のピースを配置する（この際，置かれたピースは領土からはみ出ないものとする）ことである．提案法に基づくピースの配置手順の例を図 2 に示す．ここで，各段階の手順は，一定の配置ルールの下で選択されるので，問題の解は，各段階で選択された一連のピース選択で得られる路（パス）として解釈することができる．

② 解法の原理

ポリオミノ箱詰め問題における個々の候補手順が，解のパスに載っているかどうかの判定は，その手順を試行した結果として得られる盤面が再びポリオミノ箱詰め問題になるかどうかの判定に帰着されるので，ポリオミノ箱詰め問題の解法は，再帰的に構成されるのが自然である．各ステージでは，全ての候補手順は，以降の手順を考慮して，あらかじめその妥当性をゲーム論的に評価し，さらに，有望な次の候補手順の選択へとすすむ必要がある．その際，中途ステージで解のパスからはずれた場合，一旦前のステージに戻って，再びパスに乗るように別の候補を試行する事になる（これはまさに，ゲーム論的評価に基づく縦型探索の原理である）．

以上の方針を実現するため，提案法では，2 種類のゲーム論的評価（尺度）を導入した．

ひとつは各ステージにおける候補手順の妥当性を評価する「盤面評価尺度」であり，もう一つは盤面評価に合格した妥当な候補手順に対して攻め手としての有効度を評価する「手順評価尺度」である．ここで，盤面評価は“領土と合同な持ち駒の単体複体多角形を構成できるかどうか”によって判定される．また，手順評価は，このような必要条件の下で，領土に一致する，最終的な合成手順の組み合わせの複雑度に関する情報量（エントロピー）として評価される．

結局，本ゲーム論的解法では，これら二つの評価尺度を用いたつぎのような 2 段階の評価を連動させる．

第 1 段評価:

盤面評価尺度を D 型ポリオミノの位相的特徴量に基づく不等式系で表し，正解パスの存在の可能性を評価する．この不等式系を満足することが，正解パスに載っているための必要条件である．

第 2 段評価:

第 1 段評価で合格となった候補手順に対して，手順評価尺度である，エントロピーの値をもって正解パスへの優先順位を評価する．

この 2 段階評価を伴う縦型探索は，最初の一つの解を高速に求めるために有効である．なぜなら，不能なピース配置の選択は，第 1 段評価で木の展開の早い段階で刈り取られ，第 2 段評価は，解の存在の近道となる枝を先に展開するように機能するからである．

③　アルゴリズム

提案する PPPP の解法では，探索木を利用して解のパスを生成・展開する．各ノードには，属性として，盤面の深さを示すステージ番号（#stage），ピース番号（#piece），当該手順のエントロピーの値（#Entropy）と下位ノードへのアドレスポインタである枝(#branches）を擁する構造体がノードの生成に伴って作られる．ここで，枝を有するノードを，fork とよび，枝を持たないノードを leaf と呼ぶ．以上の探索木の下で，PPPP アルゴリズムは次のように定式化される．また，本アルゴリズムの適用例を図 3 に示す．

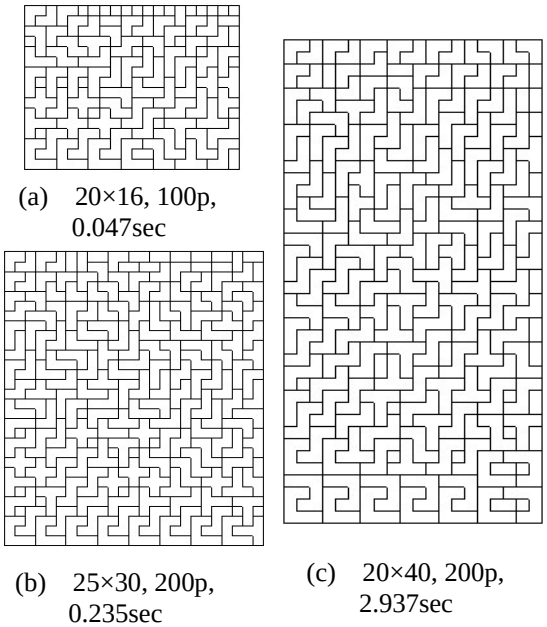
[PPPP algorithm]

Step 0: #stage=0 且つ他の属性を null とするルートノード，ROOT を生成し，盤面#0 を作成する．

Step 1: #stage が最大の leaf ノードをスキャンして，最少の#Entropy 値を有するノード:NODE を選ぶ． ROOT ノードから NODE ノードに至るパスをたどり，パス上の全てのピースを前記したルールによって順に領土に配置して，盤面#I を再生する．ここに，I=#stage of NODE とする．
if (I is n) 上記盤面が解, return;

Step 2:
for (盤面#I の持ち駒 P) {
候補ピース P を盤面#I に配置し，盤面#(I+1)を生成する．上記盤面に対し，第 1 段評価を行う．
if (第 1 段評価に合格) {
第 2 段評価を行い，盤面エントロピー ENTROPY を求める．
新たにノードを生成し，#stage=I+1, #piece=P, #Entopy=ENTROPY, #branches=null; とする．
NODE からこのノードに枝を張る．
}
}
if (第 1 段評価で全てのピースが不合格) {
NODE 自身と NODE を指す親ノードを削除する．
さらに，ROOT に向かって最上位の親が少なくとも一つの枝を残すまでこの削除操作を再帰的に行う．
}

Step 3:
Step 1 に戻る． □



(a) 20×16, 100p, 0.047sec

(b) 25×30, 200p, 0.235sec

(c) 20×40, 200p, 2.937sec

図 3　PPPP の例

(2)　PPPP の一般化

①　DPLP への展開

PPPP では，あらかじめ領土が与えられていて，この領土を与えられたピースで埋めつくす解が存在することが前提である．しかし，現実の問題においてはその前提が無く，全ピ

ースを配置するための最少の面積の領域が必要である場合が多い．そこで，これを矩形領域面積最小化問題： DPLP （D-type polyomino layout problem）としてモデル化する．これには，矩形領土は横 X を固定，縦 Y を可変として，あらかじめ面積 XY がピースの総面積 Σpj 以上となるように Y を設定し，ピース配置を可能とする最少の Y を求める問題である．したがって，問題 DPLP は領域を規定する縦 Y の値を目的関数とする問題 PPPP の変形と見做すことができる．この問題を解くためには, XY-Σpj 個の単体ポリオミノをダミーとして，ピース群に加えれば，これは，純然たる PPPP となる．もし，解が得られたら，Y=Y-1 として，同様の PPPP を解き，解が得られなくなった時点でその時の Y に 1 プラスしたものが解である．

② APLP への展開

D 型ポリオミノは寸法が規格化されたポリオミノであるが，実際の配置問題では，位相的には D 型と同相でも寸法が規格化されていないのが普通である．そこで，このタイプのポリオミノを A 型（アナログ型）ポリオミノと呼ぶこととし，このための矩形領域面積最小化問題： APLP （A-type polyomino layout problem）の検討を行った．

APLP の解法は以下に示す 2 段階の操作に基づく．まず，あらかじめ各 A 型ピースに対して（必要ならば 1 個以上の A 型ピースをひとまとめにして）これを内包する最少の D 型ポリオミノを対応させ，この D ピース群に対して DPLP を解く(Step 1)．次に，配置結果として得られた D ピースを対応する A ピースに戻す．その上で，互いの相対位置を変えずにピース間の隙間を順次詰めることによって，縦横寸法の縮減を図る．

(3) ピース間配置制約を考慮した ePPPP

最後に，標準的な箱詰め問題 PPPP にピース間制約を導入した拡張問題(ePPPP)の解法について検討した．ここで，ピース間の制約とは，VLSI フロアプランでは，モジュール間の pin-to-pin 結線に伴う最短ルーティング制約から派生する．

① 好み評価尺度の導入

ピース間配置制約をピース間の好みマトリックス M = [m(i, j)]で定義する．なお，m(i, j)は i と j が異なる場合(i≠j)はピース(i, j)間の好み値であり，i と j が等しい場合(i=j)は，ピース i と領土の境界との好み値，即ち，周辺に置くべき程度を示す値である．M の各要素 m(i, j)は，好みの度合いが大きければ，それ自身大きな値をとることにする．

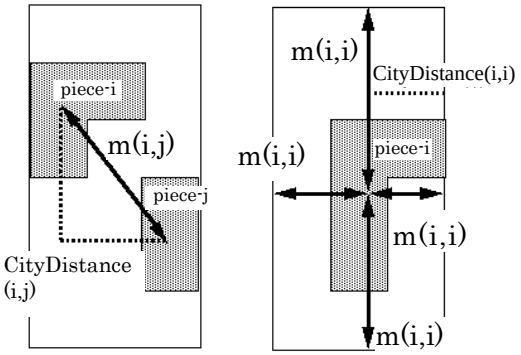


図 4　m(i, j)と CityDistance の選択

つぎに，PPPP の可能解に対して，好み評価尺度:PathPenalty を次のように定義して，ePPPP に関する目的関数とする（図 4）．

$$\text{PathPenalty} = \Sigma_{(\text{for all i,j such that } i \geqq j)}\, m(i, j) * \text{CityDistance}(i, j)$$

上記の定義によればピース間配置制約に関して良い配置になっていれば，Path-Penalty の値は小さくなる．したがって，ePPPP における解は最小の PathPenalty を有する PPPP の解である．

② 目的関数 PathPenalty の推定手順

前節で述べたように ePPPP の解を求めるためには，最小の PathPenalty を有する PPPP 解を求めればよい. 基本的には, PPPP 自身は目的関数のない，最適化問題とみなされるので，PPPP では最初の可能解は即最適解となり，最初の解を見つけた時点で終了できるのに対して，ePPPP においては，PPPP の可能解のうち， PathPenalty を最小にする解を最適解とするので，候補パスの全体にわたる大域的な探索が必要となる．以下では，効率的に PathPenalty を予測評価する手順を構成する．予測評価とは，PPPP の解が求まる前に, 即ち, 途中の候補パスの生成中に, その先に期待される解パスにおける PathPenalty の下限を予測推定する手順である．これは，PPPP アルゴリズムにおける第 1 段および第 2 段評価と連動させることによって効率的に機能するべく構成される．以下に，PathPenalty の近似的な値を求める，推定手順を示す.

[Estimation algorithm of PathPenalty]

今，ゲームのステージ#k にあって，盤面#k が生成されており，候補パスの先頭部分の k 個のピースが配置されているものとする．ただし，この部分候補パスが可能解となる保証はこの時点では存在しない.

Step 0:

LowerBound0 = $\Sigma_{(\text{for all } i,j \leqq k \text{ such that } i \geqq j)}$ m(i,j) * CityDistance(i, j)
if (k == n)
LowerBound = LowerBound0; return;

Step 1:
仮に残りの n-k 個のピースを盤面の領土に重なりを無視してランダムにおいた状態を想定する.
LowerBound1 = $\Sigma_{(\text{for } k<i \leqq n \,\&\&\, 1 \leqq j \leqq k)}$m(i,j)*CityDistance(i, j)
LowerBound2 = $\Sigma_{(\text{for } k<i \leqq n \,\&\&\, i \leqq j \leqq n)}$m(i,j)*CityDistance(i, j)
LowerBound = LowerBound0 +LowerBound1+LowerBound2;
if (k= =n-1) return;

Step 2: // pairwise substitution
for (k<i≦n&&i<j≦n) {
ピース i とピース j の位置を交換する.
Delta1=Increment of LowerBound1;
Delta2=Increment of LowerBound2;
Delta=Delta1+Delta2;
if(Delta<0)
LowerBound=LowerBound+Delta;
Else,
ピース i とピース j の位置を元に戻す
}
return; □

③ アルゴリズム

目的関数 PathPenalty 下限の推定法を用いてePPPPアルゴリズムへの拡張を行った. なお, 本アルゴリズムの考え方の骨子は, 分枝限定法による縦型探索である.

[ePPPP algorithm]

Step 0: #stage=0 且つ他の属性は null を有するノード, ROOT を生成し, 盤面#0 を作成する.
UpperBound=Infinity;

Step 1: #stage が最大の leaf ノードをスキャンして, 最少の#Entropy 値を有するノード:NODE を選ぶ.
if (NODE is null) {
最適なパスが求まった. return;
// パスはひとつ前の段階で保存される.
}
ROOT ノードから NODE ノードに至るパスをたどり, パス上の全てのピースを前記したルールによって順に領土に配置して, 盤面#I を再生する. ここに, I=#stage of NODE とする.
if (I is n) {
ROOT から NODE にいたるパスは可能解であるから, このパスを PathPenalty の値とともに保存する.
UpperBound=LowerBound;
//LowerBound は前の段階でセット済み.
このNODEとここをさす枝を削除する.;
}

Step 2:
for (盤面#I の持ち駒 P) {
候補ピース P を盤面#I に配置し, 盤面#(I+1)を生成する.
上記盤面に対し, 第 1 段評価を行う.
if (第 1 段評価に合格) {
この盤面で LowerBound を推定する.
if (LowerBound<UpperBound) {
第 2 段評価により盤面エントロピー ENTROPY を求める.
新たにノードを生成し, #stage = I+1, #piece = P, #Entopy = ENTROPY, #branches = null; とする.
NODE からこのノードに枝を張る.
}
}
}
if (新たなノードが生成されなかった) {
NODE 自身と NODE を指す親ノードを削除する.;
さらに, ROOT に向かって最上位の親が少なくとも一つの枝を残すまでこの削除操作を再帰的に行う.
}

Step 3:
Step 1 に戻る. □

4. 研究成果

本研究で構築した ePPPP アルゴリズムを実装し, シミュレーションを通して, その妥当性を検証した. ここで妥当性とは, アルゴリズムにおける探索効率を意味するものである. アルゴリズムの探索効率に関係する実験条件を以下に示す:

A. ピースの総数と領土の大きさ:
実験では次の 3 つのケースを用いた:
・Puzzle-B: 領土 5×10, 駒 13 ピース.
・Puzzle-C: 領土 10×10, 駒 25 ピース.
・Puzzle-D: 領土 10×15, 駒 40 ピース.

B. ピース配置姿勢に関する自由度:
90 度単位の回転.

C. 攻め手評価における先読みの深さ:
深さ 3 を用いる.

D. 攻め手候補の優先順位のつけ方:
PPPP アルゴリズムの結果で推奨されたように, 小さいエントロピー値を持つ攻め手候補を高順位とする.

E. 好み関数の設定:

2種類のMマトリックスを使用した．

[Matrix-1]

m(i, j) = (n – j + i + 1) * rand(0,1);　(i<j),
= (n－i + 1) * rand(0,1);　(i=j),
= m(j, i);　(i>j),

[Matrix-2]

m(i, j) = (j－i) * rand(0,1);　(i<j),
= (i－1) * rand(0,1);　(i=j),
= m(j, i);　(i>j),

ここで，0≦i, j≦n，また rand(0,1)は(0, 1)の範囲の乱数である．Matrix-1 は，ピース ID の差の小さいピース間では，好み値が多くなり，ピース ID が小さいピースは領土境界との好み値が大きくなるように設定されている．一方，Matrix-2 は，これとは反対に，ピース ID の差の大きいピース間では，好み値が多くなり，ピース ID が大きいピースは領土境界との好み値が大きくなるように設定されている．

コンピュータプログラムはプラットホーム Core2Duo E6300（1.86Ghz）1GB の PC のもとで，複雑度の異なる3種類のパズルに対して実行された．表1および図5にシミュレーションの結果を示す．この結果から，以下の知見が得られた：

(a) 先読み，エントロピー戦略，及び目的関数 Pathpenalty の下限値推定は，可能解探索と解の改良の面で効果的に機能している．

(b) しかしながら，最終結果として得られた解は，必ずしも最適解とは言えない．これは，相反する Pathpenalty の下限値推定の精度と収束速度の間を必要に応じて制御する仕組みになっているからである．即ち，必要とする結果の精度を保って，且つ高速なアルゴリズムを構成できる．

５．主な発表論文等（研究代表者、研究分担者及び連携研究者には下線）

〔雑誌論文〕（計１件）

① Yasuyuki Murai, Hiroyuki Tsuji, Hisayuki Tatsumi and Shinji Tokumasu, “Fast Placement Algorithm for Rectilinear Jigsaw Puzzle”, Journal of Advanced Computational Intelligence and Intelligent Informatics (JACIII), vol.10, no.3, pp.323-331, 2006, 査読有り.

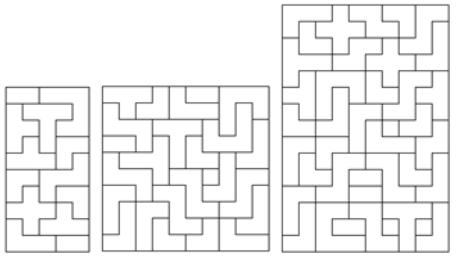
〔学会発表〕（計２件）

① Kouki Kimoto, Yasuyuki Murai, Hiroyuki Tsuji and Shinji Tokuamsu, “Rectilinear Jigsaw Puzzles: Theory and Algorithm”, IEEE World Automation Congress (WAC2006), Budapest, Hungary, July 2006.

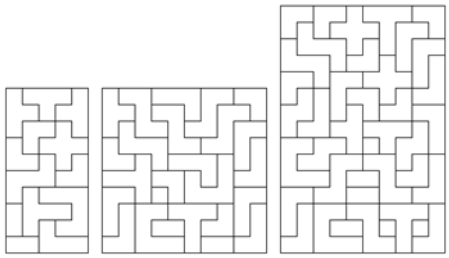
② Kouki Kimoto, Hiroyuki Tsuji, Yasuyuki Murai and Shinji Tokumasu, “A Solution of Three-dimensional Polyomino Packing Problems”, IEEE Int. Conf. on Systems, Man and Cybernetics (SMC2007), Montreal, Canada, Oct. 2007.

表１　数値実験の結果

Preference Matrix	Puzzle	PathPenalty		Used time (mm:ss.sss)
		Initial Solution	Final Solution	
Matrix-1	B	1,328	1,167	00:00.415
	C	16,415	13,176	00:47.011
	D	83,294	80,531	01:02.923
Matrix-2	B	894	714	00:01.060
	C	8,561	7,851	03:47.180
	D	45,556	42,529	00:42.805



Matrix-1



Matrix-2

図５　最終結果

６．研究組織

(1)研究代表者

徳増　眞司（TOKUMASU SHINJI）

神奈川工科大学・情報学部・教授

研究者番号：００２７８０２９

(2)研究分担者

辻　裕之（TSUJI HIROYUKI）

神奈川工科大学・情報学部・准教授

研究者番号：７０３５０６７６

村井　保之（MURAI YASUYUKI）

日本薬科大学・薬学部・講師

研究者番号：３０３７３０５４