[研究論文]

工学的配置問題のための ポリオミノ箱詰めモデルとその解法

村井保之[1]・辻裕之[2]・巽久之[3]・徳増眞司[2]

1 日本薬科大学
2 情報学部情報工学科
3 筑波技術大学

Polyomino Packing Models and Their Solutions for Engineering Layout Problems

Yasuyuki MURAI[1], Hiroyuki TSUJI[2], Hisayuki TATSUMI[3], Shinji TOKUMASU[2]

Abstract

In this paper, the authors discuss optimization algorithms for engineering layout problems in industry, such as sheet metal design, VLSI floor plan design and so forth. We propose game-theoretic algorithms by constructing polyomino packing models for those problems. First of all, we showed effective performance of topological features for minimizing the layout area. Then, introducing layout constraints among individual pieces, the optimization algorithms were extended successfully without major changes in originals. A part of these results has been presented individually in society conferences or journals including this journal. In this paper, organizing the whole results, we construct a set of polyomino packing models as a general optimization algorithm and also, show its applicability for the varieties of actual layout problems.

Keywords: engineering layout problems, polyomino packing models, VLSI floor plan, game theory

1. はじめに

物流における車両荷積み，板金製品や，モータコア部品，水車ケーシング部品の設計・製作における，母材からの平面部材の切り出し，アパレル製品の製作における，生地からのパターン裁断等の際に遭遇する問題は，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，「板取問題」と呼ばれることもある．古来より，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，ある種のアートに属する複雑度を有する問題である．ところで，この板取問題には，最適化の観点からは，3種類の切り口がある．即ち，

(1) 一定の母材からなるべく沢山の部材を切り出す問題
(2) 与えられた部材群をなるべく小さい母材から切り出す問題
(3) 前記問題に関して，部材（又は部材間）配置制約を考慮した問題

である．母材にしても定尺方形の物もあれば，ロール上のもの，不定型のものもあるが，いずれの問題も，必要となる母材のコストを最小化することが狙いであり，最適化の指標は，そのコストをとるのが基本である．この場合，均質な母材ならば，コストの代わりに，必要な母材の大きさ，ないし面積（3次元の場合は体積）を指標として用いることが出来る．しかし，たとえば，毛皮や皮革から洋服やバッグのパターンを裁断する場合とか，鮪から切り身を裁断する場合などは，相手が均質でないこと，部位毎の価値が異なることから，コストでなくプロフィットの最大化をとる場合もある．また，均質の場合であっても，加工法やその他の制約が関わる場合には，単純に必要な母材の大きさだけで考えるわけには行かない．もっとも，母材が均質で，必要な母材の大きさだけを指標として考えられる場合でも，許される部材の置き方が無数にある中で最適解を求める，NP完全の問題に属する．このような手法は一般には，ネスティングアルゴリズム（nesting algorithm）と呼ばれており，手順的には遺伝的アルゴリズム[1]や，やアニーリング手法[2]，さらには，部材間に仮想的な反発力を導入して，重なりを避けるヒューリスティック手法[3]などが知られている．なお，(1) と (2) は双対問題であり，

実際には一方だけ考えておけばよい.

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においても同様な問題がある. これらの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. 前述した (2) のタイプの問題である. この問題については, この分野の専門家等によって, 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[4]-[7]. 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である. すなわち, 実際には (3) のタイプの問題に属する.

一方, 類似の問題を数学パズルにも観ることが出来る. すなわち, ポリオミノ箱詰め問題である[8]. これは, あらかじめ与えられた前記角形素子と同等な平面ポリオミノピース群を, 一つの（これも与えられた）領域（これもポリオミノ)を完全に埋めつくすように配置する問題であり, 解が少なくとも一つ存在することが前提である. たとえば, 4 つの正方形で構成される, 5 種類のポリオミノピース 2 組を使って, 5×8 の箱を隙間無く埋めつくすピース配置を行うテトラミノパズルや, 5 つの正方形で構成される, 12 種類のポリオミノピース 1 組を使うペントミノパズルなどが知られている. これらは, 簡明且つ明確に定義された数学的対象であり, (1) 乃至 (2) を理想化したタイプの問題として考えることが出来る.

したがって, 前述した, 工学上の実際問題を直接対象とする替わりに, このようなパズルを配置問題を代表する典型的な一つのモデルとして, 理論的に考察することが問題解決のアルゴリズムを構築するために有効な知見となることが期待できる. 筆者らは, この見地にたって, 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, その有効性についても検証した[9]-[12]. しかしながら, この解法では, 純然たる配置面積最小化だけ, すなわち (1) および (2) のタイプの問題が扱われており, (3) に述べたタイプの問題の扱いが不十分なので応用範囲は限定的である. そこで次の段階として, 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, 従来のアルゴリズムに, コスト評価に基づくピース配置順序つ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した.

本論ではこれらの結果を踏まえて, 改めて, 全体を展望して体系化し, 工学的な配置問題が, ポリオミノ箱詰めモデルとして確立できること, 個別的な配置問題の解法が, 本モデルの解法の問題向きの解釈によって, 誘導できることを示すことである. 本論の構成は以下のようになる. 次の第 2 章では, 標準的な平面ポリオミノ箱詰め問題に対するモデル化とゲーム論的探索手法に関してこれまでに得られている結果を下に, その解法の骨子を述べる. 第 3 章では, 前記モデルの一般化として, 前記 (1) & (2) を考慮した配置問題向きのモデルを提案する. 第 4, 5 章では, 前記 (3) のピース間配置制約を考慮したモデル化と探索アルゴリズムの拡張を行う. 第 6 章は数値実験により妥当性を検証し, 最後に第 7 章は全体のまとめである.

2. ポリオミノ箱詰め問題におけるゲーム論的解法[9]-[12]

ここで述べるゲーム論的解法は, 角形形状を有するポリオミノ箱詰め問題に対して, 筆者らによって提案された包括的なアルゴリズムであり, テトラミノパズルやペントミノパズルのような特定の問題に対して限定的に適用するアルゴリズムではない. 以下にその方法の概要をまとめておく.

2.1 標準的なポリオミノ箱詰め問題の定式化

標準的な平面ポリオミノ箱詰め問題 (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いうことにする (Fig. 1). この問題 PPPP は明らかに NP 完全である.

そこで, PPPP の解とは, “与えられた角形ポリオミノのピースを, 後述する一定のルールの基で前記領域上に順次配置するピース選択の手順である” と解釈して, ゲーム論的に議論を進めることにする. 即ち, あらかじめ解答者（これを攻め方と呼ぶ）は 1 セット n 個の角形ポリオミノのピースが持ち駒として, また前記の領域が配置場所（守り方領土または単に領土と呼ぶ）として与えられ, 手順は次のように進められる. まず, 提示された領土に, 攻め方が一つ目の駒をその左下隅に置くことから始まる(この際, 置かれた駒は, 領土からはみ出ないものとする). それが攻め方の第 1 手である. これに対して, 置かれた駒によって占有された部分を除く領域が新たな領土となる. これに対して, 攻め方は, 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駒を配置する（この際, 置かれた駒は, 領土からはみ出ないものとする). これが攻め方の第 2 手である. 以下同様に進めて, 攻め方 n 手目で, 守り方領土を全て味方の駒で

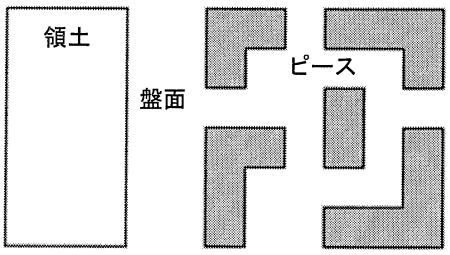
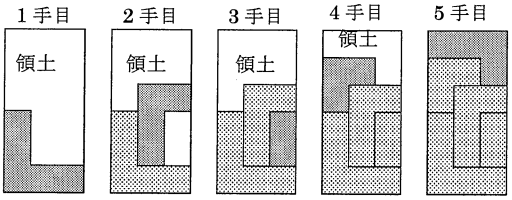


Fig. 1　Given game board



Puzzle-A

Fig. 2　An example of game solution

埋め尽くした場合，攻め方の勝ち，守り方の負けとなり，パズルが解けたことになる．このパズルは，チェスや将棋のチェックメイトパズルと類似する側面がある[13]-[15]．即ち，各段階の手順は，上記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）として解釈できる．また，ある段階の結果として得られる盤面は，やはりポリオミノ箱詰め問題である．最初の盤面はP0であり，全体が再帰的であることは真にチェックメイトパズルである（Fig. 2参照）．

2.2 ゲーム論的解法の原理[9]

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．例えば，第1ステージ，即ち第1手目の盤面の例をFig. 3に示してある．この第1手目は，実は解のパスに載っているが，決して，アプリオリに与えられるわけではない．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．筆者らは，このため，2種類のゲーム論的評価（尺度）を導入した．

各ステージにおける候補手順の評価尺度を，「盤面評価尺度」といい，その手順を試行した場合の盤面上での空間的な配置位置の妥当性を判定する基準であり，もう一つの「手順評価尺度」は，妥当な盤面評価に合格した候補手順について，攻め手としての有効度を評価する尺度である．

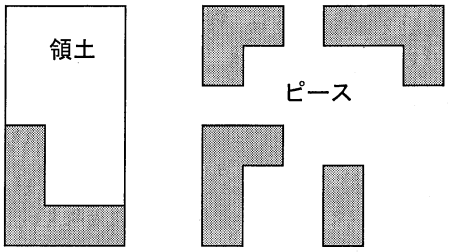


Fig. 3　Game board of the 1-st stage

このうち，盤面評価について，若干詳しく述べる．与えられたステージでの盤面評価は，その候補手順が解のパスに載っているかどうかの可能性，即ち，守り方が提示している領土を，手持ちの駒を全て使って，単体複体的に埋め尽くすための可能性を，その盤面の状態だけから直接評価するものである．ゲームにおける局面の大域的評価に相当する．この際，領土も駒も全て，D型ポリオミノであることに注意する．この可能性を必要十分条件の形で言い表すと，“領土と合同な持ち駒の単体複体多角形を構成できる”ということになる．これを少し緩めて，必要条件の形で述べると，次のようになる．

(1) 領土の面積は駒の面積の総和に等しい．

(2) ディジタル化されたD型ポリオミノの頂点や辺に関わる位相的特徴量に関して，領土のそれと一致する持ち駒の単体複体多角形を構成できる．

このうち，(1) に関しては，原問題に解が存在するという前提があるので，前項で述べたように，“領土左下隅に領土からはみ出ないように，且つ既に置かれた駒がある場合，単体複体的に接するように，攻め方の駒を配置する”という配置ルールが守られている限り，常に成立している．そこで，盤面の評価を (2) を基準にして考える．即ち，この盤面がPPPPであるならば，手持ちの駒をある順序で単体複体的に接合して行けば，その結果として得られる合成多角形は領土と合同となる．結局，接合パターンに関わる頂点の種別増減量と，前記，手持ちの駒全体にわたる各特徴量の総和と対応する領土の特徴量の差が一定の規則を満たすという事実を基に，盤面を評価する基準が盤面評価尺度である．さらに，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，情報量として，エントロピーを評価尺度として導入したものが手順評価尺度である．

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次の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．

2.3 PPPP アルゴリズムの定式化

最適化の観点からすると，PPPP の可能解は即ち最適解ということであり，解の存在が保証されているといえども，問題の性質上，膨大なピース配置の組み合わせ数に比べれば，解の総数はきわめて小さい．従って，前節で述べた 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ことがわかる．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，また，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．

Fig. 4 に探索木の生成と展開のプロセスを示す．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．Fig. 4 の下で，PPPP アルゴリズムは次のように定式化される．

[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．
}

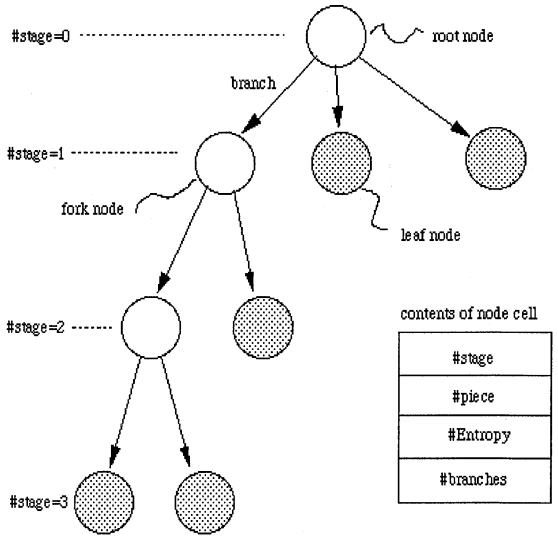


Fig. 4 Search tree

Step 3:
Step 1 に戻る．

実は，ここに述べた手順は実装されたアルゴリズムの中核部分であり，実際には，数手先読みを行うことによって，連動する 2 段階の評価の効率化と精度向上を図っている．更に，配置ピース選択のプライオリティに関するヒューリスティクスも取り入れて PPPP は，高速化されている[16]. Fig. 5 に適用結果の 1 例を示す．

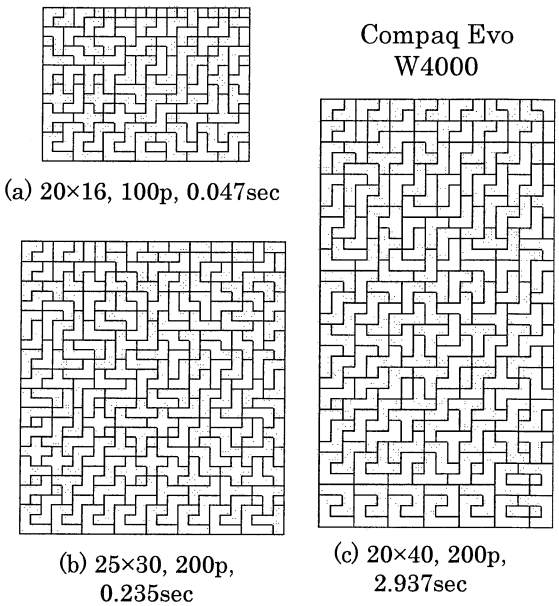


Fig. 5 An example of PPPP

3. 標準的ポリオミノ箱詰め問題の変形による一般化

理想化された配置問題である PPPP を，実際の配置問

題の要請に近づけるために，前節までに得た結果の一般化を行う．

3.1 D型ポリオミノ最適配置問題(DPLP: D-type polyomino layout problem) [9]

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，第1章で述べた（2）のタイプの配置問題では，その前提が無くて，全ピースを配置するための最少の面積の領域を求めることが問題である．筆者らは，これを矩形領域面積最小化問題: DPLP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．この操作結果の1例をFig. 6に示す．

3.2 A型ポリオミノ最適配置問題(APLP: A-type polyomino layout problem) [9]

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ことがある（むしろ，この方が普通）．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にして，このための矩形領域面積最小化問題: APLPを考える．解法は至ってシンプルであり，次のように2段階の操作に依る．

[APLP algorithm]

Step 1:

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る．

このDピース群に対してDPLPを解く．

Step 2:

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．次に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．

Fig. 7に操作（Step 2）の結果の1例を示す．

3.3 ソリッドポリオミノ箱詰め問題（SPPP: Solid polyomino packing problem) [17]

立方体を単体とした，PPPPの3次元版ソリッドポリオミノ箱詰め問題に対して，PPPPアルゴリズムを展開できる．

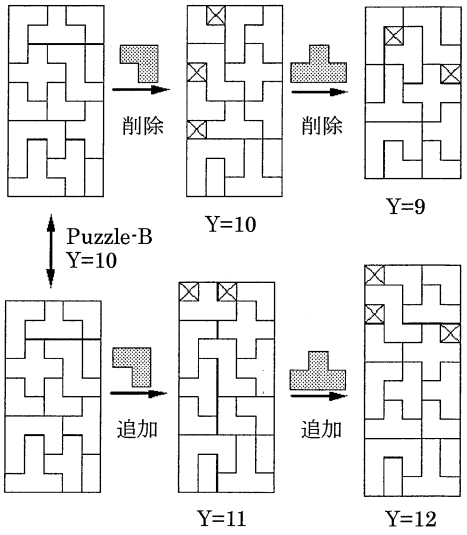


Fig. 6　An example of DPLP

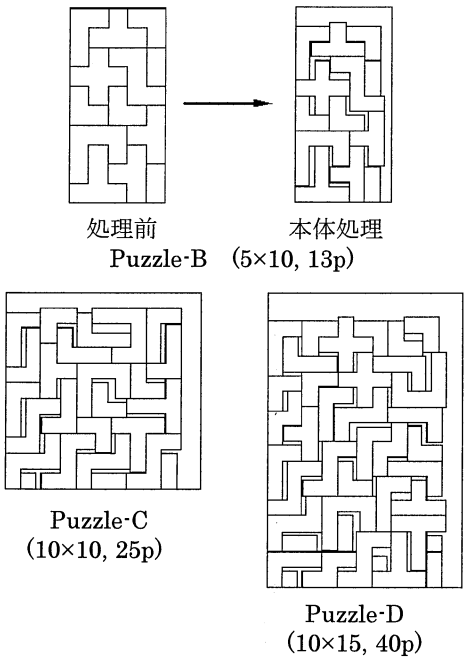


Fig. 7　An example of APLP

ソリッドポリオミノの3次元位相の特徴量を扱う複雑さを避けて，ソリッドポリオミノを平面ポリオミノに縮退させて取り扱うことを考える．まず，与えられたポリオミノピースで3次元領土が埋められた状態を考える．今，簡単のために，領土の形状が直方体だと仮定して，各ピースと領土自身を領土の境界面に平行な3方向の平面で，且つ，境界面を含む，単位寸法間隔で並べた複数の平面で切断したとする．さらに簡単のために，ピースは，あらかじめ回転を許さない，固定した姿勢でつめることにすると，3方向の切断面に対して個別に次のことが言える：

「複数の分離された領土の切断面の集合を新たに，平面領土とし，各ピースの各切断面の集合を新たに，平面ポリオミノ群と考えれば，この平面領土は，平面ポリオミノ群で

埋められる.」

即ち，3種類のPPPP問題が作られたことになる．したがって，SPPPの解の存在条件は，これら3種類のPPPPが解を持つことである．さらに，この関係から，配置に関して軸周りの90度回転を許す場合でも，3個のPPPPの第1，第2段評価を連動させて，SPPPを解くことができる．

本アルゴリズムを適用した結果の1例をFig. 8に示す．

本章では，指定される領土は矩形ポリオミノとして扱ってきたが，理論的には，D型ポリオミノであればどんな形でも良いし，従ってまた，あらかじめ決められた場所（ただし，単体寸法で計った格子位置）に，いわゆるプリセットするピースの配置も許容される．

4.　ピース間配置制約を考慮したPPPPの拡張（ePPPP）

「1. はじめに」で指摘した条件である（3）ピース間配置制約を考慮したアルゴリズムの構築を図る．この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．本章では，このアルゴリズム（ePPPP）が，PPPPアルゴリズムの自然な拡張によって達成できることを示す．

4.1 好み評価尺度の導入

はじめに，ピース間配置制約をピース間隣接配置の必要度をピース間の好み関数で表されるものとして，(n, n) 型の好みマトリックスMを定義する．即ち，

M=[m(i, j)].

ここに，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．因みに，m(i, i)は，外部との入出力のために領土の境界に接して置かれるべきモジュール（ピース）を取り扱うために導入したものである．マトリックス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．

つぎに，PPPPの可能解に対して，好み評価尺度:Path Penaltyを次のように定義して，ePPPPに関する目的関数とする（Fig. 9）.

CityDistance(i, j) = |xi-xj|+|yi-yj|,　(i≠j の場合)

CityDistance(i, i) = min(|xi-XL|, |xi-XR|, |yi-YB|, |yi-YT|)

m(i, j)=m(j, i) に注意して，

PathPenalty

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

ここで，(xi, yi)および(xj, yj)はそれぞれ，ピースiとピースjの配置座標である．また，(XL,YB), (XR,YB), (XR,YT)は(XL,YT)はそれぞれ領土の左下点位置，右下点位置，右上点位置，および左上点位置を意味する．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，

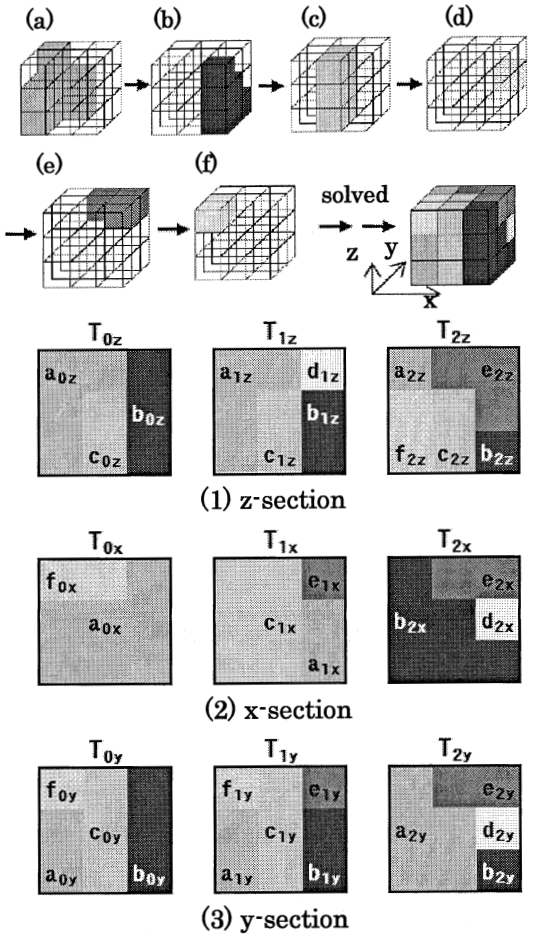


Fig. 8　An example of SPPP

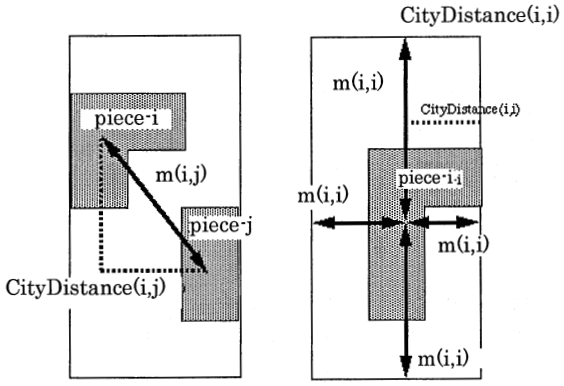


Fig. 9　Preference m(i, j) and CityDistance

PathPenalty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．

4.2 ePPPPにおける目的関数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，(目的関数である)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．このアルゴリズムについては次章で述べるが，本章では，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]
(PathPenaltyの下限を推定)
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
= $\Sigma_{(\text{for all } i,j \leqq k \text{ such that } i \geqq j)} m(i,j)*\text{CityDistance}(i,j)$
if (k == n) LowerBound=LowerBound0 and 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
= $\Sigma_{(\text{for } k<i \leqq n \,\&\&\, 1 \leqq j \leqq k)} m(i,j)*\text{CityDistance}(i,j)$
LowerBound2
= $\Sigma_{(\text{for } k<i \leqq n \,\&\&\, i \leqq j \leqq n)} m(i,j)*\text{CityDistance}(i,j)$
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;

4.3 ePPPPアルゴリズム

目的関数PathPenalty下限の推定法を用いて，PPPPアルゴリズムはePPPPアルゴリズムとして拡張できる．先にも断ったように，実際のコンピュータアルゴリズムの骨格のみを示したものである．なお，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ことであり，return;
// パスはひとつ前の段階で保存されている．
}
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．

5. 数値実験による検証と考察

前章で構築したアルゴリズムをコンピュータプログラムに変換し，その探索効率についての妥当性を検証する

（実行可能性については本論の前半ですでに証明済み）．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：

(1) ピースの総数と領土の大きさ，即ち，問題の複雑さ：
今回の実験では次の 3 ケースのデータを用いる．
・Puzzle-B:
盤面は 5×10 の領土と 13 ピースのポリオミノ．
・Puzzle-C:
盤面は 10×10 の領土と 25 ピースのポリオミノ．
・Puzzle-D:
盤面は 10×15 の領土と 40 ピースのポリオミノ．

(2) ピース配置姿勢に関する自由度：
90 度単位の回転の自由度

(3) 攻め手評価における先読みの深さ：[9]
PPPP アルゴリズムの結果で推奨された，深さ 3 を用いる．

(4) 攻め手候補の優先順位のつけ方：[9]
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．

(5) 好み関数の設定：
次に示す 2 種類の M マトリックスを用意した．

[Matrix-1]

```
m(i, j) = (n-j+i+1)*random_number;   (i<j),
          (n-i+1)*random_number;     (i=j),
          m(j, i);                   (i>j),
```

ここで，i, j=1~n，また random_number は(0, 1)の範囲の乱数である．

[Matrix-2]

```
m(i, j) = (j-i)*random_number;       (i<j),
          (i-1)*random_number;       (i=j),
          m(j, i);                   (i>j),
```

ここで，i, j=1~n，また random_number は(0, 1)の範囲の乱数である．

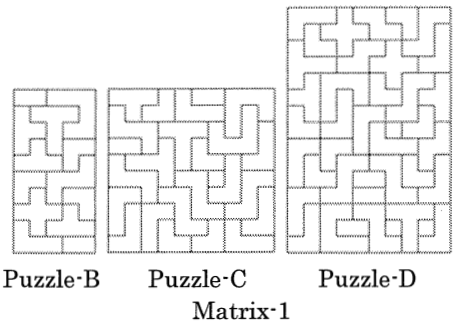
Matrix-1 は，ピース ID の差の小さいピース間では，好み値が多くなり，ピース ID が小さいピースは領土境界との好み値が大きくなるように設定されている．一方，Matrix-2 は，これとは反対に，ピース ID の差の大きいピース間では，好み値が多くなり，ピース ID が大きいピースは領土境界との好み値が大きくなるように設定されている．

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，実行された．Table 1 および Fig. 10 にその結果を示す．ここで，配置問題 Puzzle-B，Puzzle-C および Puzzle-D は，(2) で注意した 3 種類の複雑度を代表する．これらの結果から次のような知見が得られる．即ち，

(a) 先読み，エントロピー戦略および，目的関数 Path penalty 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．

Table 1 Results of numerical experiment

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



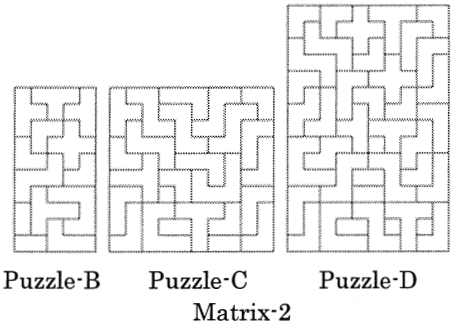


Fig. 10 Final solution

(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 Path penalty 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．

(c) D 型ポリオミノ最適配置問題 DPLP，A 型ポリオミノ最適配置問題 APLP および，ソリッドポリオミノ箱詰め問題 SPPP を，それぞれピース間配置制約を考慮して拡張した問題 eDPLP，eAPLP および eSPPP に対しても，本アルゴリズムの発展形として，容易にアルゴリズムを構成できる．

6. おわりに

本論文では，工学的な配置問題をポリオミノ箱詰め問題でモデル化して扱うアプローチの方法について論じた．前半は配置占有面積最小化に関してこれまでに得られた研究結果を纏めたものである．即ち，

（1）あらかじめ解の存在が保証されている，寸法的に規格化された標準的な（D 型）平面ポリオミノ箱詰め問題 PPPP，また，その一般化として，
（2）解の存在を仮定しない D 型ポリオミノ最適配置問題 DPLP，さらに，
（3）寸法的な規格化制約を除外した A 型ポリオミノ最適配置問題 DPLP，
また前記 PPPP の発展形として，
（4）立方体を単体とした，PPPP の 3 次元版ソリッドポリオミノ箱詰め問題 SPPP
を対象とした高速な解法を，ポリオミノの位相的特徴量に着目したゲーム論的に構成できることを示した．

後半では，上記（1）～（4）では取り扱わなかった，たとえば，VLSI フロアプランにおける pin-to-pin 配線にかかわる制約などのピース間の配置制約を考慮したモデルについて論じた．まず，PPPP を対象として，これに上記制約をピース間に関わるコスト PathPenalty としてあらわし，これを目的関数として加味した拡張モデルを ePPPP とし，目的関数最小化を，PathPenalty の下限値評価に基づく分枝限定法で達成できることを示した．このアルゴリズムは，PPPP のそれに若干の拡張を図るだけで十分であり，数値実験においても有効性が確認できた．なお，この手法は，上記の DPLP，APLP そして SPPP に対する，同様な意味での拡張モデル: eDPLP，eAPLP および eSPPP にも容易に転用できるものである．

今回の報告で，実際的な配置問題に対する，抽象化モデルとしてのポリオミノ箱詰め問題モデルの構築は，ほぼ完結したものと考える．もちろん，実際の問題に適用する場合には，対象特有の性質を加味した，アルゴリズムの解釈が必要である．今後の課題であるが，上記の拡張モデル: eDPLP，eAPLP または eSPPP に対しても，本論の手法の検証を行うこと，さらには，実際の問題での適用を図ることである．

参考文献

[1] 山内重樹，手塚研治: 遺伝的アルゴリズムを用いた自動ネスティングシステムの開発，日本造船学会論文集，No.178, pp.707-712 (1995).
[2] Jain, P., Fenyes, P. and Richter, R.: Optimal Blank Nesting Using Simulated Annealing, *Advances in Design Automation 1990 DE-Vol.23-2 ASME*, pp.109-116 (1990).
[3] 長島智樹，丹波太，平野光，岡田哲男：FiNest アルゴリズムによる鋼板自動ネスティングの実用化，石川島播磨技報，Vol.44, No.3, pp.229-236 (2004).
[4] Heller, W.R., Sorkin, G. and Maling, K.: The Planar Package for System Designers, *Proc.19th D.A.Conf.*, pp.253-260 (1982).
[5] Otten, R.H.J.M.: Automatic Floorplan Design, *Proc. 19th D.A.Conf.*, pp.261-267 (1982).
[6] Wong, D.F. and Liu, C.L.: A New Algorithm for Floorplan Design, *Proc.D.A.Conf.*, pp.101-107 (1986).
[7] Fujiyoshi, K., Komada, C. and Ikeda, A.: A fast algorithm for rectilinear block packing based on selected sequence-pair, *INTEGRATION, the VLSI journal*, Vol.40, pp.274-284 (2007).
[8] Golomb, S.W.: Polyominoes, Princeton Univ. Press (1994).
[9] 村井保之，巽久行，徳増眞司：位相的特徴量に基づく平面ポリオミノ箱詰め問題の解法，情報処理学会論文誌，Vol.40, No.12, pp.4009-4022 (2002).
[10] Murai, Y., Tatsumi, H. and Tokumasu, S.: Game-theoretic Algorithm to Rectilinear Blocks Layout Problem, *Proc. of the 2nd Korea-Japan Joint Symposium on Multiple-Valued Logic*, pp.157-161 (2001).
[11] Murai, Y., Tatsumi, H. and Tokumasu, S.: Game-theoretic Approach to Placement Problem of Rectilinear Blocks---Solution of Rectilinear Jigsaw Puzzle---, *Proc. of the IEEE-NANO '2001 Conference*, pp.128-133 (2001).
[12] Kimoto, K., Murai, Y., Tsuji, H. and Tokumasu, S.: Rectilinear Jigsaw Puzzle: Theory and Algorithms, *IEEE World Automation Congress (WAC2006)*, (CD-ROM) (2006).
[13] Feigenbaum, E.A. and Feldman, J.: Computers and Thought, McGraw-Hill (1963).
[14] Greenblatt, R.D.: The Greenblatt Chess Program, FJCC, pp.801-810 (1963).
[15] 越智利夫，亀井達弥，内ヶ崎儀一郎，徳増眞司：計算機が解く詰め将棋，数学セミナー，Vol.46, No.6, pp.44-48 (1969).
[16] Murai, Y., Tsuji, H., Tatsumi, H. and Tokumasu, S.: Fast Placement Algorithm for Rectilinear Jigsaw Puzzles, *Journal of Advanced Computational intelligence and Intelligent Informatics*, Vol.10, No.3, pp.323-331 (2006).
[17] Kimoto, K., Tsuji, H., Murai, Y. and Tokumasu, S.: A Solution of Three-Dimensional Polyomino Packing Problems, *Proc. of IEEE International Conference on Systems, Man and Cybernetics*, pp.3725-3730 (2007).

謝辞　本研究の一部は文部科学省科学研究費補助金・基盤研究（C）18500122 のもとで行った．