集積回路の高密度レイアウト設計手法に関する研究

１９９９年3月

埼玉大学大学院理工学研究科（博士後期課程）

情報数理科学専攻（主指導教官　前川　仁）

中谷　直司

要　旨

大規模集積回路（LSI）レイアウト設計において計算機支援設計（CAD）システムによる支援は必須となっており，現状においてもCADシステムによる自動化がなされている．しかし，今だ多くの課題が残されており今後の研究・開発が望まれている．そこで本研究では，解決すべき課題の一つとして知られる，配置手法における配置対象の形状自由度を向上させる新しい配置手法の提案を行っている．

また，近年注目される最適化手法の一つである，遺伝的アルゴリズム（GA）の配置手法への応用を研究している．それに伴い遺伝的アルゴリズムよりも高速な探索手法として，ウイルスによる急速な進化が特徴的な，ウイルス進化論に基づく新しい進化型アルゴリズムを提案している．

第１章では本研究の社会的背景と本研究の目的について述べている．また，各章の概要も示している．

第２章では本研究の対象であるLSIレイアウト設計の現状について各種設計方式の違いや設計段階の分類について述べている．またその中で，本論文において提案する“任意形状ブロックの配置手法”の位置付けについても触れた．

第３章では基本的な配置問題を対象に遺伝的アルゴリズムを用いた配置手法の提案と，計算機実験の結果について述べている．GAを用いた従来手法においては探索効率の低下や，探索がランダムサーチ的な振る舞いをするようになるという問題があったが，本研究では情報に否定情報あるいは冗長性を持たせることにより，探索効率の低下を回避し，ランダムサーチ的な振る舞いを抑制して，最適解を得ることが可能な新しいコーディング法を提案している．否定情報を持たせたコーディング法の場合は配置不能な解の生成こそは抑制可能だが，実用に足る結果を得ることはできなかったが，情報に冗長性を持たせたコーディング法の場合は，従来手法の問題点であった配置不能な解の生成と，ランダムサーチ的な振る舞いを共に抑制することにより，従来よりも優れた探索結果を得ることが可能なことが計算機実験により確認されている．

第４章では任意形状を持つ要素を対象とした配置手法の提案と，計算機実験の結果について述べている．提案手法は形状に応じた評価値を定義し，その値を用いることで従来手法では難しかった凹部を含む要素の配置や，配置領域を制限しての配置を可能にするものである．また組立式配置手法でありながら，配置順序に遺伝的アルゴリズムによる制御を用いることで，組立式手法に共通する欠点を回避可能なことが計算機実験により確認されている．

第５章ではウイルス進化論に基づく新しい進化型アルゴリズムを提案している．このアルゴリズムではウイルスを遺伝子を運ぶ生物器官の一つと捉え，ウイルスの感染による遺伝子操作を，GAと組み合わせて探索能力の向上を目的とするものではなく，唯一の探索操作とした点が，GAや従来のウイルス進化論に基づく手法とは大きく異なっている．また，ウイルスの感染により集団全体へ遺伝子断片を急速に拡散可能であり，また遺伝子操作の処理の軽さから，GAに比べ計算時間が短かく，収束も速く，また，最適解を得る確率も高いということが計算機実験により確認されている．

第６章では，前の章で提案したウイルス進化型アルゴリズムの基本的な配置問題への適用と，ウェハー規模集積回路（WSI）の再構成問題への適用を行っている．その結果，これらの応用問題においても，ウイルス進化型アルゴリズムは遺伝的アルゴリズムに比べ，計算時間が短かく，収束も速く，また，最適解を得る確率も高いということが確認されている．

最後に，第７章では本研究の総括と今後の展望を述べている．

以上のように本研究は，新たなLSIレイアウト設計手法の開発を目的とし，配置対象の形状自由度を向上させる新しい配置手法の提案，遺伝的アルゴリズムの配置手法への応用，さらには新しい進化型アルゴリズムの提案を行ったものである．これらの成果はLSIレイアウト設計環境を向上させると共に，進化型アルゴリズムの可能性を広げるものになると期待される．

目 次

第 1 章

序論

1.1 研究の背景と目的

近年のLSI製造技術の向上は回路規模の増大や複雑化を生み，それに伴いLSIレイアウト設計においての人手のみの作業を事実上不可能にしている．また，ユーザーの目的に応じてその都度設計するカスタムLSIの需要拡大は，コスト，時間の両面からもCADシステムによる可能な限りの自動設計の必要性を生じている．しかし，現在のCADシステムには解決すべき数多くの問題が残されており，今後の研究・開発が望まれている[1]．そこで，本研究ではLSIレイアウト設計に用いられる自動設計手法を研究対象とし，問題の解決と，予想されるいっそうの規模の増大や複雑化にも対応可能な，新たな手法の開発を目的とする．

1.1.1 任意形状ブロックの配置手法

LSI設計は次に示すように，いくつかの段階にわけてトップダウンで設計が行われている．

システム設計：LSIをシステムとしてとらえ，その動作を設計する．
機能・論理設計：システム設計で決定した動作を実現するために必要な論理回路を設計する．
レイアウト設計：論理回路の配置，配線を設計する．
テスト設計：LSIの動作をテストするためのデータを設計する．

さらに，レイアウト設計も回路規模と複雑さの増大に伴い，現状では階層的に処理されており，一般的には上から順に次の五つの階層に分けられる．

フロアプラン処理：機能ブロック単位の概略配置を決める処理.

配置処理：形状と端子位置が確定したセル群を配置する処理.

概略配線処理：配線に利用可能な領域をチャネルに分け，各配線ネットがどのチャネルを使うかを決定する処理.

詳細配線処理：各チャネル内で個々の配線経路を決定する処理.

コンパクション処理：配置・配線結果を図形集合と見て，接続関係を維持したまま空き領域を詰めていくことにより，チップ面積を縮小する処理.

本研究では，この階層の中のフロアプランおよび配置処理で用いられる配置手法を研究対象とする．配置手法は，セルと呼ばれる設計済みの部分回路を制約条件の下でレイアウトする手法を指し，現在利用されている手法のほとんどはセルの形状として矩形のみを対象としている．したがって現在のレイアウト設計は，セルの設計段階から“形状は矩形”という制約が事実上存在することになり，自由な設計・製作を妨げている．あるいは矩形以外を対象とした処理手法[10, 11, 12]もいくつか提案されているが，その形状はＬ型[10]や凸XY多角形と呼ばれる形状[11]などに限定され，配置対象の形状自由度は決して高くなく凹部を含む対象の配置などには対応できない．また，形状に制限がない手法[12]も提案されているが，文献[11]の手法も含めコンパクション処理のための手法であり，初期配置として何らかの配置が与えられなければ利用できない．すなわち現状では，凹部を含む形状に制限のない配置対象（任意形状ブロック）のための配置手法は存在しない．そこで本研究では，“任意形状ブロック”のための配置手法の研究を行う．

1.1.2 遺伝的アルゴリズムを用いた配置手法

また近年，生物の遺伝と進化を計算機上で模倣して人工物の適応，学習，最適化を行おうとする遺伝的アルゴリズム（Genetic Algorithm:GA）[2, 3, 4, 5]が注目されており，多くの分野で工学的応用が試みられるようになってきた．本研究の対象であるLSIレイアウト設計はNP困難な問題でもあり，その最適解の探索にGAを用いることは，興味深い研究課題でもある．しかし，GAにはコーディングやパラメータの調整など多くの自由度があり，数学的な解析がほとんどなされていない現状もあって，応用を成功させるにはGAについての一定の理解と経験が必要とされる．ここでGAの動作過程を図1.1に示す．なお，各過程では次のようなことが行われる．

初期集団：与えられた問題の解となりうる候補をランダムに複数生成する.

適応度の評価：問題の解としての正しさの度合い（適応度）を評価する．
選択：解の適応度と選択される回数の期待値が対応するように選択を行う．
交叉：選択された解を二個ごとに組にして，互いの持つ情報を入れ替える．
突然変異：ランダムに解の持つ情報を変化させる．

すなわちGAは，より正しい解の持つ情報を，選択，交叉，突然変異を繰り返すことで解集団の中に蓄積させ，最終的に有用な情報を全て組み合わせた解を得るという動作を行っている．このとき問題になるのは，次のような要素を経験的に決定しなければならないことである．

1. 一般にGAでは解を一次元配列で表現するが，そのコーディング方法．
2. 適応度の評価式．
3. 選択の方法．
4. 交叉の方法とその確率．
5. 突然変異の方法とその確率．
6. 探索に用いる解の候補の総数．
7. 選択，交叉，突然変異の繰り返し回数．

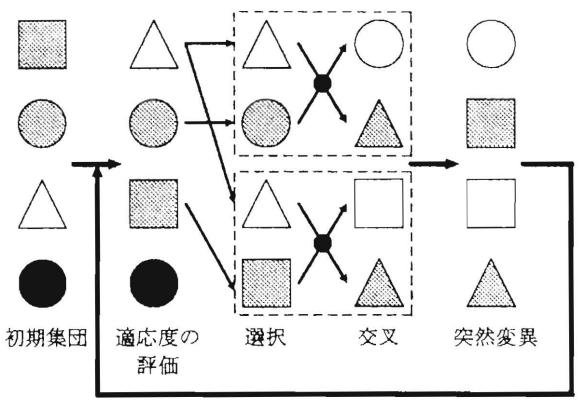


図 1.1　GAの動作過程

そこで本研究では，GAの実用的な配置手法への応用を目的に，基本的な配置問題への適用から研究をはじめる．しかし，単純にGAを適用した手法においては，交叉や突然変異といったGAのオペレーションにより，多くの配置不可能な解を生成し探索効率を低下し，局所最適解に陥る可能性も高くなる．これは探索点の持つ解に関する情報が，GAのオペレーションにより互いに矛盾を生じること

に原因がある．あるいは，従来からある配置不可能な解の生成を抑制する工夫を行うと，探索がランダムサーチ的な振る舞いをするようになるという問題があった．これは，探索点の持つ解に関する情報を新たな探索点に継承させる際に，情報を変更することで配置不可能な解の生成を回避するため，探索の連続性が一部失われることによる．研究ではこれらの問題点を考慮した新しい手法を提案する．

1.1.3 ウイルス進化型アルゴリズム

GAは，その広域探索能力の高さから，局所最適解を避け最適解を得ることが可能という特長を持つが，同時に計算が長時間におよぶという欠点も持っている．先に述べたようにGAは生物の進化の過程を模倣しているわけだが，その対象となっているものはダーウィンの進化論であり，このことが計算が長時間におよぶ原因と考えられる．すなわち，ダーウィンの進化論において爆発的進化が起こり得ない様に，GAにおいても急速な探索が行われることはないということである．しかし，進化論として知られるものはダーウィンの進化論ばかりではなく，生物学では様々な進化論が唱えられている．“ウイルス進化論[15]”はそれら多くの進化論の一つであり，生物はウイルスの感染により進化するという考えに基づいている．ウイルスはインフルエンザなどの例からもわかるように，爆発的な感染力を持つことで知られている．したがって，このウイルスの感染で進化するとするウイルス進化論は，爆発的な進化を予想させるものである．そこで本研究では，計算時間の短縮を目的にウイルスによる急速な進化を模倣した，ウイルス進化論に基づく新しい進化型アルゴリズムの研究を行う．なお，ウイルス進化論に基づく進化型アルゴリズムは既にいくつか提案されている[16, 17, 18]．しかし，これらのアルゴリズムはGAにウイルスによる進化を加え，GAの探索能力の向上を目指したものであり，ウイルス進化論の特徴であるウイルスによる急速な進化を実現したものではない．すなわち，本研究とは根本的に目的が異なるものである．

1.2 研究の内容と成果

この節では，本研究で行った研究内容とその成果について述べる．本研究は，新たなLSIレイアウト設計手法の開発を目的とし，配置対象の形状自由度を向上させる任意形状ブロックの配置手法の提案，遺伝的アルゴリズムの配置手法への応用，さらにはウイルス進化論に基づく新しい進化型アルゴリズムの提案を行った．これらの成果はLSIレイアウト設計環境を向上させると共に，進化型アルゴ

リズムの可能性を広げるものになると期待される．以下では，個々の内容と成果について述べる．

1.2.1 任意形状ブロックの配置手法

第1.1.1節で述べたように，現状では，任意形状ブロックのための配置手法は存在しない．そこでブロック形状に応じた評価値を定義することにより，任意形状ブロックの配置を行う一手法を提案した．この手法は組立式配置手法と呼ばれる手法の一種であり，配置結果は配置順序に依存する．そこで本研究では遺伝的アルゴリズムを用いることで，この問題に対処した．すなわち，本手法の特長は以下の通りである．

1. 従来手法では難しかった凹部を含む，任意形状ブロックの配置が可能である．
2. 非矩形の領域制限を含む，配置領域を制限しての配置が可能である．
3. 配置順序に遺伝的アルゴリズムによる制御を組み合わせることで，組立式手法に共通する欠点である配置結果の順序依存問題に対応できる．
4. 従来のコンパクション手法のための初期配置を与えることに利用可能である．

また，本手法の有用性を確認するため計算機実験を行い，凹部を含む完全な任意形状ブロックの配置や，配置領域に制限を設けての配置も可能であることを示した．

1.2.2 遺伝的アルゴリズムを用いた配置手法

GA を用いた配置手法の研究として，基本的な配置問題への GA の適用を試みた．第1.1.2節で述べたように，従来手法を適用すると，配置不可能な解を生成し探索効率が低下するか，あるいは探索がランダムサーチ的な振る舞いをするようになるという問題がある．そこで本研究では配置不可能な解の生成を抑制し，かつ，ランダムサーチ的な振る舞いをも抑制する，次の２つの新しいコーディング法を提案した．

1. 配置位置を否定する情報を持たせたコーディング
   二つの否定情報は互いに矛盾しないことに着目した手法．
2. 配置位置情報に冗長性を持たせたコーディング
   情報の矛盾を避けるため冗長性を持たせた手法．

提案手法の有用性を確認するため，提案手法と従来手法を比較する計算機実験を行った．“否定情報を持たせたコーディング法の場合”は配置不能な解の生成こそは抑制可能だが，デコーディング手順の煩雑化にともなう計算機時間の増大と，従来手法に比べ情報量が増加するための情報の継承の難しさにより，実用に足る結果を得ることはできなかった．“情報に冗長性を持たせたコーディング法の場合”は従来手法の問題点であった配置不能な解の生成と，ランダムサーチ的な振る舞いを共に抑制することにより，従来よりも優れた探索結果を得ることが可能なことが確認された．

1.2.3 ウイルス進化型アルゴリズム

ウイルス進化論に基づく新しい進化型アルゴリズムを提案した．このアルゴリズムではウイルスを遺伝子を運ぶ生物器官の一つと捉え，ウイルスの感染による遺伝子操作を，GA と組み合わせて探索能力の向上を目的とするものではなく，唯一の探索操作とした点が，GA や従来のウイルス進化論に基づく手法とは大きく異なっている．また，ウイルスの感染により集団全体へ情報を急速に拡散可能であることから，局所探索能力が向上し，さらに情報操作の処理の軽さから，その計算時間はGA に比べ短くなる．

提案手法の有用性を確認するため，ナップザック問題と巡回セールスマン問題という基本的組み合わせ問題を解くことにより，提案手法とGA を比較する計算機実験を行った．また基本的な配置問題，およびウェハー規模集積回路の再構成問題への適用を試みた．その結果，GA に比べ提案手法では次のことが確認された．

1. 計算時間が減少する．
2. 収束速度が上昇する．
3. 最適解を得る確率が高くなる．

1.3 論文の概要

本論文は任意形状を持つ要素を対象とした配置手法の提案と，配置手法への遺伝的アルゴリズムの適用，ウイルス進化論に基づく新しい進化型アルゴリズムの提案および，それらに対する各種計算機実験結果とその考察が主な内容となっている．以下では各章ごとの概要を述べる．

第１章 序論

本研究を行うにあたっての社会的背景と本研究の目的について述べる．また，各章ごとにまとめた論文の概要を示す．

第２章 LSI レイアウト設計の現状

本研究の対象であるLSI レイアウト設計の現状について述べ，主要な内容でもある“任意形状ブロックの配置手法”の位置づけについても触れる．なお，本研究の対象としているLSI レイアウト問題は，セミカスタム・スタンダードセル方式の中の，ビルディングブロック設計方式を対象とした配置およびフロアプラン手法である．

第３章 遺伝的アルゴリズムを用いた配置手法

この章では遺伝的アルゴリズムの一般論について述べた後，基本的配置問題を対象に遺伝的アルゴリズムを用いた配置手法の提案と，計算機実験の結果について述べる．従来手法においては探索効率の低下や，探索がランダムサーチ的な振る舞いをするようになるという問題があったが，情報に否定情報あるいは冗長性を持たせることにより，従来手法よりも高い探索効率を持ちつつ，ランダムサーチ的な振る舞いを抑制して，局所最適解を回避して最適解を得ることが可能になる手法を提案する．なお，既に述べたようにGA を応用で成功させるにはある程度の経験が必要であり，この章の研究は次章以降のための基礎研究としての意味も持っている．

第４章 任意形状ブロックの配置手法

任意形状を持つ要素を対象とした配置手法の提案と，計算機実験の結果について述べる．提案手法は形状に応じた評価値を定義し，その値を用いることで従来手法では難しかった凹部を含む要素の配置や，配置領域を制限しての配置を可能にするものである．また組立式配置手法でありながら，配置順序に遺伝的アルゴリズムによる制御を用いることで，組立式手法に共通する欠点を回避している．

第５章 ウイルス進化型アルゴリズム

ウイルス進化論に基づく新しい進化型アルゴリズムの提案と，計算機実験の結果について述べる．提案するウイルス進化型アルゴリズムは，ウイルス進化論が

主張する生物はウイルスの感染により進化するという考えに基づき，ウイルスの感染を唯一の探索動作とする遺伝的アルゴリズムとは異なる新しい進化型アルゴリズムである．また，計算機実験を行い遺伝的アルゴリズムと比較したところ，提案手法は計算時間が短かく，収束も速く，また，最適解を得る確率も高いということが確認された．

第６章 ウイルス進化型アルゴリズムを用いた配置手法

前の章で提案したウイルス進化型アルゴリズムを応用した配置手法の提案と，計算機実験の結果について述べる．対象とする配置問題は，第3章でも扱った“基本的な配置問題”および“ウェハー規模集積回路の再構成問題”である．計算機実験の結果ここでも，ウイルス進化型アルゴリズムは遺伝的アルゴリズムに比べ，計算時間が短かく，収束も速く，また，最適解を得る確率も高いということが確認された．

第７章 結論

本研究の総括と今後の展望を述べる．

第 2 章

LSIレイアウト設計の現状

2.1 まえがき

この章ではLSIレイアウト設計の現状として，各種設計方式の違いや設計段階の分類について述べる．その中で，本論文において提案する“任意形状ブロックの一配置手法”の位置付けについても触れる．

2.2 設計方式

LSIは大きく２つに分類される．１つは汎用LSI，もう１つはカスタムLSIである．汎用LSIは文字どうり一般的に使われるため大量に生産される．したがって，チップ面積を小さく設計する必要があり，また一度の設計に比較的時間をかけることが可能なため，人手を主体とした設計がなされている．もっとも，近年のLSI規模の増大や複雑化に伴い，CADシステムによる支援も必須となっている．カスタムLSIは用途に応じてその都度設計されるものである．そのため，コストの面からも設計は短時間に，しかも人手に頼らずに行う必要がありCADシステムの利用が盛んである．

カスタムLSIはその設計方式から図2.1のように分類される．フルカスタム方式とは全てを一から設計する方式であり，セミカスタム方式は既に設計済みの要素を用いて設計する方式である．このうちCAD化は特に後者において進んでいる．セミカスタム方式は更に幾つかに分類されるが，ここでは主な方式としてスタンダードセル方式とゲートアレー方式について説明する．

スタンダードセル方式は設計済みのセルを用いて設計する方式である．セルは

レイアウト情報が回路構造，機能，使用条件などと共にセルライブラリーに蓄積されており，これらスタンダードセルを組み合わせる形で希望する機能を実現する．このとき，セルの高さがほぼ一定のものを用いるのがポリセル方式（図 2.2），任意形状（通常は矩形）のものを用いるのがビルディングブロック方式（図 2.3）である．本論文において提案する任意形状ブロックの一配置手法は，このビルディングブロック方式の一種といえる．

ゲートアレー方式はゲートがアレー状に配置された半製品のチップに，配線のみを行うことで機能を実現する方式である．ゲート列の間に配線領域がある方式と，チップ全域にゲートが敷き詰められている SOG(Sea of Gates) 方式とがある．

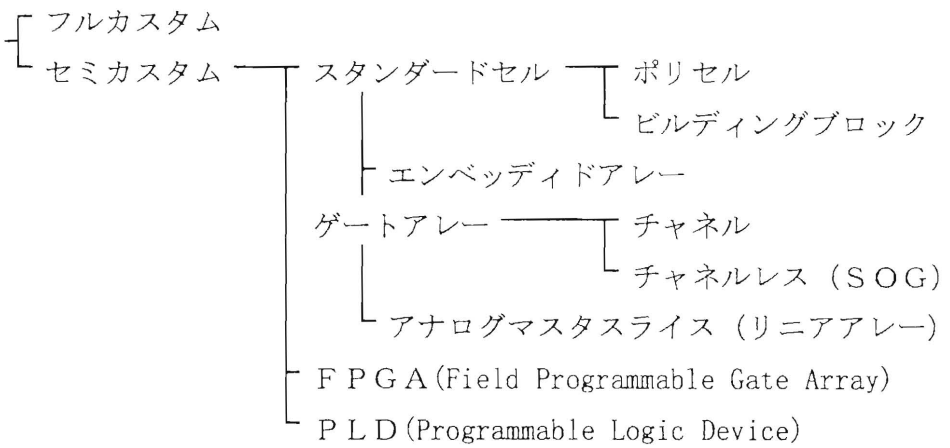


図 2.1　カスタム LSI の設計方式

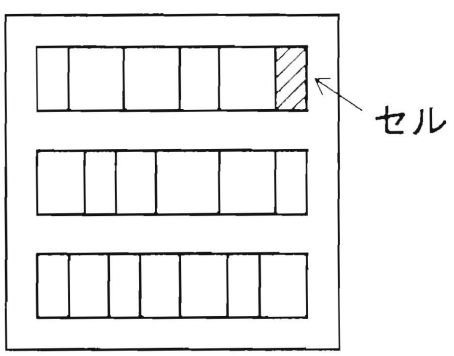


図 2.2　ポリセル方式

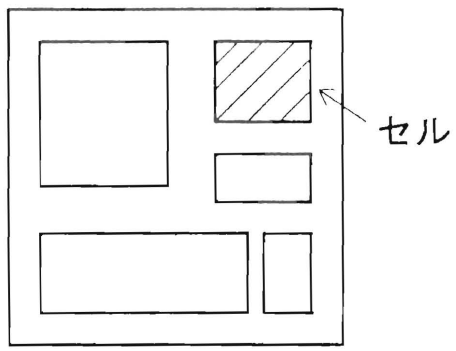


図 2.3　ビルディングブロック方式

2.3　設計段階

LSI 設計は幾つかの段階にわけて行われる．まず最初が機能設計，これにより LSI に持たせる機能を決定する．次に論理設計，機能設計で決定した機能を実現するために必要な論理回路を設計する．そしてレイアウト設計により実際のチップを設計する．レイアウト設計は回路規模の増大と複雑化に対処するため階層的に行われている．スタンダードセル方式を例にとると上から順に次の５つの階層に分けられる．

フロアプラン処理：機能ブロック単位の概略配置を決める処理．ブロックには内部レイアウトと形状の固定されたハードマクロと，それらが未決定なソフトマクロとがある．ソフトマクロはその回路規模から面積が推定され形状が大まかに決められる．内部レイアウトはこれを制約条件として下位階層により決定される．

配置処理：形状と端子位置が確定したセル群を配置する処理．フロアプラン処理の特殊な場合ともいえる．

概略配線処理：配線に利用可能な領域をチャネルに分け，各配線ネットがどのチャネルを使うかを決定する処理．

詳細配線処理：各チャネル内で個々の配線経路を決定する処理．

コンパクション処理：配置・配線結果を図形集合と見て，接続関係を維持したまま空き領域を詰めていくことにより，チップ面積を縮小する処理．必ずしも必要とは限らない．

基本的に階層は上から順に処理されるが，必要に応じて相互に行き来する．例えばフロアプラン，配置，概略配線処理を行き来しての設計はよく行われる．提案

する任意形状ブロックの一配置手法は，配置処理及びフロアプラン処理を行なうものである．

2.4　配置処理

配置処理において最適解を得るには指数関数的計算時間を必要とするため，数十以上のモジュールを扱うことはコスト的に実際的ではない．そこで配置処理は一般に，初期配置と配置改善の２段階に分けて行われる．すなわち初期配置によりある程度の配置結果を得てから，それに対し配置改善を繰り返し行い改善率が０，あるいはあらかじめ定めたある値以下になったところで最終解とする．この方法で得られた解は最適とはいえないが，実用的な処理時間で満足のいく解を得ることができる．配置結果の良否は厳密には配線を行なった後で評価すべきものだが，個々の配置結果に対し実際に配線してみることは処理時間が膨大となり実用的ではない．そこで精度は落ちるが，仮想配線長やチップ面積などを用いて評価を行なう．この場合当然ながら仮想配線長は短いほうが，チップ面積は配置結果を内包する最小矩形の面積が小さい方が評価が高い．

初期配置手法には大きく分けて組立式手法と分割法の２つの手法が存在する．組立式手法は，まず種になるモジュールを配置し，それに強い接続を持つ未配置モジュールをその近くに配置していく方法である．この手法の特長は処理が単純であるため処理時間が極めて短いことにある．しかし一方では，配置処理過程では全体的な見通しがなされないため，配置結果の品質はあまり良くない．提案する任意形状ブロックの一配置手法は，この組立式初期配置手法の一種に分類されるが，配置順序の制御に遺伝的アルゴリズムを用いることで，配置結果の品質を向上させている．分割法は配置すべきモジュールをグループに分け，さらにグループ（サブグループ）をサブグループに分割することを繰り返し，それらグループやサブグループ単位で配置していく階層的手法である．複雑な処理を繰り返し行なう必要から処理時間は長くなるが，配置結果の品質は優れたものになる．

配置改善手法は基本的に配置モジュールを入れ替えた結果，評価値が高くなれば入れ替えを採用し，改善されなければ入れ替えないという処理を繰り返すものである．このとき入れ替え対象モジュールを２つに限定する場合，複数で行なう場合，あるいは幾つかのモジュールを一固まりとして扱う場合などがある．この方法では配置状態が評価が高くなる方向にしか変化しないため，多くの場合局所的最適解に収束することが知られている．これを回避し最適解に近い解を求める

手法としてシミュレーテッドアニーリング法，遺伝的アルゴリズム（第3.3節参照）などがある．

2.5 むすび

この章ではLSI レイアウト手法の現状として，各種設計方式の違いや設計段階の分類について述べた．本論文において扱う配置問題は，ビルディングブロック設計方式のための配置およびフロアプラン手法である．

第 3 章

遺伝的アルゴリズムを用いた配置手法

3.1 まえがき

遺伝的アルゴリズム（GA）を用いた配置手法はいくつか提案されているが，それらの手法においてはGAのオペレーションにより，多くの配置不可能な解を生成し探索効率が低下する．また，配置不可能な解の生成を抑制する工夫を行うと，探索がランダムサーチ的な振る舞いをするようになるという問題がある．そこで本章では配置情報に否定情報あるいは冗長性を持たせることにより，配置不可能な解の生成，およびランダムサーチ的な振る舞いをも抑制する新しいコーディング法を提案する．提案するコーディング法の有用性を確認するため計算機実験を行った結果，従来手法よりも高い探索効率を持ち，局所最適解を回避して最適解を得ることが可能なことが確認された．

3.2 研究の目的

遺伝的アルゴリズム（GA）は自然界における生物の進化に着想を得た最適解探索手法であり，その特長として広範囲にわたる解の探索により，局所最適解に陥ることなく最適解に到達することが可能なことがあげられる．しかし，この特長も問題に対する染色体設計（コーディング）の方法次第では，探索効率が低下し局所最適解に陥ることも考えられ，コーディング法はGAにおいて非常に重要な位置を占める．

今日までにGAを用いた配置手法がいくつか提案されているが，それらの手法においては交叉や突然変異といったGAのオペレーションにより，多くの配置不

可能な解を生成し探索効率を低下させてしまうという問題があった．すなわち，配置不可能な解が多数存在することにより，事実上の探索点が減少し広範囲にわたる探索空間が維持できなくなるため，局所最適解に陥る可能性も高くなる．あるいは，配置不可能な解の生成を抑制する工夫を行うと，探索がランダムサーチ的な振る舞いをするようになるという問題があった．これは，探索点の持つ解に関する情報を新たな探索点に継承させる際に，情報を変更することで配置不可能な解の生成を回避するため，探索の連続性が一部失われることによる．

そこで本研究では配置不可能な解の生成を抑制し，かつ，ランダムサーチ的な振る舞いをも抑制する新しいコーディング法を提案する [25, 26, 31, 34]．この手法においては配置情報に否定情報あるいは冗長性を持たせることにより，配置不可能な解の生成を抑制し，情報の継承を実現している．

3.3 遺伝的アルゴリズムの概要

この節では，最初に遺伝的アルゴリズムの概要について説明する．遺伝的アルゴリズム (Genetic Algorithm) は，生物進化（選択淘汰・突然変異）の原理に着想を得たアルゴリズムであり，確率的探索・学習・最適化の一手法と考えることができる．

3.3.1 基礎用語

基本的な遺伝的アルゴリズムに用いられる用語について説明を行う．遺伝的情報を伝える実体として染色体 (chromosome) が，存在する．実際の生物ではこれは塩基 (base) で構成される物理的実体であるが，GA では，データ領域や配列を用いる．染色体の各位置にどのような遺伝情報が存在するかは決まっているがこの位置のことを遺伝子座 (locus) と呼ぶ．個々の遺伝子座は，どのような形態や機能の発現を制御するか等が決まっており，生物体においてはそれぞれの遺伝子座にどのように塩基配列が存在するかによって，その個体の遺伝子的特徴を決定する．この，各遺伝子座に対してその形質を決定する塩基配列で表現されたコードを遺伝子 (gene) と呼ぶ．そして，遺伝子の組み合わせのパターンを遺伝子型 (genotype) という．遺伝子型にもとづいて形成された個体を表現型 (phenotype) と呼ぶ．

本物の染色体は，二重螺旋の DNA が凝縮することによって形成されている．GA では通常これを 1 次元の配列に抽象化することが多く，その配列上の位置が遺伝

子座となる．遺伝子は，各位置がとりうる値のことである．遺伝子型は，配列上に表現された値のパターンのことである．

3.3.2 GA の動作過程

GA は，基本的には Generate-and-Test 型のアルゴリズムで，一般に次の 3 種類の遺伝的操作 (genetic operation) を使用する．

- 選択 (selection)
- 交叉 (crossover)
- 突然変異 (mutation)

GA ではこれらの操作を解の候補集団に対して行う．解の候補は，遺伝子型 (genotype) として染色体 (chromosome) に 1 次元的に表現される．GA の動作過程は図 3.1の様に示される．

GA では最初に，初期集団の生成を行う．一般には，決められた個体数の染色体をランダムに生成する．次にこの初期集団の各個体の問題への適応度を調べ，適応度と残せる子孫の数の期待値が対応するように選択交配する個体対を決定する．この個体対それぞれの染色体の特徴をあわせ持った，新しい個体を交叉によって作成する．次に，交叉によって生成した個体に対してある一定の確率で突然変異を発生させる．これらの操作が終了すると，新しい世代の個体集団が作られたことになる．そして，この新しい集団に対し，適応度評価，選択，交叉，突然変異を行い，さらに新しい世代を作っていく．こうした，世代交代を重ねていくうちに，問題の解決に有効なパターンが染色体中に発生し，これが選択と交叉によって個体集団中に蓄積されて，これらが組合されることによって次第に大局的な解が発見されていく．

3.3.3 遺伝的操作

3.3.3.1 スケーリング

スケーリングとは，何らかの関数を導入して適応度の違いを拡大または縮小させることをいう．これにより個体対選択時の確率を変化させることができる．例えば，集団全体で適応度に差があまり無いときには解の探索がうまく行かないため，スケーリングによって適応度の違いを拡大するという処理をする．基本的な手法としては，線形スケーリング，シグマ切断，べき乗スケーリングなどがある．

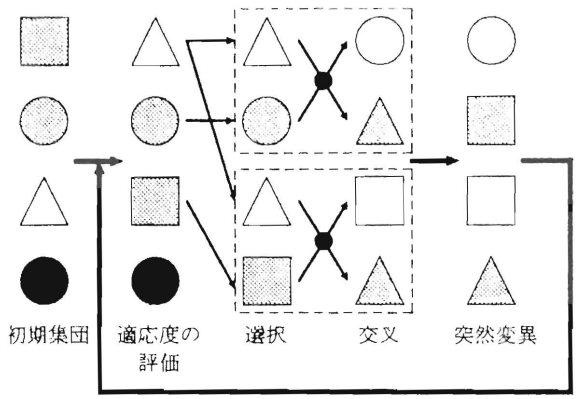


図 3.1　GA の動作過程

3.3.3.2　選択

選択は，適応度に応じてどの個体対を交配させるかを決定するものである．最も基本的なものとして適応度比例戦略がある．この方法では各個体の適応度に比例した確率で子孫を残せる可能性がある．ある個体 i が，各々の選択で選ばれる確率 $pselect_i$ は，次式であらわせる．

$$pselect_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

選択確率の大きな個体は複数回の交配に参加するため，集団の中にその遺伝子が広がっていくことになる．選択の方法としては他に，期待値戦略，ランク戦略，トーナメント選択戦略などがある．

これらの戦略と組み合わされてよく使われる戦略に，エリート保存戦略がある．これは集団中で最も適応度の高い個体をそのまま次世代に残す方法である．この方法を用いると，交叉や突然変異でその時点で最も優れた個体が破壊されるのを防ぐことができる．しかし，エリート保存された個体の遺伝子が集団中に急速に広がる可能性が高く局所解に陥る危険性もある．

3.3.3.3　交叉

交叉 (crossover) は，ある確率 p_c で 2 つの親の遺伝子を組み替えて子の遺伝子を作る操作のことを言う．例えば，1 点交叉と呼ばれる交叉では図 3.2 に示されるように，交叉する位置を 1 つ決めてその前と後で，どちらの親の遺伝子型，つまりどちらの親の形質を受け継ぐかを変える方法である．図 3.2 では，子 1 は前の

部分では親 1 の前部分の遺伝子，後の部分では親 2 の後部分の遺伝子を引き継いでいる．また，その逆が他の新しい遺伝子となる．交叉の方法は他に，交叉位置を複数にする複数点交叉や，一様交叉などがある．

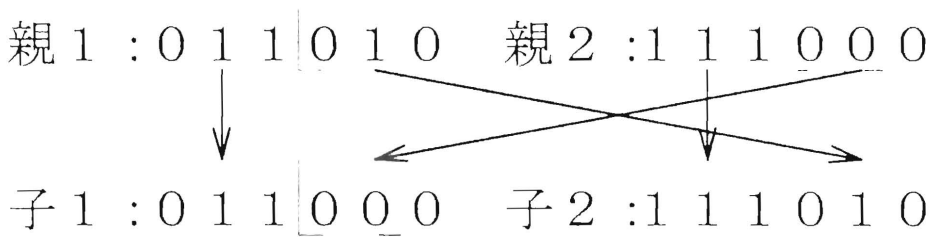


図 3.2 交叉

3.3.3.4 突然変異

突然変異 (mutation) は，遺伝子を一定の確率 p_m で変化させる操作である．図 3.3 では，左側の遺伝子が破線で囲まれた部分のビットが反転することによって突然変異を行って右側の遺伝子に変異している．突然変異は，あまり大きな変異確率に設定すると遺伝子中の優れた特質が破壊されランダムサーチとなるが，ある程度の変異は GA において必要である．なぜなら，GA では，3.3.2 節で説明したように交叉と選択によって個体集団中に蓄積された有効な部品が組合されることによって解を発見する手法を取っているために，初期集団に含まれる部品のみの組合せだけだと求まる解の質にも限界がでてくる．そのため突然変異は，確率的に変異を起こさせることによって個体集団中に多様性を与えることによって初期集団だけでは探索できない解を求めることを可能にしている．

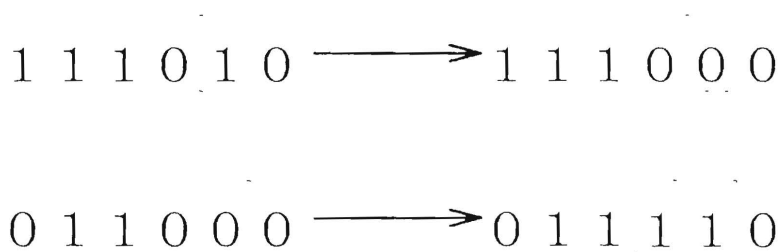


図 3.3 突然変異

3.3.4 GA の簡単な動作例

GA の動作を理解するために，簡単な関数を使って動作を追ってみる．ここでは例として染色体の長さを 10 ビットとして，その中にある 1 のビットが 5 個までは適応度が 0.5 で，5 個以上になると，1 つ増えるごとに 0.1 ずつ増加する関数を考える．これは以下の様な関数，式(3.1) となる．ただしこの値は数値が大きいほど評価が高くなっている．

$$f(x) = \max\left(0.5, 0.1 \times \sum_{i=0}^{9} x_i\right) \tag{3.1}$$

局所探索法では，このような解空間では，探索に失敗する危険がある．なぜなら，探索開始の点が 3 個だけ 1 のビットがあるような場所であったなら，たとえ他のビットを変化させても $f(x) = 0.5$ のままで変化しないように解空間の勾配がなく，ランダムにもう 1 度別の点から探索を再開する以外に方法がないからである．しかし，GA では，最初から多点サンプルを導入しているために探索が可能だと考えられる．

ここで，ランダムに初期集団として 10 個を生成した，これとその適応度を表 3.1 に示す．個体 0 は，遺伝子型が 1101111010 であり，1 が 7 個ある．これを式 3.1 で計算すると適応度は 0.7 となる．このような評価を全ての個体に対して行なう．

第 2 世代は，初期集団の個体が選択，交叉，突然変異を行なった結果作られる，これを表 3.2 に示す．表 3.2 の個体 0 は，第 1 世代の個体 1 と 8 が 6 番目での位置で交叉して作られた．つまり第 1 世代の個体 1 の最初から 6 番目（6 ビット目）までの遺伝子と個体 8 の 7 番目以降の遺伝子の組合せが第 2 世代の個体 0 の遺伝子型である．この個体の生成時には突然変異は起きていない．この世代の最大適応度は 0.800，平均適応度は 0.600 であり，初期集団の最大適応度 0.700，平均適応度 0.580 よりも改善されている．

また，表 3.3, 3.4 に第 3 世代，第 4 世代の集団とその適応度を示している．世代が進むにつれて最大適応度や平均適応度が改善されてきているのがわかる．GA ではこの様にして最適解の探索を行う．

表 3.1　初期集団（最大適応度 0.700，平均適応度 0.580）

no.	遺伝子型	親 1	親 2	交叉位置	適応度
0	1101111010	0	0	0	0.700
1	1111110001	0	0	0	0.700
2	1011010111	0	0	0	0.700
3	1101100101	0	0	0	0.600
4	1101100110	0	0	0	0.600
5	1110001010	0	0	0	0.500
6	1001010010	0	0	0	0.500
7	1011000110	0	0	0	0.500
8	0101101001	0	0	0	0.500
9	1010001101	0	0	0	0.500

表 3.2　第 2 世代（最大適応度 0.800，平均適応度 0.600）

no.	遺伝子型	親 1	親 2	交叉位置	適応度
0	1111111001	1	8	6	0.800
1	1111100110	4	7	2	0.700
2	1101111010	9	0	0	0.700
3	1011010111	2	1	0	0.700
4	1111100001	2	1	0	0.600
5	1001000110	4	7	2	0.500
6	1010001101	9	8	0	0.500
7	0101101001	9	8	0	0.500
8	1010001101	9	0	0	0.500
9	0101100001	1	8	6	0.500

表 3.3　第 3 世代（最大適応度 0.900，平均適応度 0.650）

no.	遺伝子型	親 1	親 2	交叉位置	適応度
0	1111111110	0	5	7	0.900
1	1111111001	0	1	0	0.800
2	1101111010	8	2	1	0.700
3	1111100110	0	1	0	0.700
4	1010101111	8	3	8	0.700
5	1010101101	8	2	1	0.600
6	1011010101	8	3	8	0.600
7	1001000001	0	5	7	0.500
8	1010001101	8	5	0	0.500
9	1001000110	8	5	0	0.500

表 3.4　第 4 世代（最大適応度 0.900，平均適応度 0.750）

no.	遺伝子型	親 1	親 2	交叉位置	適応度
0	1111111110	0	1	0	0.900
1	1111111110	6	0	0	0.900
2	1111111110	4	0	0	0.900
3	1111111001	0	1	0	0.800
4	1110101101	4	5	4	0.700
5	1010101111	4	5	4	0.700
6	1111100110	3	5	0	0.700
7	1010101111	4	0	0	0.700
8	1010101101	4	0	0	0.700
9	1011010101	6	0	0	0.600

3.4　対象とする配置問題

GA を用いた配置手法を研究するにあたり，まずは基本的な問題を扱うことにする．具体的には次のような問題を対象とする．配置領域は図3.4に示すように，碁盤の目のような正方形の区画 $\{(x_1, y_1), \ldots, (x_N, y_N)\}$ が縦横に連なったものとし，そのサイズは $N \times N$ （図3.4の場合 $N = 3$）とする．そして，配置するモジュール $(m_1, \ldots, m_{N^2})$ も図3.5に示すように正方形のものが N^2 個あるとし，1つのモジュールが1つの区画を占めるものとする．したがって，この場合チップ面積は固定値となるため，配置の善し悪しは仮想配線長により判断される．仮想配線長の算出にあたっては，各モジュール間の接続関係は既知であるとして，モジュール中心間のマンハッタン距離を用いる．このような配置問題は実際のLSI配置では実用的ではないが，配置手法の評価方法としては一般的であるため，以下ではこの問題を対象として研究を行った．

(1, 1)	(2, 1)	(3, 1)
(1, 2)	(2, 2)	(3, 2)
(1, 3)	(2, 3)	(3, 3)

（x, y）：配置区画座標

図 3.4　配置領域の例

3.5　GA を単純に適用した配置手法

前節で述べた配置問題にGAを単純に適用する場合を考える．配置する各モジュールには，図3.5に示すようにモジュール番号 $(1, 2, 3, \ldots, N^2)$ が与えられているとする．また配置領域の各区画は，図3.4のように x 座標と y 座標の組み（(1,1)(1,2)...(N, N)）で表すことができる．したがって，配置結果は各モジュールに対し“モジュール m_i ならば座標 (x_i, y_i) の配置区画に配置する”という情報を

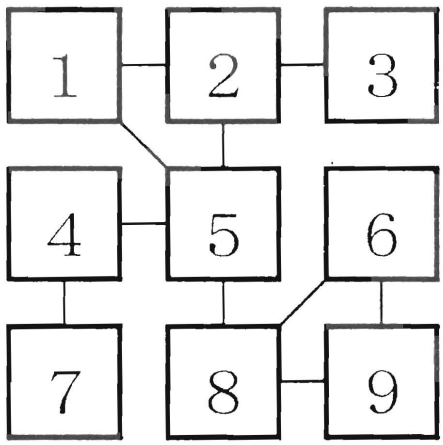


図 3.5 配置対象モジュールの例

N^2 個用意すれば表現可能である．具体的には“モジュール 1 は (2,3) に，モジュール 2 は (3,1) に，モジュール 3 は (1,2) に，...，モジュール N^2 は (2,2) に配置する”などとして表現でき，染色体としは N^2 の長さの 1 次元配列を用意し，“配列の 1 番目の要素にはモジュール 1 の配置座標を，配列の 2 番目の要素にはモジュール 2 の配置座標を，...，配列の N^2 番目の要素にはモジュール N^2 の配置座標を入れる”ことで表現可能である．この場合，各モジュールの配置位置情報は，1 つの遺伝子座に，他のモジュールとは無関係に存在することになる．したがって，同じ染色体上の複数の遺伝子座の持つ情報は互いに独立であるがゆえに，矛盾する可能性が高くなる．つまりは，交叉や突然変異の結果できる染色体を実際の配置に戻したときに，配置位置の重複が起こり配置不可能な解を生成する確率が高くなる．

例えば，図 3.6，3.7，に示した配置の染色体はそれぞれ次のようになる．

親 1： (2,2) (1,1) (3,2) (1,3) (3,1) (1,2) (2,1) (3,3) (2,3)
親 2： (2,2) (2,3) (1,1) (2,1) (3,2) (3,1) (1,3) (3,3) (1,2)

ここで仮に 4 番目の遺伝子座と 5 番目の遺伝子座の間で交叉するとする．

親 1： (2,2) (1,1) (3,2) (1,3) | (3,1) (1,2) (2,1) (3,3) (2,3)
親 2： (2,2) (2,3) (1,1) (2,1) | (3,2) (3,1) (1,3) (3,3) (1,2)

すると，生成される子は

子 1： (2,2) (1,1) (3,2) (1,3) | (3,2) (3,1) (1,3) (3,3) (1,2)
子 2： (2,2) (2,3) (1,1) (2,1) | (3,1) (1,2) (2,1) (3,3) (2,3)

となる．このとき子 1 について考えると，モジュール 3 と 5，モジュール 4 と 7 が同じ区画に配置されることになり配置不可能である．また，突然変異によっても配置位置に重複が起こり得ることは明らかである．与えられた配置問題は，配置可能な仮想配線長が最短になる配置を求めることなので，この例のように配置不可能になる解は求める解にはなり得ない．しかし，こうした配置位置の重複はこの染色体設計では頻繁に起こり，個体集団の大多数を占めるようになる．当然，配置不可能な解になる個体の適応度は低いため，選択交配される個体はごく一部の配置可能な個体でほぼ占められることになる．つまり，集団の多様性が失われることになる．集団の多様性の低下は探索点の事実上の減少にあたり，探索効率が低下し局所最適解に陥りやすくなるという問題を生じる．

2	7	5
6	1	3
4	9	8

図 3.6　親 1 の配置

3	4	6
9	1	5
7	2	8

図 3.7　親 2 の配置

これを回避するには，

- 交叉や突然変異のときに何らかの工夫をして配置不可能解の発生を抑える．
- 交叉や突然変異によって配置不可能解の発生しにくい染色体設計をする．

などが考えられる．前者の工夫をした手法はいくつか提案されている[6, 7, 8, 9]が，これらはGAにおける交叉や突然変異そのものを変更するため，GAの特長の一つでもある遺伝的操作の自由な設計を行うことができない．そこで，本研究では後者の染色体設計（コーディング）を工夫した手法を提案する．

3.6　提案する手法

この節では初めに，情報の矛盾を避けるコーディングとして知られる従来手法について説明する．しかし，この従来手法には情報を正確に継承することができないという問題がある．そこで，この点をも考慮した，次の２つの新しいコーディング法を提案する．

1. 配置位置を否定する情報を持たせたコーディング
2. 配置位置情報に冗長性を持たせたコーディング

3.6.1　従来のコーディング法

まず，コーディングの工夫により情報の矛盾を避ける手法として知られる，“巡回セールスマン問題”におけるGAの構成法[19]を配置問題に適用してみる．

この手法における配置問題に対するコーディング法は次のようになる．各座標に配置されたモジュールを，座標$(1,1),(2,1),\ldots,(N,1),(1,2),(2,2),\ldots,(N,N)$の順に並べたものを配置結果$(p_1,\ldots,p_{N^2})$とする．またモジュールを適当な順序で並べたリスト$W$（定数）を用意する．そして配置結果$P$の$i$番目にあるモジュールが，未配置モジュールリスト$W-(p_1,\ldots,p_{i-1})$の何番目であるかを表し，これを遺伝子$l_i$とする．このとき，$1 \leq l_i \leq N^2-i+1$が成立する．このようにして得られるリスト$L=(l_1,\ldots,l_{N^2})$を染色体とする．このコーディング法により染色体と配置結果とは1対1に対応するため，交叉や突然変異による配置不可能な解の生成は避けられる．

例として配置結果として

P_1=(3 7 8 6 5 2 4 1 9)

P_2=(2 9 6 5 8 4 7 1 3)

の2つを考える（それぞれ図3.8と3.9）．

W=(1 2 3 4 5 6 7 8 9)

としてコーディングすると

L_1=(3 6 6 5 4 2 2 1 1)

L_2=(2 8 5 4 5 3 3 1 1)

となる．これを3番目と4番目の遺伝子座の間で交叉させると，

L'_1=(3 6 6　4 5 3 3 1 1)

L'_2=(2 8 5　5 4 2 2 1 1)

となり，配置結果は

P'_1=(3 7 8 5 9 4 6 1 2)

P'_2=(2 9 6 7 5 3 4 1 8)

となる．このようなコーディング法を用いることで，確かに配置不可能な解の生成は避けられる．しかしこの方法では，交叉等により新しく生成される解は交叉する前の2つの解の持つ情報を正確に継承することができず，一部においては全く無関係な情報と入れ替わるため，その探索効率には単純にGAを適用した場合ほどではないにしろ問題が生じる．

3	7	8
6	5	2
4	1	9

図 3.8　配置結果 P_1

3.6.2　否定情報を用いたコーディング法

GAを単純に適応した場合に配置位置に重複が生じるのは，配置位置を肯定的に決定する情報を染色体が持っているからである．すなわち，“モジュール3ならば(3,2)に配置する”などの“Aならばb”という肯定的な情報は，容易に“Cならばb”（例えば“モジュール5ならば(3,2)に配置する”）といった他の肯定的な情報と重複する．そこで本研究では，染色体に否定的な情報を持たせたGAを提案する．つまり染色体に，“Aならばbではない”という情報を持たせることにな

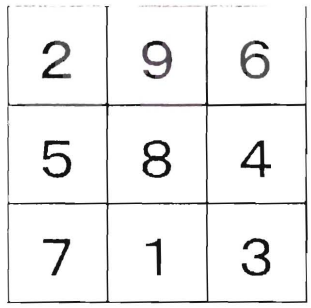


図 3.9　配置結果 P_2

る．例えば“モジュール 3 ならば (3,2) に配置しない”，“モジュール 5 ならば (3,2) に配置しない”という情報ならば，この 2 つの情報は互いに矛盾しない．この点に着目し，具体的には次に示す手法を提案する．

染色体の各遺伝子座には，“モジュール m_i ならば x_i 行および y_i 列には配置しない”という情報を持たせる．したがって染色体は

染色体：　(m_1, x_1, y_1)　(m_2, x_2, y_2)　(m_3, x_3, y_3)　(m_4, x_4, y_4)　$\cdots$

となる．配置位置を否定するのに x 行および y 列という表現を用いたのは，区画を 1 つ 1 つ否定しては染色体の長さが長くなりすぎることを考慮したものである．なお，染色体の長さは当然 N^2 では必要な情報を表すことができないため，それよりも長いものが必要になる．その長さ C_l は，配置情報として必要なものを含む長さで，かつ情報過多で配置位置を否定し過ぎない長さを適時選択することになる．ここで，与えられた染色体の否定情報から配置位置を決定する手順を述べる．

はじめに，各モジュールごとの配置可能位置情報を保存する変数 A_{mxy} $(m = 1, 2, \ldots, N^2 : x, y = 1, 2, \ldots, N)$ を導入する．この変数 A_{mxy} はモジュール m に関して座標 (x, y) の区画に配置可能ならば 1 を，配置不可能ならば 0 を持つものである．また，各モジュールの情報が遺伝子座の何番目に初めて出てきたかを保存する変数 O_m $(m = 1, 2, \ldots, N^2)$ も導入する．これらの変数を用いると配置位置決定手順は次のようになる．

ステップ 1

$A_{mxy} = 1$，$O_m = C_l + m$ $(m = 1, 2, \ldots, N^2 : x, y = 1, 2, \ldots, N)$ として初期化する．

ステップ 2

染色体の遺伝情報を展開する．遺伝子座 i 番目の (m_i, x_i, y_i) という情報は

$A_{m_i x_i y} = A_{m_i x y_i} = 0\ (x, y = 1, 2, \ldots, N)$,

If $O_{m_i} = C_l + m_i$,

Then $O_{m_i} = \iota$ として展開される.

ステップ3

If $\sum_{m,x,y} A_{mxy} \neq 0$,

Then

If $\sum_{x,y} A_{mxy}$ が最小になり,

かつ $\sum_{x,y} A_{mxy} \neq 0$ を満たす m を選択し,

Then

If O_m が最小になる m を選択し,

Then $m_p = m$.

Else **ステップ7**へ.

ステップ4

If $\sum_{x,y} A_{m_p xy} = 1$,

Then

If $A_{m_p xy} = 1$ を満たす (x, y) を選択し,

Then $(x_p, y_p) = (x, y)$.

Else

If $\sum_m A_{mxy}$ が最小になり,

かつ $A_{m_p xy} = 1$ を満たす (x, y) を選択し,

Then

If $x + N(y-1)$ が最小になる (x, y) を選択し,

Then $(x_p, y_p) = (x, y)$.

ステップ5

m_p を (x_p, y_p) に配置し,

$A_{mx_p y_p} = A_{m_p xy} = 0\ (m = 1, 2, \ldots, N^2 : x, y = 1, 2, \ldots, N)$ とする.

ステップ6

ステップ3へ.

ステップ7

If $k = 1$ (k:未配置モジュール数),

Then 未配置モジュールを空いている配置区画に配置する.

Else

If $k \geq 2$,

$Then\ N_p = k - 1$

(N_p:最終未配置モジュール数).

例として，3×3の配置領域に9個のモジュールを配置する場合を考える．染色体の長さC_lは15とし，その染色体は次に示すものとする．

(5,3,2)　(2,2,2)　(1,2,2)　(5,3,2)　(8,1,2)
(2,1,1)　(9,3,1)　(5,3,3)　(3,1,3)　(4,2,2)
(1,2,2)　(5,3,3)　(2,1,3)　(1,3,1)　(9,1,1)

これを展開すると次のようになる．

	(x,y)におけるA_{mxy}										
m	(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)	(1,3)	(2,3)	(3,3)	$\sum_{x,y} A_{mxy}$	O_m
1	0	0	0	0	0	0	1	0	0	1	3
2	0	0	0	0	0	0	0	0	0	0	2
3	0	1	1	0	1	1	0	0	0	4	6
4	1	0	1	0	0	0	1	0	1	4	7
5	1	1	0	0	0	0	0	0	0	2	1
6	1	1	1	1	1	1	1	1	1	9	15
7	1	1	1	1	1	1	1	1	1	9	16
8	0	1	1	0	0	0	0	1	1	4	4
9	0	0	0	0	1	0	0	1	0	2	5
$\sum_m A_{mxy}$	4	5	5	2	4	3	4	4	4		
配置結果											

ここからステップ3により$m_p = 1$となり，ステップ4で$\sum_{x,y} A_{m_p xy} = 1$を満たすため，$(x_p, y_p)$は$A_{m_p xy} = 1$を満たす(1,3)となる．そしてステップ5で配置とA_{mxy}の変更が行われ，次のようになる．

	(x,y)におけるA_{mxy}										
m	(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)	(1,3)	(2,3)	(3,3)	$\sum_{x,y} A_{mxy}$	O_m
1	0	0	0	0	0	0	0	0	0	0	3
2	0	0	0	0	0	0	0	0	0	0	2
3	0	1	1	0	1	1	0	0	0	4	6
4	1	0	1	0	0	0	0	0	1	3	7
5	1	1	0	0	0	0	0	0	0	2	1
6	1	1	1	1	1	1	0	1	1	8	15
7	1	1	1	1	1	1	0	1	1	8	16
8	0	1	1	0	0	0	0	1	1	4	4
9	0	0	0	0	1	0	0	1	0	2	5
$\sum_m A_{mxy}$	4	5	5	2	4	3	0	4	4		
配置結果							1				

この場合ステップ 3 で $\min \sum_{x,y} A_{mxy}$ (ただし $\sum_{x,y} A_{mxy} \neq 0$) を満たすモジュールは 5 と 9 の 2 つがある．そこで両者の O_m の値を比較すると，$O_5 < O_9$ なので $m_p = 5$ となる．ステップ 4 の $\sum_{x,y} A_{m_p xy} = 1$ の条件は満たさないので次へ進む．$A_{m_p xy} = 1$ を満たす (x, y) は (1,1) と (2,1) したがって $\min \sum_m A_{mxy}$ (ただし $A_{m_p xy} = 1$) より，$(x_p, y_p) = (1, 1)$ となる．ステップ 5 で配置と A_{mxy} の変更が行われ，次のようになる．

	(x,y) における A_{mxy}										
m	(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)	(1,3)	(2,3)	(3,3)	$\sum_{x,y} A_{mxy}$	O_m
1	0	0	0	0	0	0	0	0	0	0	3
2	0	0	0	0	0	0	0	0	0	0	2
3	0	1	1	0	1	1	0	0	0	4	6
4	0	0	1	0	0	0	0	0	1	2	7
5	0	0	0	0	0	0	0	0	0	0	1
6	0	1	1	1	1	1	0	1	1	7	15
7	0	1	1	1	1	1	0	1	1	7	16
8	0	1	1	0	0	0	0	1	1	4	4
9	0	0	0	0	1	0	0	1	0	2	5
$\sum_m A_{mxy}$	0	4	5	2	4	3	0	4	4		
配置結果	5						1				

前回と同様にステップ 3 で 4 と 9 のモジュールが競合するが，O_m の値からモジュール 9 が選ばれる．ステップ 4 の $\sum_{x,y} A_{m_p xy} = 1$ の条件は満たさないので次へ進む．ここで $\min \sum_m A_{mxy} = 4$ (ただし $A_{m_p xy} = 1$) となり，(x_p, y_p) を特定できない．そこで $x + N(y-1)$ の値から (2,2) を選択する．以下同様に繰り返すと，8 個のモジュールを配置した段階で次のようになる．

	(x,y) における A_{mxy}										
m	(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)	(1,3)	(2,3)	(3,3)	$\sum_{x,y} A_{mxy}$	O_m
1	0	0	0	0	0	0	0	0	0	0	3
2	0	0	0	0	0	0	0	0	0	0	2
3	0	0	0	0	0	0	0	0	0	0	6
4	0	0	0	0	0	0	0	0	0	0	7
5	0	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0	15
7	0	0	0	0	0	0	0	0	0	0	16
8	0	0	0	0	0	0	0	0	0	0	4
9	0	0	0	0	0	0	0	0	0	0	5
$\sum_m A_{mxy}$	0	0	0	0	0	0	0	0	0		
配置結果	5	3	6	7	9		1	8	4		

このときステップ 3 の，すべての m に対して $\sum_{x,y} A_{mxy} = 0$ となったらステップ 7 にという処理が選択される．そして，ステップ 7 でまだ配置していないモジュー

ルが 1 個だけならば，空いている配置区画にそのモジュールを配置する，が実行され次のようになる．

	(x,y)におけるA_{mxy}										
m	$(1,1)$	$(2,1)$	$(3,1)$	$(1,2)$	$(2,2)$	$(3,2)$	$(1,3)$	$(2,3)$	$(3,3)$	$\sum_{x,y} A_{mxy}$	O_m
1	0	0	0	0	0	0	0	0	0	0	3
2	0	0	0	0	0	0	0	0	0	0	2
3	0	0	0	0	0	0	0	0	0	0	6
4	0	0	0	0	0	0	0	0	0	0	7
5	0	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0	15
7	0	0	0	0	0	0	0	0	0	0	16
8	0	0	0	0	0	0	0	0	0	0	4
9	0	0	0	0	0	0	0	0	0	0	5
$\sum_m A_{mxy}$	0	0	0	0	0	0	0	0	0		
配置結果	5	3	6	7	9	2	1	8	4		

なお，この場合配置不可能だったモジュール数N_pは 0 である．

このように否定情報を持たせたコーディング法を用いれば，たとえまったく同じ遺伝子が 1 つの染色体上にあっても（前述の例ならば(5,1,2)など），配置位置には重複を生じることなく配置可能になるケースが多くなる．したがって，交叉や突然変異によって情報に重複が生じてもその多くは配置可能な解となるため，個体集団の多様性は保たれ局所解に陥ることなく最適解の探索が可能になると予想される．

3.6.3　冗長な情報を用いたコーディング法

単純にGAを適用した場合探索効率が低下し局所最適解に陥りやすいのは，配置位置情報が各モジュールに対して独立に与えられるためである．また，従来のコーディング法であるリストによる配置位置情報では，各モジュールの配置位置が他のモジュールの配置と密接に関係するため，情報が正しく継承されないという問題を生じていた．そこで，本研究では両者の問題点を踏まえ，次のようなコーディング法を提案する．染色体には“モジュールm_iならば座標(x_i, y_i)を中心として，十字をなす5つの配置区画のいずれかに配置する”という情報をN^2個持たせることにする．そして，実際のモジュールの配置位置は他のモジュール配置との関係から決定する（この手法については後述する.）．このことにより各モジュールの配置位置情報は独立せず，なおかつ，他のモジュール配置との関係をある程度維持しながら，情報を次の世代に継承することが可能になる．

例として“モジュール 2 を (2,3) を中心として，十字をなす 5 つの配置区画のいずれかに配置する”という情報（図 3.10）と，“モジュール 6 を (2,3) を中心として，十字をなす 5 つの配置区画のいずれかに配置する”という情報（図 3.11）が同じ染色体上に存在する場合を考える．このとき，仮にモジュール 2 が (2,3) に配置されたとすると，モジュール 6 は (2,3) を除く 4 つの区画のいずれかに配置されることになりこれら 2 つの情報は矛盾しない．すなわち配置不能な解の生成が抑えられ，集団の多様性も維持されることが予想され，局所最適解に陥ることなく最適解の探索が可能になると考えられる．また，交叉による情報の継承もある程度維持されるため，リストによる従来のコーディング法よりも探索効率が向上すると考えられる．与えられた情報から実際にモジュールを配置する手法は次の通りである．

	2		
2	2	2	
	2		

図 3.10　モジュール 2 を (2,3) の近傍に配置可能とする

	6		
6	6	6	
	6		

図 3.11　モジュール 6 を (2,3) の近傍に配置可能とする

はじめに，染色体の i 番目の遺伝子にはモジュール m_i の情報として“配置可能区画の中心座標 (x_i, y_i) と，この情報の優先度 O_i”が与えられているとする．したがって染色体は

染色体：　(x_1, y_1, O_1)　(x_2, y_2, O_2)　(x_3, y_3, O_3)　(x_4, y_4, O_4)　$\cdots$

となる．ここで，各モジュールごとの配置可能位置情報を保存する変数 A_{mxy} $(m = 1,2,\ldots,N^2 : x,y = 1,2,\ldots,N)$ を導入する．この変数 A_{mxy} はモジュール m に関して座標 (x,y) の区画に配置可能ならば 1 を，配置不可能ならば 0 を持つものである．これらの変数を用いると配置位置決定手順は次のようになる．

ステップ 1

$A_{mxy} = 0$ $(m = 1,2,\ldots,N^2 : x,y = 1,2,\ldots,N)$ として初期化する．

ステップ 2

染色体の遺伝情報を展開する．遺伝子座 i 番目の (x_i, y_i) という情報は

$A_{m_i x_i (y_i - j)} = 1$ $(j = -1,0,1$ ただし $1 \leq (y_i - j) \leq N)$,

$A_{m_i (x_i - j) y_i} = 1$ $(j = -1,0,1$ ただし $1 \leq (x_i - j) \leq N)$ として展開される．

ステップ 3

If $\sum_{m,x,y} A_{mxy} \neq 0$,

Then

If $\sum_m A_{mxy}$ が最小になり，

かつ $\sum_m A_{mxy} \neq 0$ を満たす (x,y) を選択し，

Then

If $x + N(y-1)$ が最小になる (x,y) を選択し，

Then $(x_p, y_p) = (x,y)$.

Else **ステップ 7 へ**.

ステップ 4

If $\sum_m A_{m x_p y_p} = 1$,

Then

If $A_{m x_p y_p} = 1$,

Then $m_p = m$.

Else

If O_m が最小になり，かつ $A_{m x_p y_p} = 1$ を満たす m を選択し，

Then

If m が最小になる m を選択し，

Then $m_p = m$.

ステップ 5

m_p を (x_p, y_p) に配置し，

$A_{m x_p y_p} = A_{m_p xy} = 0$ $(m = 1,2,\ldots,N^2 : x,y = 1,2,\ldots,N)$ とする．

ステップ 6

ステップ3へ.

ステップ7

If $k = 1$ (k:未配置モジュール数),

Then 未配置モジュールを空いている配置区画に配置する.

Else

If $k \geq 2$,

Then $N_p = k - 1$

(N_p:最終未配置モジュール数).

例として，3×3の配置領域に9個のモジュールを配置する場合を考える．染色体は次に示すものとする.

(1,3,2) (3,3,2) (2,1,1) (2,3,6) (3,3,4)
(2,2,5) (1,1,1) (1,2,7) (1,3,4)

これを展開すると次のようになる.

	(x,y)におけるA_{mxy}									
m	(1,1)	(2,1)	(3,1)	(1,2)	(2,2)	(3,2)	(1,3)	(2,3)	(3,3)	O_m
1	0	0	0	1	0	0	1	1	0	2
2	0	0	0	0	0	1	0	1	1	2
3	1	1	1	0	1	0	0	0	0	1
4	0	0	0	0	1	0	1	1	1	6
5	0	0	0	0	0	1	0	1	1	4
6	0	1	0	1	1	1	0	1	0	5
7	1	1	0	1	0	0	0	0	0	1
8	1	0	0	1	1	0	1	0	0	7
9	0	0	0	1	0	0	1	1	0	4
$\sum_m A_{mxy}$	3	3	1	5	4	3	4	6	3	
配置結果										

ここからステップ3により$(x_p, y_p) = (3,1)$となり，ステップ4で$\sum_m A_{mx_py_p} = 1$を満たすため，m_pは$A_{mx_py_p} = 1$を満たす3となる．そしてステップ5で配置とA_{mxy}の変更が行われ，次のようになる.

	(x,y)におけるA_{mxy}									
m	$(1,1)$	$(2,1)$	$(3,1)$	$(1,2)$	$(2,2)$	$(3,2)$	$(1,3)$	$(2,3)$	$(3,3)$	O_m
1	0	0	0	1	0	0	1	1	0	2
2	0	0	0	0	0	1	0	1	1	2
3	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	1	0	1	1	1	6
5	0	0	0	0	0	1	0	1	1	4
6	0	1	0	1	1	1	0	1	0	5
7	1	1	0	1	0	0	0	0	0	1
8	1	0	0	1	1	0	1	0	0	7
9	0	0	0	1	0	0	1	1	0	4
$\sum_m A_{mxy}$	2	2	0	5	3	3	4	6	3	
配置結果			3							

この場合ステップ3で$\min \sum_m A_{mxy}$ (ただし $\sum_m A_{mxy} \neq 0$)を満たす座標は$(1,1)$と$(2,1)$の2つがある．そこで$x+N(y-1)$の値から$(x_p, y_p) = (1,1)$となる．ステップ4の$\sum_m A_{mx_py_p} = 1$の条件は満たさないので次へ進む．O_mの値を比較すると，$O_7 < O_8$なので$m_p = 7$となる．ステップ5で配置とA_{mxy}の変更が行われ，次のようになる．

	(x,y)におけるA_{mxy}									
m	$(1,1)$	$(2,1)$	$(3,1)$	$(1,2)$	$(2,2)$	$(3,2)$	$(1,3)$	$(2,3)$	$(3,3)$	O_m
1	0	0	0	1	0	0	1	1	0	2
2	0	0	0	0	0	1	0	1	1	2
3	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	1	0	1	1	1	6
5	0	0	0	0	0	1	0	1	1	4
6	0	1	0	1	1	1	0	1	0	5
7	0	0	0	0	0	0	0	0	0	1
8	0	0	0	1	1	0	1	0	0	7
9	0	0	0	1	0	0	1	1	0	4
$\sum_m A_{mxy}$	0	1	0	4	3	3	4	6	3	
配置結果	7		3							

以下同様に繰り返すと，最終的に次のようになる．

	(x,y)におけるA_{mxy}									
m	$(1,1)$	$(2,1)$	$(3,1)$	$(1,2)$	$(2,2)$	$(3,2)$	$(1,3)$	$(2,3)$	$(3,3)$	O_m
1	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	0	2
3	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	6
5	0	0	0	0	0	0	0	0	0	4
6	0	0	0	0	0	0	0	0	0	5
7	0	0	0	0	0	0	0	0	0	1
8	0	0	0	0	0	0	0	0	0	7
9	0	0	0	0	0	0	0	0	0	4
$\sum_m A_{mxy}$	0	0	0	0	0	0	0	0	0	
配置結果	7	6	3	9	4	2	8	1	5	

このときステップ3の，すべてのmに対して$\sum_{x,y} A_{mxy} = 0$となったらステップ7にという処理が選択され，配置処理が終了する．なお，この場合配置不可能だったモジュール数N_pは0である．

このように情報に冗長性を持たせるコーディング法を用いれば，配置位置に重複を生じることなく配置可能になるケースが多くなる．したがって，交叉や突然変異によって配置不可能な解となることが減少し，個体集団の多様性は保たれ局所解に陥ることなく最適解の探索が可能になると予想される．

3.7　計算機実験

提案したコーディング法の有用性を確認するため計算機実験を行なった．この節では実験における幾つかの条件および，それらの条件下での実験結果について述べる．

3.7.1　実験条件

本実験では3.4節で述べた基本的な配置問題を対象とする．また，GAの諸設計についても基本的なものを採用し，次のように定める．

初期集団の生成：決められた個体数の染色体M個をランダムに生成する．

適応度の評価：適応度fは，仮想配線長Dとモジュール間の総接続本数C，さらに配置不可能だったモジュール数N_pから

$$f = \frac{1}{1 + \frac{D}{C} + \alpha N_p}$$

として定義する．なお，αはN_pをペナルティとするための重みである．

選択：適応度比例戦略とエリート保存戦略を用いる．なお，このときのエリート保存数は$\max\{0.01M, 3\}$とする．

交叉：２点交叉を用いる．

突然変異：確率p_mで選ばれた或る個体の，ランダムに選ばれた或る1つの遺伝子座の遺伝子のみを変異させる1点突然変異を用いる．

この設計のもとコーディング法のみを，“GAを単純に適応した場合”と“リストによる従来のコーディング法の場合”さらに提案した“否定情報を持たせたコーディング法の場合”及び“情報に冗長性を持たせたコーディング法の場合”で変え比較実験を行った．

3.7.2 実験結果と考察

実験におけるGAのパラメータは表3.5に示す値を用いた．実験モデルとしては

表 3.5 計算機実験パラメータ

個体数	M	1000
世代数	T	500
交叉率	p_c	0.6
突然変異率	p_m	0.03

3.4節で述べた配置問題において$N = 4$のモデルを用い，接続関係は図3.12に示すように格子状に接続されたものとする．したがって，仮想配線長の理論最短値は24になり，その時の適応度は

$$f = \frac{1}{1 + \frac{24}{24} + 7.5 \times 0} = 0.5$$

となる．なお，このモデルではαは7.5として適応度を求めた．また“否定情報を持たせたコーディング法の場合”において染色体の長さC_lは経験的に36とした．

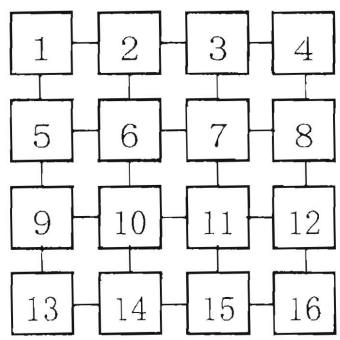


図 3.12 実験モデルの接続関係

実験結果を表3.6に示す．実験はそれぞれのコーディング法に対し５回行った．また，$f_{\max}$はその世代の最大適応度，Dは最大適応度のときの仮想配線長を表す．なお，表3.6に示す結果は最終結果であり，D，$f_{\max}$は500世代目の値である．計算時間はHP-9000/755ワークステーションで要した時間である．

表 3.6 計算機実験結果

	単純		リスト		否定情報		冗長情報	
	D_s	$f_{\max}$	D_l	$f_{\max}$	D_n	$f_{\max}$	D_p	$f_{\max}$
1st	63.0	0.276	36.0	0.400	35.0	0.407	24.0	0.500
2nd	54.0	0.308	36.0	0.400	34.0	0.414	24.0	0.500
3rd	52.0	0.316	34.0	0.414	33.0	0.421	33.0	0.421
4th	61.0	0.282	33.0	0.421	36.0	0.400	24.0	0.500
5th	55.0	0.304	24.0	0.500	32.0	0.429	32.0	0.429
平均	57.0	0.298	32.6	0.427	34.0	0.414	27.4	0.470
平均計算時間(s)	299.6		237.9		2275.5		945.3	

“否定情報を持たせたコーディング法の場合” 提案手法の仮想配線長 D_n は平均値で比較して D_s を下回ったが D_l の値を上回っており，その減少率（%）は D_s に対しては

$$\frac{D_s - D_n}{D_s} \times 100 = 40.3\ (\%)$$

D_l に対しては

$$\frac{D_l - D_n}{D_l} \times 100 = -4.3\ (\%)$$

となり，提案手法の探索能力は“リストによる従来のコーディング法”とほぼ同等という結果になった．また，5回の実験において“提案手法”では最適解（仮想配線長24）を得ることができなかった．提案手法は，デコーディング手順の煩雑化にともなう計算機時間の増大も激しく，また染色体長が増加するため情報の継承も難しくなっている．したがって，配置不能な解の生成を押さえることは可能だが，提案手法を実用に用いるのは難しいと考えられる．

“情報に冗長性を持たせたコーディング法の場合” 提案手法の仮想配線長 D_p は平均値で比較して D_s や D_l の値を下回っており，その減少率（%）は D_s に対しては

$$\frac{D_s - D_p}{D_s} \times 100 = 51.9\ (\%)$$

D_l に対しては

$$\frac{D_l - D_p}{D_l} \times 100 = 16.0\ (\%)$$

となり，提案手法の方が従来の手法よりも優れていることが確認できた．また，5回の実験において“単純な手法”では最適解（仮想配線長24）を得ることができなかったが，“リストによる手法”と“提案手法”では最適解を得ることができた．しかし，“リストによる手法”においては5回中1回だけしか最適解を得ることができず，ランダムサーチ的な振る舞いをしていることに原因すると考える．“提案手法”において3回最適解を得ることができたのは，ランダムサーチ的な振る舞いが抑制されたものと思われる．

3.8　むすび

この章ではGAを用いた配置手法の研究として，基本的な配置問題へのGAの適用を試みた．しかし，単純にGAを適用したのでは探索効率が低く局所解に陥りやすい．そこで本研究では探索効率の低下を回避する，次の２つの新しいコーディング法を提案した．

1. 配置位置を否定する情報を持たせたコーディング
2. 配置位置情報に冗長性を持たせたコーディング

提案手法の有用性を確認するため，提案手法と従来手法を比較する計算機実験を行った．“否定情報を持たせたコーディング法の場合”は配置不能な解の生成こそは抑制可能だが，実用に足る結果を得ることはできなかった．“情報に冗長性を持たせたコーディング法の場合”は従来手法の問題点であった配置不能な解の生成と，ランダムサーチ的な振る舞いを共に抑制することにより，従来よりも優れた探索結果を得ることが可能なことが確認された．

第 4 章

任意形状ブロックの配置手法

4.1 まえがき

VLSI レイアウト手法の多くにおいては，配置対象ブロックの形状は矩形に限定され矩形以外のブロックを扱うことは困難である．また矩形以外のブロックを対象とする手法においても，配置対象ブロックの形状自由度は高くない．そこで本章では，従来手法では難しかった凹部を含むブロックの配置や，配置領域を制限しての配置を可能にする，任意形状ブロックの配置手法を提案する．提案手法は，ブロックの形状に応じた評価値を定義し，その値を用いることで任意形状ブロックを配置する組立式配置手法と，その配置順序に遺伝的アルゴリズムによる制御を組み合わせたものである．また最後に，計算機実験により本手法の有用性を示す．

4.2 研究の目的

多くのVLSI レイアウト手法においては，配置対象ブロックの形状は矩形に限定され，矩形以外のブロックを扱う事は困難である．また矩形以外のブロックを対象とする手法においても，そのブロック形状はＬ型[10]や凸XY 多角形と呼ばれる形状[11]などに限定され，配置対象ブロックの形状自由度は決して高くなく，凹部を含むブロックの配置などには対応できない．また，ブロック形状に制限がない手法[12]も提案されている．しかし，文献[11]の手法も含めコンパクション手法であるため，配置領域を限定して配置を行うことは困難という問題がある．また，これらコンパクション手法を用いるには何らかの初期配置が必要になるが，

任意形状ブロックに適した初期配置手法は提案されていない．

そこで本研究ではこれらの問題に対し，形状に応じた評価値を用いることで凹部を含むブロックの配置や，配置領域を制限しての配置も可能とする任意形状ブロックの配置手法を提案する [24, 30, 33]．本手法はVLSIレイアウトにおける組立式配置手法の一種であるが，その配置順序に遺伝的アルゴリズムによる制御を組み合わせることで，組立式手法に共通する欠点である配置結果の順序依存問題にも対応可能である．また，本手法は既に提案されているコンパクション手法の初期配置などにも有効である．

4.3 任意形状ブロックの組立式配置手法

本節ではブロック形状に応じた評価値を定義し，その値を用いて任意形状ブロックを配置する組立式配置手法を提案する．本手法では配置領域に制限を設けての配置を，配置領域に制限がないときの特殊な場合として捉え配置を行う．したがって，ここでは基本となる配置領域に制限がない場合の提案手法から述べる．なお，本手法では組立式手法において重要な配置順序の制御を，遺伝的アルゴリズムにより行い良好な配置結果を得ているが，遺伝的アルゴリズムによる制御に関しては後述する．

4.3.1 配置領域に制限がない場合の手法

配置領域に制限がない場合の任意形状ブロックの配置について考える．ここでは初めに，任意形状ブロックとブロック形状に応じた評価値を定義する．次に，定義した評価値を使いブロックを配置する手順について述べ，さらに仮想配線長を用いて配置手順の拡張を行う．

4.3.1.1 任意形状ブロックと隣接度

本手法で対象とする任意形状ブロックとは，図4.1に示すようないくつかのセル（１つのセルは図に斜線で示したもの）の塊である．また，レイアウト領域は碁盤の目のように正方形の区画が縦横に連なったものとし，１つのセルは１つの区画を占めるとする．

本手法では，レイアウト領域の各区画に対してブロック形状に応じた評価値を定める．この評価値は，各区画の周りの８つの区画にいくつセルがあるかを数え

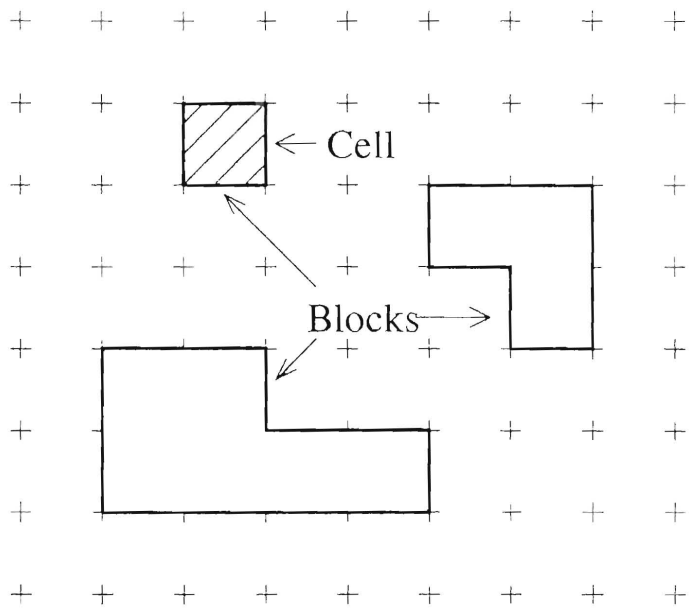


図 4.1 任意形状ブロックの例

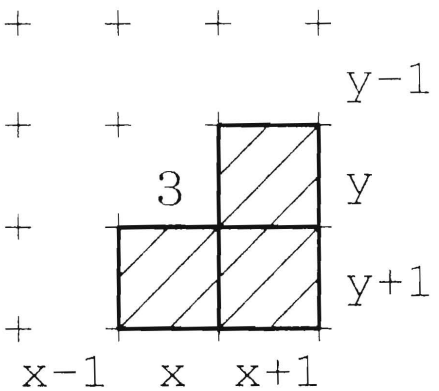


図 4.2 隣接度の例

たものであり，その区画にセルを置いたとき既に配置されているセルと隣接する度合を示している．したがって，以下ではこの評価値を「隣接度」とよぶこととする．すなわち座標（x,y）の区画の隣接度 R_{xy} は次式で与えられる．

$$R_{xy} = \sum_{i=-1}^{1} \sum_{j=-1}^{1} S_{x+i,y+j} \tag{4.1}$$

但し　S_{xy} :座標 x,y の区画に

セルが存在するとき　1

〃　　しないとき　0

なお，隣接度は既にセルが配置されている区画については定義しない．

例として図4.2の場合，中央の区画（座標（x,y））の隣接度は式（4.1）において，座標（x,y+1）と（x+1,y）と（x+1,y+1）の区画にセルが存在するので

$$\begin{aligned} R_{xy} &= 0+0+0+0+0+1+0+1+1 \\ &= 3 \end{aligned}$$

となり隣接度は３である．同様にすべての区画の隣接度を求めることができる．

4.3.1.2　隣接度による配置

前述したように隣接度はその区画にセルを配置したとき，既に配置されているセルと隣接する度合を示している．そこで，ブロックを配置するときは配置した区画の隣接度の和が最大になる区画に配置すると効率がよく，チップ面積の最少化にもつながる．したがって，ブロック配置の手順は以下のようになる．

ステップ１： レイアウト領域の各区画に対して式（4.1）に従い隣接度を求める．

ステップ２： ブロックを配置したとき，隣接度の和が最大になる区画を選ぶ．なお，このとき既にセルの存在する区画を選ぶことは許されない．

ステップ３： ステップ２で選んだ区画にブロックを配置する．

ステップ４： 配置するブロックがなくなるまでステップ１，２，３を繰り返す．

例として図4.3-aのようにセルが既に配置されているときに，図4.3-bのブロックを配置する場合を考える．ステップ１に従い隣接度を求めると，その分布は図4.3-cのようになる（図に示した以外の区画の隣接度は０である）．この隣接度分布から考えてステップ２で選ばれる区画は，隣接度の和が８で図4.3-dの斜線の部分になる．したがって，ステップ３で配置した結果は図4.3-eとなる．ここでまだ配置するブロックが残っている場合は，図4.3-eに示したセル配置に対し隣接度分

布を求め（つまり，ステップ 1 に戻る），同様に繰り返す．以上が今回提案する組立式配置手法の基本である．

なお，既に述べたように本手法は配置対象である任意形状ブロックを，正方形のセルが縦横に連なったものとしている．したがって，ブロック各辺の長さはセルの長さの整数倍に制限され，この点での自由度は決して高いとはいえない．もちろんセルを十分に小さく考えれば，理論的にはどのような精度でもブロックを表現できるが，その場合，精度に応じた処理時間の増大を避けることはできない．そこで，長さについて高い精度を必要とするときは実用性を考慮して，次のような方法で本手法を用いることを提案する．まず，ブロックの各辺の長さを実用上問題のない程度に近似し，本手法を用いて配置する．その上で，近似したブロックを本来のブロックに置き換える．すると，ブロック同士の重なりや，あるいは無駄領域を生じることになるが，それらを解消するコンパクション手法[12]は既に提案されている．そこで，この従来手法を用いてコンパクションをかける．つまり，本手法を使って従来のコンパクション手法の初期配置を与えることで，長さについて十分な精度を持ち，なおかつ実用的な処理時間での配置が可能となる．

4.3.1.3 仮想配線長による拡張

隣接度は形状に応じた評価値であるため，形状が対称の場合，隣接度の分布も対称となり，ブロックを配置したときに隣接度の和が最大になる区画も複数になる．そこで配置区画候補が複数になった時の選択法として，他の評価法との併用が必要となる．本手法では仮想配線長を用いることにし，これによりブロック配置の手順は次のように拡張される．

ステップ 1： レイアウト領域の各区画に対して式（4.1）に従い隣接度を求める．

ステップ 2： ブロックを配置する区画を選ぶ．なお，このとき既にセルの存在する区画を選ぶことは許されない．

ステップ 2．1： ブロックを配置したとき，隣接度の和が最大になる区画を選ぶ．

ステップ 2．2： ステップ 2．1 で選ばれた区画が複数存在するときは，その中から配置したとき仮想配線長が最短になる区画を選ぶ．

ステップ 2．3： ステップ 2．2 で選ばれた区画も複数存在するときは，それらの区画は等価として，探索時に先に探索された区画を選ぶ．

ステップ 3： ステップ 2 で選んだ区画にブロックを配置する．

ステップ 4： 配置するブロックがなくなるまでステップ 1，2，3 を繰り返す．

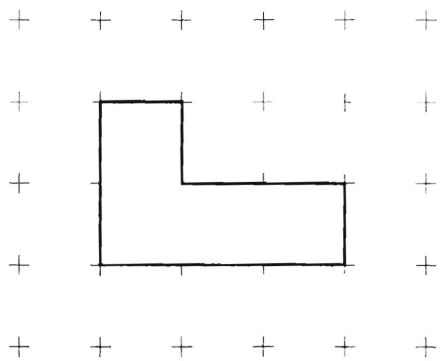


図 4.3-a　配置済みブロック

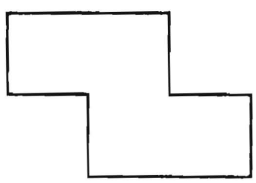


図 4.3-b　次に配置するブロック

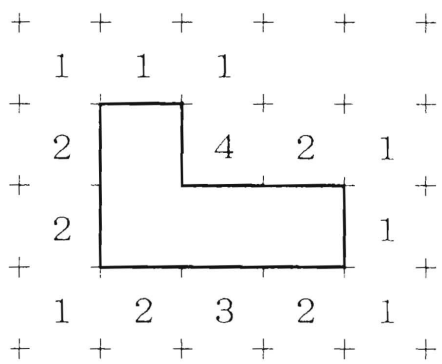


図 4.3-c　隣接度の分布

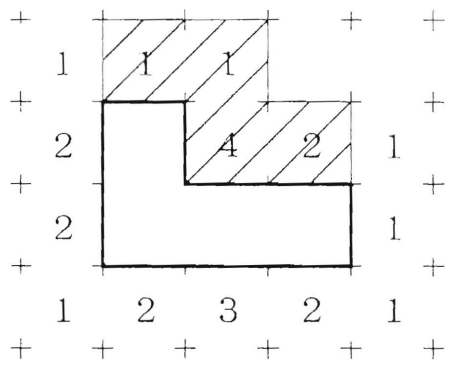


図 4.3-d　隣接度の和が最大になる区画

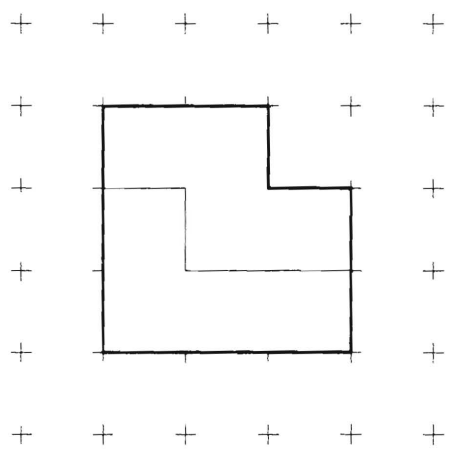


図 4.3-e　配置結果

図 4.3　隣接度による配置の例

ステップ２．２での仮想配線長は，各ネットごとにそれを構成する端子を内包する最小矩形を考え，その周囲長の1/2をそのネットの仮想配線長とするものを採用する．なお本手法の場合，配置過程で仮想配線長を算出する必要があるため，ネットを構成する全ての端子が配置済みとは限らない．そこで，配置過程では配置されていない端子は無視し，配置済みの端子のみで仮想配線長を算出する．すなわち，ネットn_iに含まれる配置済み端子のリストを$P_i = (p_j \cdots p_k)$とし，それらの端子のx座標を$p_{j,x}$，y座標を$p_{j,y}$，とするとネットn_iの仮想配線長$n_{i,length}$は

$$n_{i,length} = \max_{P_i}(|p_{j,x} - p_{k,x}|) + \max_{P_i}(|p_{j,y} - p_{k,y}|)$$

となり，したがって総ネット数をCとすると，総仮想配線長Dは

$$D = \sum_{i=1}^{C} n_{i,length}$$

となる．また，上記手順ではステップ２．３が定められており，探索順という明確な根拠のない選択がなされている．しかし，本手法では仮想配線長の算出が端子位置を基づいているため，実際的なブロックではステップ２．２で仮想配線長が最短になる区画が，複数存在することは極めてまれになる．なお，後述する計算機実験の最終配置結果においても，ステップ２．３の利用はなされていない．ただし，配置済みブロックと配置しようとしているブロックの間にネットが存在しない場合にはステップ２．２での特定ができないため，ステップ２．３が適用される．しかし，こういった場合は，複数ある配置区画候補のいずれに配置しても等価であるとして，本手法では探索順による配置を採用した．

4.3.2 配置領域に制限がある場合の手法

次に，配置領域に制限がある場合について考える．既に述べた配置領域に制限がない場合においては，レイアウト領域は無限の広がりを持つものとして扱ってきた．そこで配置領域に制限がある場合においても同様に，レイアウト領域は無限の広がりを持つものとする．そして配置領域制限は，レイアウト領域の配置可能区画以外はすべてセルで埋め尽くされている，とすることで実現する．これにより配置領域に制限がある場合においても同様に，隣接度を求めその和が最大になる区画に配置するという手法を用いることができる．ただし，この場合，配置可能区画に制限が存在するため，配置対象となるブロックが全て配置可能とは限らない．そこで，配置手順は次のように修正される．

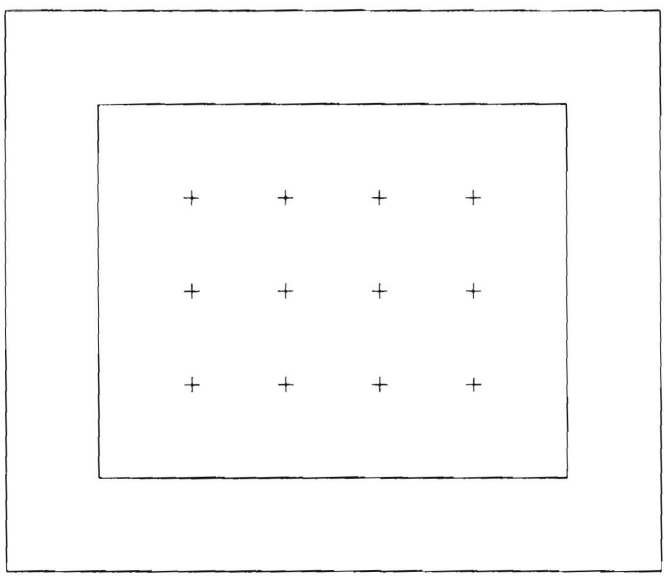


図 4.4-a　配置領域制限の例

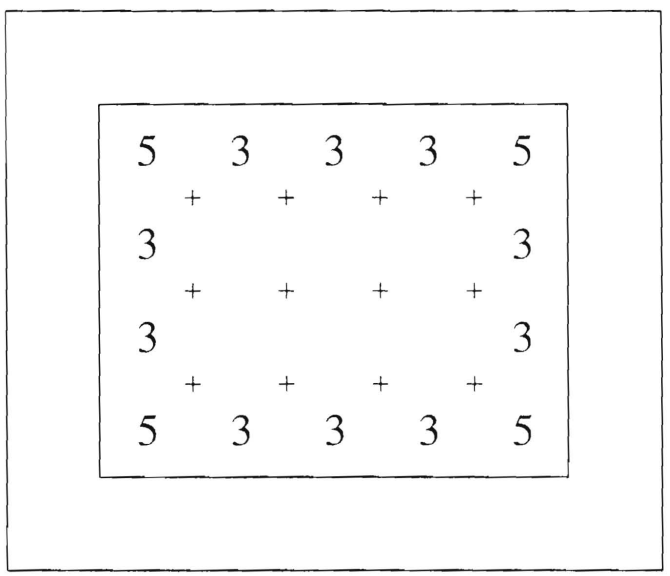


図 4.4-b　配置領域制限の隣接度の例

図 4.4　領域制限があるときのレイアウト領域

ステップ1： レイアウト領域の各区画に対して式（4.1）に従い隣接度を求める．

ステップ2： ブロックを配置する区画を選ぶ．なお，このとき既にセルの存在する区画を選ぶことは許されない．また，既に十分な配置可能区画が存在しないときは，未配置のブロックがあっても配置手順を終了する．

ステップ2．1： ブロックを配置したとき，隣接度の和が最大になる区画を選ぶ．

ステップ2．2： ステップ２．１で選ばれた区画が複数存在するときは，その中から配置したとき仮想配線長が最短になる区画を選ぶ．

ステップ2．3： ステップ２．２で選ばれた区画も複数存在するときは，それらの区画は等価として，探索時に先に探索された区画を選ぶ．

ステップ3： ステップ２で選んだ区画にブロックを配置する．

ステップ4： 配置するブロックがなくなるまでステップ１，２，３を繰り返す．

例として４行５列の配置領域制限がある場合について述べる．このときレイアウト領域は図4.4-aのように４×５の配置可能区画を残してセルで埋め尽くされているとして考える（図に示した以外の区画は全てセルで埋め尽くされている）．これに対し隣接度を式（4.1）に従い求めると，図4.4-bのような分布が得られる(図に示した以外の区画の隣接度は０である)．後は配置領域に制限がない場合と同様に，既に述べたブロック配置の手順に従いブロックを配置する．なお，この例では配置領域制限を矩形としたが，本手法としては領域制限が矩形である必要はない．たとえ領域制限が非矩形であっても，隣接度を求め配置することは可能である．

また，配置領域制限内に配置対象のブロックが収まるかどうかは重要な問題ではあるが，今のところそれを事前に知る方法は知られていない．もちろん，領域制限の面積よりもブロックの総面積が大きい，言い換えるならば配置可能区画の数よりも，ブロックの総セル数が多い場合は収まらないのは明らかである．しかし，面積の大小関係に問題がないからといって必ずしも収まるとは限らない．本手法において配置領域制限内に配置対象のブロックが収まるかどうかは，一般的な「ポリオミノの敷き詰め」問題と同様な問題といえるが，この問題において可能を証明する手段がないように，本手法も収まることを証明する手段はない．ポリオミノの敷き詰めと同様に，パリティを使った方法で不可能を証明[14]することはできるが，それは可能を証明したことにはならない．すなわち，収まらないと証明されなかったとしても収まるとは限らない．あくまでも収まる可能性があるだけであり，実際に収まるかどうかは別の問題である．以上のような理由から，配置領域制限内に全ブロックが収まるかどうかを事前に知ることは現状では困難

である．

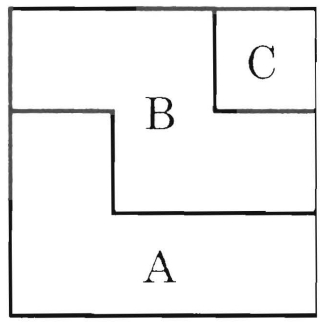


図 4.5-a　ＡＢＣの順に配置

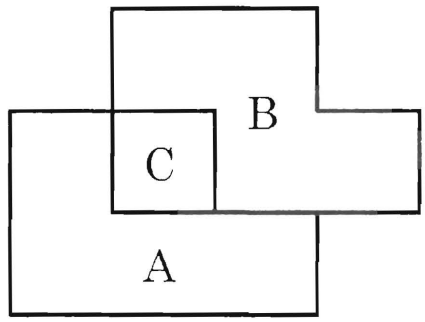


図 4.5-b　ＡＣＢの順に配置

図 4.5　配置順序依存の例

4.4　遺伝的アルゴリズムによる配置順序制御

本手法では配置結果は配置順序に依存する．これは本手法が組立式手法の一種であるため，配置処理過程では全体的な見通しがなされないことに関係する．すなわち，図4.5のようなA, B, Cの３つのブロックを配置するとき，A, B, Cの順で配置すると図4.5-aのような結果が得られ，A, C, Bの順で配置すると図4.5-bのような結果になる．この場合，最終的結果としては図4.5-aの配置結果が形状としては優れているが，これは手法としてこの配置を目指していたわけではなく配置順序から偶然に得られた結果にすぎない．すなわち配置結果は配置順序に強く依存するため，本手法では配置順序に適当な制御を必要とする．そこで本研究で

は遺伝的アルゴリズム（GA）を用いて，本手法に最適な配置順序を求めることとする．GAは生物が遺伝と進化により環境に適応することに範をとった最適解探索アルゴリズムの一つであり，広範囲にわたる探索に威力を発揮することで知られている．しかし，GAを用いて順列組合せ問題を解く場合には，コーディングやオペレータに何らかの工夫をしなければ満足のいく解は得られない．そこで本手法では，次のようなコーディングの工夫により問題を解決する．

ブロックの配置順序を$T=(t_1\cdots t_N)$とし，またブロックを適当な順序で並べたリストW（定数）を用意する．そして，i番目に配置するブロックを，未配置ブロックリスト$W-\{t_1,\cdots,t_{i-1}\}$の何番目であるかを表わし，これを遺伝子l_iとし，リスト$L=(l_1\cdots l_N)$を染色体とする．このコーディングにより，遺伝子型と表現型が一対一に対応づけられ，通常のオペレータの適用が可能になる．なお，このコーディングはGrefenstetteらによる巡回セールスマン問題のための方法[19]を，ブロックの配置順序に応用したものである．

例としてブロックの配置順序が$T_1=(B\,D\,A\,C\,E)$のときのコーディングを考える．ブロックを適当な順序で並べたリストを$W=(A\,B\,C\,D\,E)$とすると，1番目に配置するブロックBは，未配置ブロックリストWの2番目なので遺伝子l_1は2となる．次に2番目に配置するブロックDは，未配置ブロックリスト$W-\{B\}=(A\,C\,D\,E)$の3番目なので遺伝子l_2は3となる．以下これを繰り返すと染色体はリスト$L_1=(2\,3\,1\,1\,1)$として求まる．また，ブロックの配置順序が$T_2=(E\,D\,C\,B\,A)$のときの染色体は，リスト$L_2=(5\,4\,3\,2\,1)$となる．GAでは，これらの染色体を交叉することで新しい染色体を作り探索を行うが，ここでは上記のコーディングの工夫により通常の交叉を用いての探索が可能になる．すなわち，上記の2つの染色体の例で考えると染色体はそれぞれ，

$L_1=(2\,3\,1\,1\,1)$

$L_2=(5\,4\,3\,2\,1)$

である．これを2番目の遺伝子と3番目の遺伝子の間で交叉したとすると，新しい染色体は

$L_1'=(2\,\underline{3}\,3\,2\,1)$

$L_2'=(5\,4\,\underline{1}\,1\,1)$

となる．これをコーディングと逆の手順でブロックの配置順序に戻すと，

$T_1'=(B\,D\,E\,C\,A)$

$T_2'=(E\,D\,A\,B\,C)$

となり，ブロックの重複を含まない配置順序，すなわち順列という条件を満足す

る解が得られ，効率の良い探索が可能になる．なお，突然変異は遺伝子l_iを，条件$1 \leq l_i \leq N-i+1$を満足する遺伝子にランダムに置き換えることで実現する．

GAを用いる上での評価値である適応度fは，仮想配線長Dと総ネット数C，さらに配置結果を内包する最小矩形を考えたときに，セルが配置されていない区画の数Eから

$$f = \frac{1}{1+\alpha\frac{D}{C}+\beta E} \tag{4.2}$$

として定義する．すなわち，仮想配線長が短く，かつ配置結果が隙間なくコンパクトにまとまっている方が評価が高いとする．また，配置領域に制限がある場合では，最終的に未配置のブロックが存在することが考えられるが，この定義を用いればセルが配置されていない区画の数Eが増えるため評価が低くなり，世代を重ねることで淘汰されると考えられる．なお，αとβは，仮想配線長と配置結果の隙間という二つの評価要素の，適応度fに占める割合を決定する係数であり，配置の目的に応じて設定する．

GAによる配置順序制御の導入により，本配置手法全体は次のようになる．

ステップ1： 初期集団としてM個の個体を生成する．このとき各個体の持つ情報は，上記のコーディングによりブロックの配置順序を表わしている．

ステップ2： 各個体に対し次の処理を行う．

ステップ2．1： 個体の持つ配置順序情報に従い，前述のブロック配置の手順を使って配置可能な全てのブロックを配置する．このとき配置結果は配置順序により一意に決定される．

ステップ2．2： 配置結果から仮想配線長Dと空き区画数Eを求め，式（4.2）に従い適応度fを算出する．

ステップ3： ステップ２で求めた適応度に従い，選択，交叉，突然変異などのオペレータにより個体集団を進化させる．

ステップ4： ステップ２，３を一世代とし定められた世代数を繰り返す．

ステップ5： 最終的に得られた最も優れた適応度を持つ個体の配置結果を，本手法の配置結果とする．

4.5　計算機実験

本手法の有用性を確認するため計算機実験を行った．遺伝的アルゴリズムのオペレータとパラメータは経験的に次のように決定した．これは以下の実験において共通である．

適応度：適応度 f は式（4.2）において，仮想配線長による評価と配置結果の隙間による評価を適度に取り入れることを目的に，$\alpha = \beta = 1$ とする．

選択：適応度比例戦略とエリート保存戦略を用いる．なお，このときのエリート保存数は 1 とする．

交叉：一点交叉を用いて，交叉確率 p_c は 0.95 とする．

突然変異：突然変異確率 p_m は 0.1 とする．

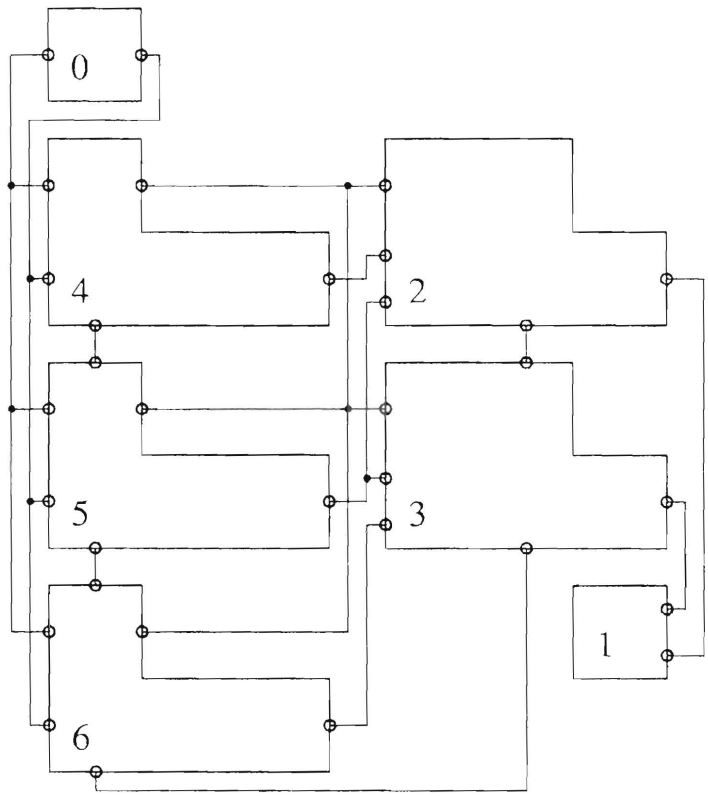


図 4.6　配置対象ブロックの例 1

まず，図4.6のブロックを配置する実験例を示す．このとき個対数 M を 30，世代数 T を 20 とした．配置領域に制限を設けないときの配置結果を図4.7に示す．なお，配置順序はブロック番号 5，3，2，6，4，1，0 の順で，仮想配線長はセルの一辺を 1 として 46.5 である．配置結果を内包する最小矩形を考えると 5 × 5 の区画になり，その中で左上角の区画が空いているので空き区画数は 1 となる．また，配置領域に制限がある場合についても計算機実験を行った．ここでも例として図4.6に示したブロックを配置することを考える．図4.6に示したブロックの総セル

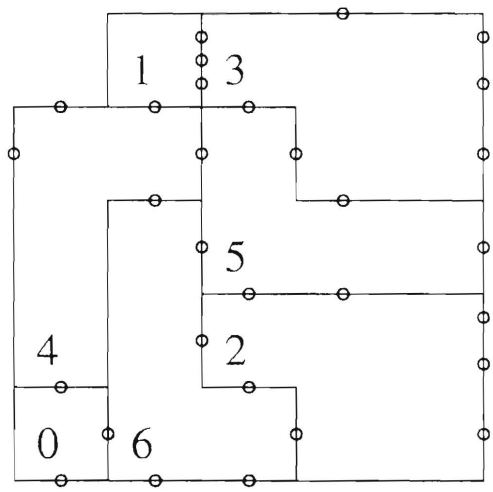


図 4.7　領域制限がないときの配置（例 1）

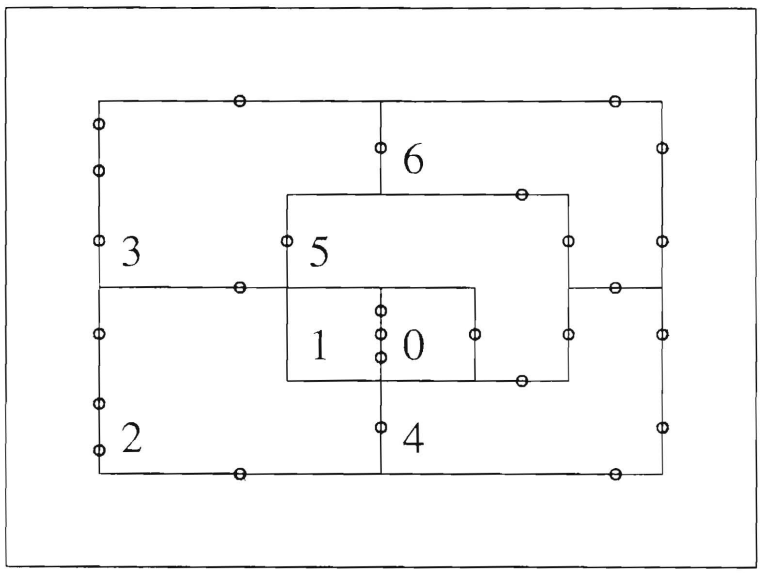


図 4.8　領域制限があるときの配置（例 1）

数は24，そこで4行6列の配置領域制限を設けることとした．結果は図4.8に示すように，与えられた配置領域内に全てのブロックが配置された．さらに本手法においては，配置領域制限は必ずしも矩形である必要はない．そこで図4.6に示すブロックを図4.9のような非矩形の配置領域へと配置した．結果は図4.10に示すように与えられた配置領域内に全てのブロックを配置することができた．

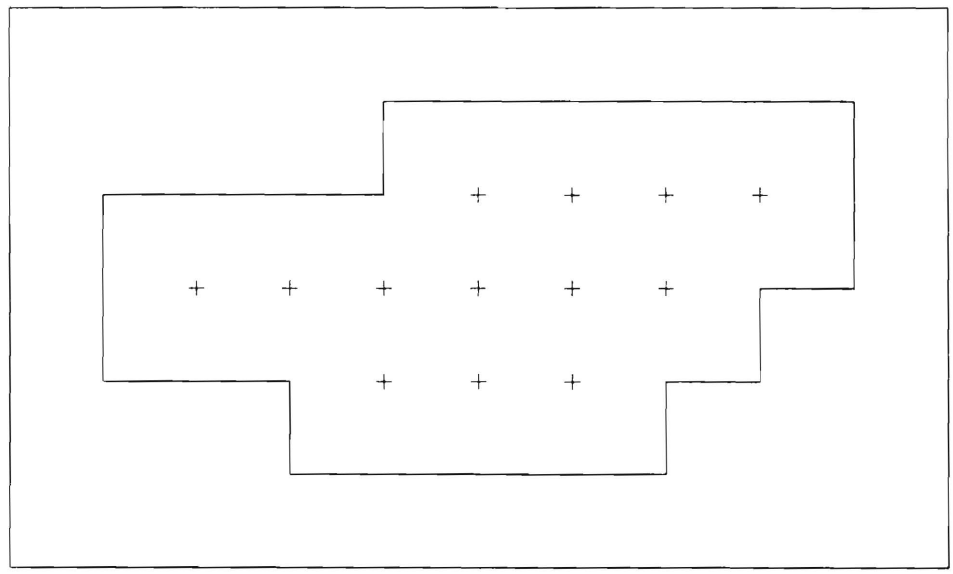


図 4.9　非矩形の領域制限

次に50個のブロックを配置する実験例を示す．このとき個対数 M を100，世代数 T を50とした．この例では総セル数は240，そこで15行16列の配置領域制限を設ける．その結果，図4.11に示したように，50個のブロック全てを隙間なく配置することができた．なお，この例では凹部を持つブロックが10個存在するが，その全てが無駄領域を生ずることなく配置された．また，非矩形の配置領域への配置も行い，図4.12に示したように全てのブロックが配置されることを確認した．

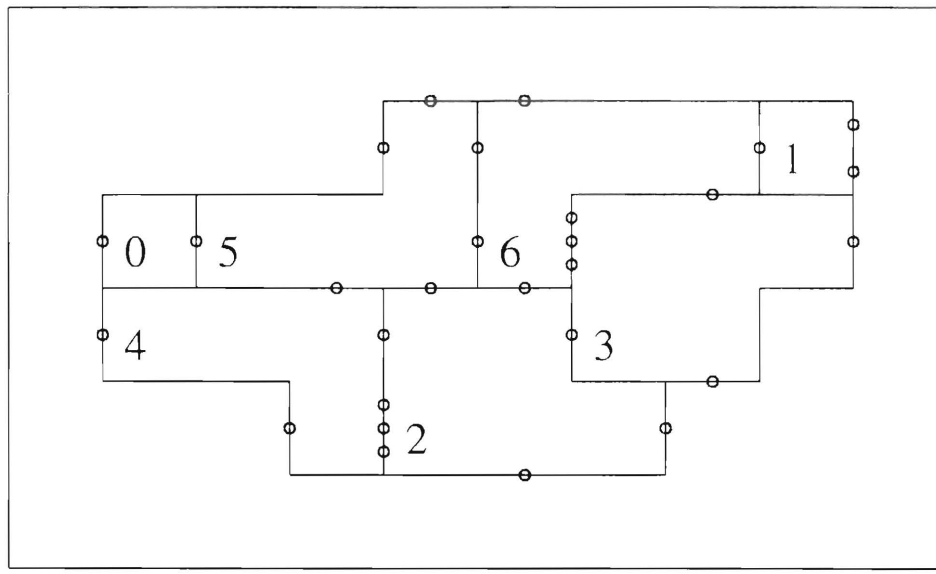


図 4.10　非矩形の領域制限があるときの配置（例 1）

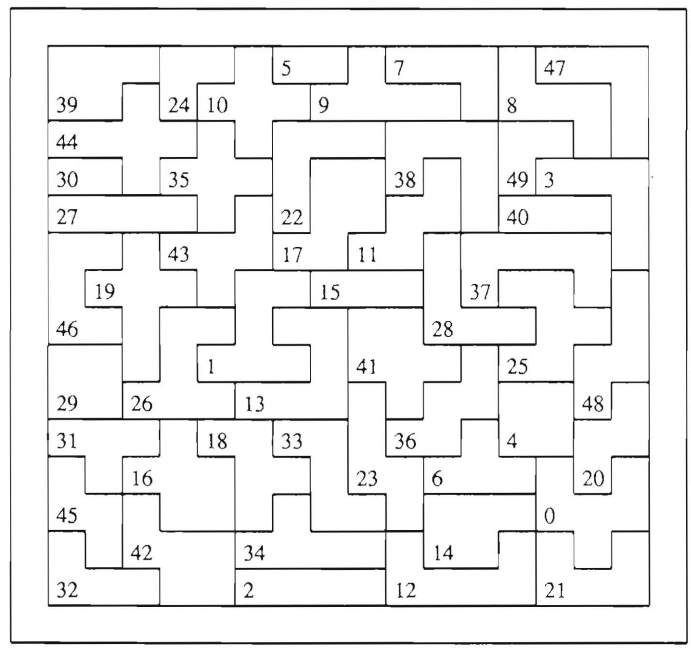


図 4.11　領域制限があるときの配置（例 2）

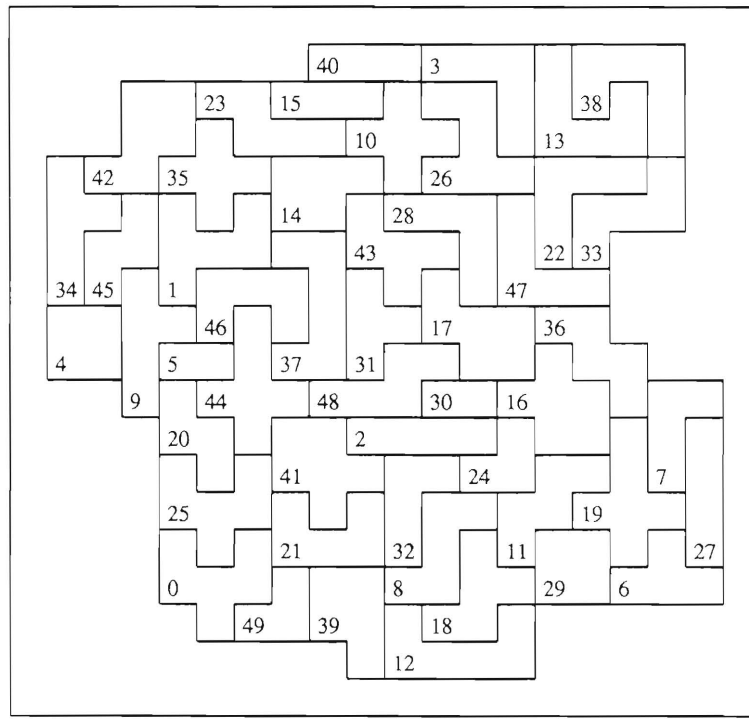


図 4.12　非矩形の領域制限があるときの配置（例２）

4.6　むすび

この章ではVLSIレイアウトにおける組立式配置手法として，ブロック形状に応じた評価値を定義することにより，任意形状ブロックの配置を行う一手法を提案した．本手法の特長は以下の通りである．

1. 従来手法では難しかった凹部を含む，任意形状ブロックの配置が可能である．
2. 非矩形の領域制限を含む，配置領域を制限しての配置が可能である．
3. 配置順序に遺伝的アルゴリズムによる制御を組み合わせることで，組立式手法に共通する欠点である配置結果の順序依存問題に対応できる．
4. 従来のコンパクション手法のための初期配置を与えることに利用可能である．

また，本手法の有用性を確認するため計算機実験を行い，凹部を含む完全な任意形状ブロックの配置や，配置領域に制限を設けての配置も可能であることを示した．

第 5 章

ウイルス進化型アルゴリズム

5.1 まえがき

遺伝的アルゴリズムはダーウィンの進化論に基づいた最適解探索手法であるが，その一方でダーウィンの進化論以外にも多くの進化論が提案されている．ウイルス進化論はその一つであり，生物はウイルスの感染により進化するという考えに基づいている．本章ではこのウイルス感染による進化という点に着目し，ウイルス進化論に基づく新しい進化型アルゴリズムを提案する．提案するアルゴリズムでは，ウイルスを単体の生物ではなく遺伝子を運ぶ生物器官の一つとして捉えるとともに，交叉を用いずウイルス感染を唯一の探索操作としている．また，このウイルス感染はいくつかの遺伝子が急速に集団全体に拡散，進化することを意味しており，提案手法は遺伝的アルゴリズムよりも短時間での解の探索が可能となる．提案手法の有用性を確認するため計算機実験を行い，提案手法と遺伝的アルゴリズムとを比較した．その結果，提案手法は計算時間が短かく，収束も速く，また，最適解を得る確率も高いということが確認された．

5.2 研究の目的

遺伝的アルゴリズム（Genetic Algorithm:GA）[2, 3, 4]はダーウィンの進化論に基づいた最適解探索手法である．その一方で進化論はダーウィンの進化論だけではなく，他にも多くの進化論が提案されている．ウイルス進化論[15]はその中の一つであり，生物はウイルスにより進化するという考えに基づいている．本章ではウイルス進化論に基づく新しい進化型アルゴリズムを提案する．

現在までにウイルス進化論に基づく手法は，いくつか提案されている [16, 17, 18]．しかし，これらの手法ではウイルスを生物のように扱っているのに対し，本研究ではウイルスを生物としては捉えない．したがってウイルス自体は集団を作らず，適応度をもたず，進化もしない．すなわち本研究では，ウイルスを遺伝子を運ぶ生物器官の一つとして捉え，その機能は染色体の一部遺伝子を単純に置き換えるものとする．また，従来手法がGAに何らかの形でウィルスの感染を組み合わせたものであるのに対し，提案手法はGAとは異なる新しい進化型アルゴリズムである [27, 32]．

提案手法では，まず個体集団はGAの集団と同じものとする．また交叉は用いず，個体集団はウイルスの感染により進化する．感染は，まず適応度に応じて選択された一つの染色体上に，遺伝子座を二つランダムに選択する．それら二つの遺伝子座に挟まれた遺伝子をウイルスとし，そのウイルスを他の染色体に感染させる．言い換えるならば，適応度の高い染色体の持つ遺伝子の一部を集団全体に拡散させることになるが，必ずしも集団は改善されるとは限らないため，局所最適解に陥ることなく最適解を得ることが可能となる．突然変異は感染時のエラーとして振る舞い，感染時に置き換えられた遺伝子の一部を他の遺伝子に書き換えることにより実現する．以上の選択，感染，突然変異の操作を繰り返すことで，最適解を得ることが可能となる．この提案手法は，ウイルスの感染により集団全体へ遺伝子断片を急速に拡散可能であることから，局所探索能力が向上し，また遺伝子操作の処理の軽さから，その計算時間はGAに比べ短くなるという特長を有する．

提案手法の有用性を確認するため計算機実験を行い，同一問題において提案手法とGAを比較した．その結果，提案手法の計算時間はGAのものよりも短かく，ナップザック問題において53.1%，巡回セールスマン問題において37.9%の減少が認められた．また，収束も速く，最適解を得る確率も提案手法の方が高いことが分かった．以上のように，提案手法は従来のGAに比べ高い有用性を持つことが確認された．

5.3　ウイルス進化論

本節では，まずウイルス進化論の一般的特徴について述べ，次に提案手法の概要について述べる．

GAは最適解探索手法の一つであり，初期条件に依存せず局所最適解を避け最

適解を探索可能という特長を持つ．しかし，最適解を得るまでの計算時間は長いという欠点があり，これはGAがダーウィンの進化論に基づくことによるものと考えられる．すなわちダーウィンの進化論において，ある個体に生じた性質が個体集団全体に波及するのに多くの世代が必要なように，GAにおいても集団全体がある解に収束するには時間を必要とするということである．一方，進化論はダーウィンの進化論だけではなく，多くの進化論が提案されている．ウイルス進化論はその中の一つであり，生物はウイルスの感染により進化するという考えに基づいている．このことによりウイルス進化論においては，ある個体に生じた性質が個体集団全体に波及するのに世代を重ねる必要はなく，ウイルスの爆発的な感染力によって個体から個体へ遺伝子断片が急速に拡散し，進化することが可能となる．したがって，この点に着目し，ウイルス進化論に基づく新しいアルゴリズムを構築すれば，計算時間の短縮が期待できる．

ウイルス進化論に基づく手法は，今日までにいくつか提案されている[16, 17, 18]．しかし，これらの手法では，ウイルス感染の目的が本論文のそれとは大きく異なっている．文献[16]の手法は，制約充足問題の部分解をウイルスとしてウイルス集団を形成し，通常のGAに加えウイルスによる感染を導入することで，探索能力の向上をねらったものである．このとき，各ウイルスに適応度を持たせて，探索に有効なウイルスを確率的に感染させることで解の収束を高めている．また，文献[17, 18]の手法もGAにウイルスの感染をプラスしたものであるが，この手法ではウイルスは感染により適応度が改善されるときのみ感染を許される．すなわち，ウイルスの感染は局所探索を意味する．したがって，探索能力を向上させるためウイルス自体の進化が必要である．

このように，これら従来の手法においてはウイルスは生物のように扱われ，またウイルスの感染はGAと組み合わせた探索能力の向上が目的となっている．しかし，ウイルス進化論においてはウイルスは生命体ではなく，個体から個体へ遺伝子断片を運ぶ生物器官の一つと考えられている．すなわち，ウイルスは単に染色体の一部遺伝子を置き換える機能を持つ生物の一部にすぎず，ウイルス自体が単独で集団を作ったり，進化することはない．そこで，提案するアルゴリズムではこの発想に基づき，ウイルスは集団を作らず，適応度を持たず，また進化もしない．そして，単細胞生物はウイルスの感染で進化しているというウイルス進化論の主張を受け，ウイルスの感染を唯一の探索操作とし，世代を重ねることで進化する従来のGAとの組み合わせではない，新しい進化型アルゴリズムを提案する．

5.4　ウイルス進化型アルゴリズムの提案

ここではウイルス進化論に基づく新しいアルゴリズムを提案する．以下では，これを「ウイルス進化型アルゴリズム（Virus Evolutionary Algorithm : VEA）」と呼ぶこととする．

5.4.1　アルゴリズム

提案するVEAはGAと同様に，与えられた問題の解を表わす多数の染色体により個体集団を生成し，その集団に対し遺伝子操作を繰り返すことによって集団を進化させ，最適解を探索するアルゴリズムである．したがって，全体の処理手順は次の通りになる．

1. 集団をランダムに生成する．
2. 適応度に応じて染色体を一つ選択する．
3. ウイルスを感染させる．
   (a) ステップ 2 で選ばれた染色体上の遺伝子座を二つランダムに選択する．
   (b) 二点に挟まれた遺伝子をウイルスとして取り出す．
   (c) ウイルスを他の染色体に感染させる．
   (d) ステップ (c) で置き換えられた遺伝子に対し，突然変異を起こす．
4. ステップ 2,3 を繰り返す．

以下では，上記の手順について順に説明する．

1. **集団**（*population*）
   VEAにおいて集団はGAの集団と同じものを意味し，集団サイズ M でランダムに生成されるものとする．
2. **選択**（*selection*）
   選択は適応度に応じて染色体を一つだけ選択する．選択方法にはGAで用いられるルーレット戦略や，トーナメント戦略などが考えられ問題に応じて適当な戦略を用いる．また，この染色体は探索を担うウイルスの元になる重要なものであるため（図5.1），最大適応度を持つ染色体を選択する戦略も有効である．
3. **感染**（*infection*）
   交叉はVEAでは用いず，集団は感染によってのみ進化する．感染は次の手順で行われる．

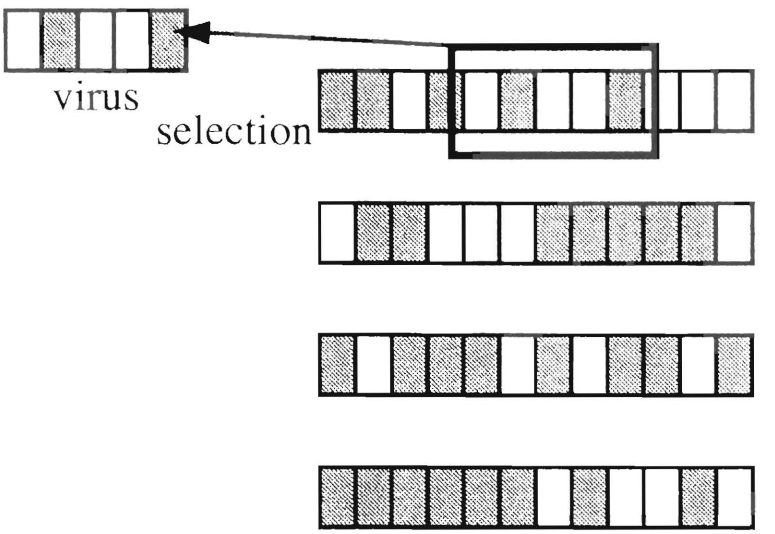


図 5.1　選択の例

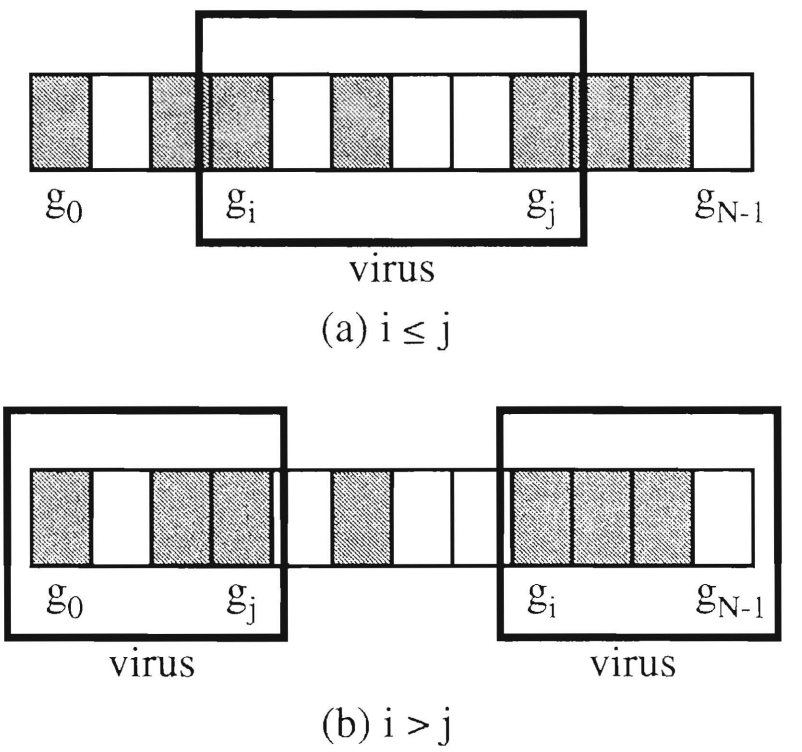


図 5.2　ウイルスの取り出しの例

(a) 選択によって選ばれた染色体上の遺伝子座を $g_0 \cdots g_{N-1}$ とし，その中から二点をランダムに選択する．

(b) それらを選択した順に g_i，g_j とし，$i \leq j$ のときは g_i から g_j までの遺伝子を取り出しウイルスとする．また $i > j$ のときは，ウイルスは g_0 から g_j までと，g_i から g_{N-1} までの２つの断片で構成されるものとする（図 5.2）．

(c) そして，ウイルスを感染率 p_i で，他の染色体に感染させる．感染は各染色体の遺伝子をウイルスの持つ遺伝子に，それぞれの遺伝子座ごとに置き換えることで実現する（図 5.3）．

以上の操作は，いくつかの遺伝子が集団全体に急速に拡散することを意味する．このとき，ウイルスの元になった染色体を適応度に応じて選択しているため，集団全体は確率的に適応度が高くなると期待される．しかし，ウイルス自体はランダムに生成されているため，ウイルスとして取り出された遺伝子の持つ情報が，必ずしも適応度を高める要素であるとは限らない．すなわち，集団全体としては必ずしも良くなるとは限らず，ときには悪くなることも考えられる．したがって，このアルゴリズムはGAと同様に局所最適解からの脱出が可能となり，最適解を探し出すことができる．

(d) **突然変異**（*mutation*）

突然変異は感染時のエラーとして振る舞うとする．すなわち，ウイルスが感染するとき遺伝子にコピーミスが起こったと考え，ウイルス感染により置き換えられた遺伝子を突然変異率 p_{mv} で，他の遺伝子に書き換えることで実現する（図 5.4）．これはウイルス進化論においてウイルスの持つ遺伝子が，宿主の遺伝子に比べ急速に変化することを表わすものである．なお，感染により同じ情報が集団の広範囲に一度に波及するVEAにおいては，突然変異は集団の多様性を維持するために重要な要素である．

4. **サイクル**（*cycle*）

以上の選択，感染，突然変異の一連の操作をここではサイクルと呼ぶことにする．これはGAにおける世代にあたるものであるが，VEAではGAのように親集団から子集団が生成されるのではなく，集団そのものが遺伝子操作により変化している．したがって，世代という言葉は適当ではないため，遺伝子操作の繰り返し単位をここではサイクルとする．このサイクルを繰り返すことで探索が進み，最終的に最適解を得ることが可能となる．

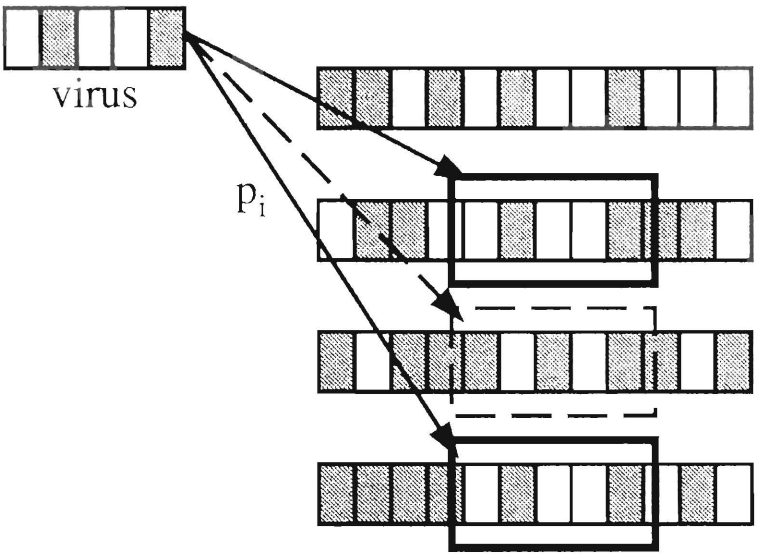


図 5.3　感染の例

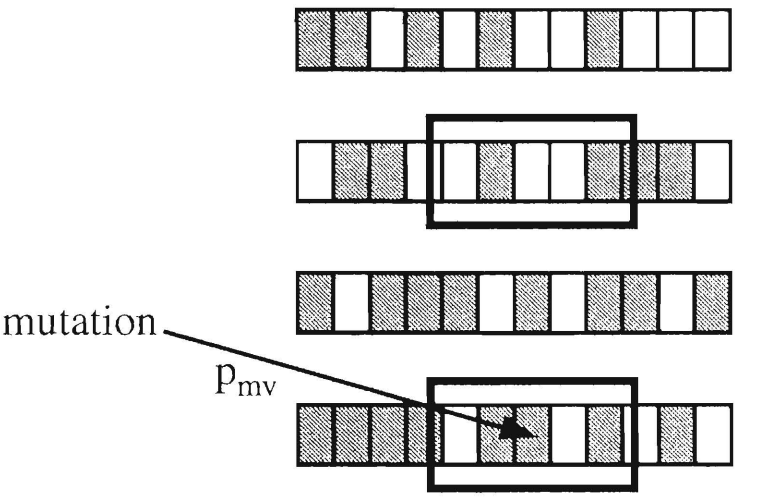


図 5.4　突然変異の例

5.4.2　計算時間の短縮

VEAは，その操作が従来のGAに比べ軽いため，計算時間が短いという特長を持つ．この理由としては，次の3つが考えられる．

1. 選択操作が，VEAではサイクルごとに1回なのに対し，GAでは世代ごとにM回になりVEAの方が操作回数が少ない．
2. 交叉，感染，突然変異での操作の対象となる遺伝子の個数は，染色体長をNとして，VEAがサイクルごとに平均$(N+1)M/2$なのに対し，GAでは世代ごとにNMとなり，VEAの方が少なく短時間での処理が可能である．
3. VEAが集団を直接操作しているのに対し，GAでは基本的に親集団から子集団を生成し，その子集団を新たな親集団にするといった操作が必要であり，VEAの方が有利となる．

以上の理由から，VEAの計算時間はGAに比べ短縮される．

5.4.3　集団の多様性と初期収束

VEAは感染という「いくつかの遺伝子を集団全体に急速に拡散させる」遺伝子操作により探索を行うため，GAよりも集団の多様性が低く押さえられ初期収束に陥りやすいと考えがちである．しかし，VEAはGAとは異なる新しい進化型アルゴリズムであるため，その振る舞いも根本的にGAとは異なり多様性の維持は重要ではないと考えられる．

ここで，多様性が完全に失われ同一の個体で集団が占められた場合を仮定してみる．GAの場合，交叉によっては新しい解を生成することはできないため，探索は突然変異によってのみ行われることになる．しかし突然変異による探索も，それによって生じた個体の適応度が適当な値を持たなければ，次の選択のときに淘汰され個体集団はまったく変わらないということが起こり得る．したがって，GAの場合ある程度の多様性を維持し続けなければ，初期収束に陥る可能性を持っており，また失われた多様性を回復させることは難しい．一方，VEAの場合も，感染によっては新しい解を生成することはできないため，探索は突然変異によってのみ行われることになるのは同様である．しかしVEAの場合，突然変異によって生じた個体の適応度の値は重要ではなく，たとえ0であったとしても問題にならない．なぜなら，VEAには淘汰にあたる操作が存在しないため，突然変異によって生じた個体は確実に次の個体集団に受け継がれ，すなわち，新たに生じた遺伝子の多様性は失われることはないからである．したがって，VEAの場合は多

様性が完全に失われた状態からも，サイクルを重ねることで確実に新たな多様性を生じることになる．

また，VEA では突然変異を感染時のエラーとして振る舞うものとし，突然変異を感染により置き換えられた遺伝子にのみ限定して起こしている．このことは，感染が多様性を失わせると同時に，突然変異による確実な多様性の発生を約束するものであることを意味している．以上のようにVEAは，選択（淘汰），感染，突然変異というGAとは異なる遺伝子操作のバランスの上に構成された，新しい進化型アルゴリズムであり，GA とはその振る舞いも根本的に異なっている．

5.5　計算機実験

提案手法の有用性を確認するため計算機実験を行った．実験においては，VEA と GA で同じ問題を解くことで両者を比較している．対象とする問題は，典型的な組み合せ問題として知られる，ナップザック問題と巡回セールスマン問題とする．

5.5.1　ナップザック問題

ナップザック問題は次のようなものとする．荷物が N 個あり，各荷物 i ($i = 1, \cdots, N$) は重量 w_i と価値 v_i を持ち，そしてナップザックの容量を c とする．このとき，ナップザックの容量内で価値を最大にする荷物の組み合わせを求める．なお，荷物の状態を示す変数 x_i を導入し，その値は荷物 i がナップザックに入っているときは1，入っていないときは0とする．

5.5.1.1　実験条件

ここでは実験条件について述べる．まず，VEA と GA の両方に共通する条件は次の通りである．

コーディング: 染色体は x_i を $1 \sim N$ まで順に並べたものとする．

適応度: 適応度 f は次式で定義する．

*

$$f = \max\left[0, \sum_{i=1}^{N} v_i x_i - \alpha \max\left\{0, \sum_{i=1}^{N} w_i x_i - c\right\}\right] \tag{5.1}$$

初期集団: 初期集団は集団サイズ M でランダムに生成する．

次に，GAにおける条件は次の通りである．

選択: ルーレット戦略とエリート保存戦略を用い，エリート保存数は1とする．

交叉: 1点交叉および2点交叉を用い，交叉率はp_cとする．

突然変異: 突然変異率はp_{mg}とする．

GAの構成には上記のような基本的なもの以外にも様々な構成が考えられるが，どのような構成が望ましいかは問題に依存し，また，構成の善し悪しの決定は経験に依らざるを得ない．そこで，本実験では経験による影響の回避を目的に，基本的なGAの構成を用いて実験を行うこととした．なお，交叉については提案手法における感染が2点交叉の遺伝子操作に近いことから，両者を比較するため2点交叉についても実験を行うものとする．

最後に，VEAにおける条件は次の通りである．

選択: 最大適応度の染色体を選択する．

感染: 感染率はp_iとする．

突然変異: 突然変異率はp_{mv}とする．

なお上記のVEAにおける条件では，エリート保存戦略を用いてはいないが，実際には最大適応度を持つ染色体は次のサイクルにそのまま継承される．これは，VEAでは選択された染色体が，感染や突然変異によって変更されることが一切ないからである．したがって，上記の条件下では，エリート保存戦略を用いる必要はない．

以上の条件下で，VEAとGAの比較実験を行う．

5.5.1.2　実験結果と考察

実験は荷物の個数が異なる次の2つのモデルについて行った．

モデル1 荷物の個数を24とし，集団サイズ（M）を50，染色体長（N）を24とする．

また，適応度の式（5.1）において$\alpha = 100$とする．

モデル2 荷物の個数を100とし，集団サイズ（M）を1000，染色体長（N）を100とする．

また，適応度の式（5.1）において$\alpha = 2000$とする．

上記の集団サイズ，染色体長はGA，VEAに共通とし，その他の実験パラメータは表5.1に示す．ここでGAの交叉率，突然変異率，VEAの感染率，突然変異率は次のように決定した．パラメータの決定にはモデル1を用いGAの交叉は1点交叉とした．まず突然変異率を0.05と固定し，交叉率，感染率を変えそれぞれ実

験を10回行い，その平均として表5.2のような結果を得た．これらの結果から交叉率，感染率を0.95とした．次に交叉率，感染率を0.95と固定し，突然変異率を変え同様に実験を行い，その平均として表5.3のような結果を得た．これらの結果からGAの突然変異率を0.01，VEAの突然変異率を0.10とした．なお，この突然変異率の差は，各アルゴリズムにおいて突然変異での操作対象とする遺伝子が異なり，突然変異の探索過程における意味も違うことに起因すると考えられる．

表 5.1　実験パラメータ

遺伝的アルゴリズム		
世代数	T_g	500
交叉率	p_c	0.95
突然変異率	p_{mg}	0.01
提案手法		
サイクル数	T_c	500
感染率	p_i	0.95
突然変異率	p_{mv}	0.10

実験結果を表5.4，表5.5に示す．実験はそれぞれのアルゴリズムおよび交叉について10回行い，その平均を算出した．なお，表5.4，表5.5は最終結果であり，最大適応度は500世代もしくは500サイクル後の値である．また，計算時間はHP-9000/755ワークステーションで要した時間である．GAでは交叉の違いにより探索能力に差が見られ，どちらのモデルにおいても2点交叉の結果の方が優れている．しかし，VEAとの比較においては交叉の違いによる本質的な差は認められないため，以下では特に断りのない限りGAでは2点交叉の結果を対象として，計算時間，収束速度，最適解を得る確率について考察する．

1. **計算時間**

   表5.4，表5.5からわかるように，VEAの計算時間はGAのものよりも短く，その減少率はモデル1で53.1%，モデル2で70.5%となり，VEAの有用性が認められる．なお，この減少率の差は主に選択回数の差によるものと考えられる．すなわち，GAでは世代ごとにM回と集団サイズに比例して計算時間が増すが，VEAでは集団サイズとは無関係に常に1回であり計算時間が変化しないことによる．また，図5.5に示すモデル2の計算時間と適応度

表 5.2　交叉率，感染率と適応度の関係

交叉率・感染率	最大適応度	
	遺伝的アルゴリズム	提案手法
0.75	189.6	190.1
0.80	189.5	189.6
0.85	189.6	189.9
0.90	189.5	190.2
0.95	189.9	190.5

表 5.3　突然変異率と適応度の関係

突然変異率	最大適応度	
	遺伝的アルゴリズム	提案手法
0.01	190.2	184.6
0.03	190.1	187.8
0.05	189.9	189.8
0.10	189.7	191.0
0.15	188.1	190.8

表 5.4 ナップザック問題の実験結果（モデル1）

	最大適応度		
	遺伝的アルゴリズム		提案手法
	1点交叉	2点交叉	
1st	**191**	**191**	**191**
2nd	**191**	190	**191**
3rd	190	189	**191**
4th	**191**	**191**	**191**
5th	190	**191**	**191**
6th	190	**191**	**191**
7th	**191**	**191**	**191**
8th	188	**191**	**191**
9th	190	**191**	**191**
10th	190	**191**	**191**
平均	190.2	190.7	**191.0**
平均計算時間（s）	7.99		3.75

表 5.5　ナップザック問題の実験結果（モデル2）

	最大適応度		
	遺伝的アルゴリズム		提案手法
	1点交叉	2点交叉	
1st	4457	4506	4639
2nd	4471	4493	4646
3rd	4486	4555	4639
4th	4511	4526	4637
5th	4547	4538	4651
6th	4484	4515	4643
7th	4536	4509	4642
8th	4436	4517	4645
9th	4525	4503	4641
10th	4479	4551	4649
平均	4493.2	4521.3	4643.2
平均計算時間（s）	1192		352

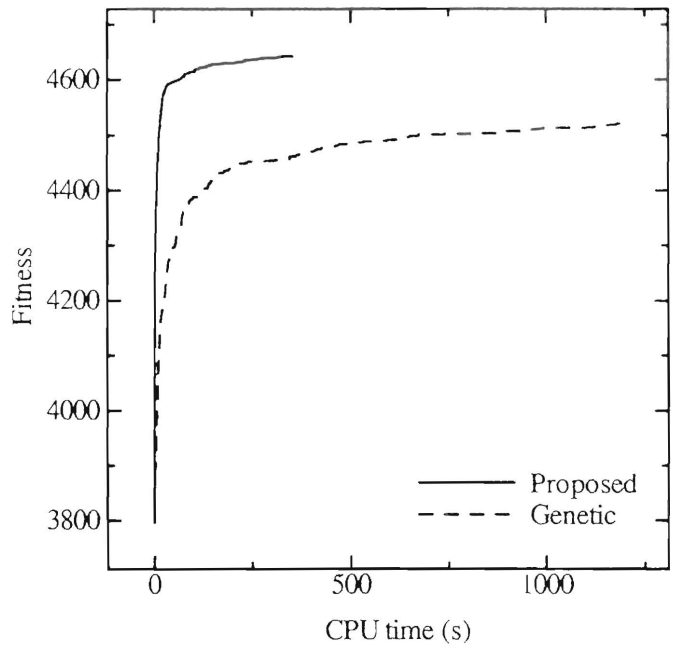


図 5.5　計算時間と適応度の関係（モデル2）

の関係からも明らかなように，VEA はGA に比べ短い時間で同様の探索が可能になっている．なお，この図でいう適応度は，各10回の実験における適応度の最大値を世代またはサイクルごとに平均したものである．これらの結果はモデル1においても同様の傾向を示す．

2. **収束速度**

図5.6にGA における世代数と適応度の関係を，図5.7にVEA におけるサイクル数と適応度の関係を，それぞれモデル1について示す．これらの図で示す適応度は，各10回の実験での適応度の最大値，平均値，最小値を世代またはサイクルごとに平均したものである．初期世代またはサイクルの最大値の変化から若干であるがVEA の方がGA よりも収束が速い，すなわち，短いサイクルで最適解に近づくことが分かる．なお，モデル2においても同様の傾向が認められる．

3. **最適解を得る確率**

モデル1では全探索により，最適解の適応度が191であることが確認されている．GA では，この最適解を1点交叉では10回中4回，2点交叉では10回中8回しか得られなかったが，VEA では10回すべてで最適解を得ることが

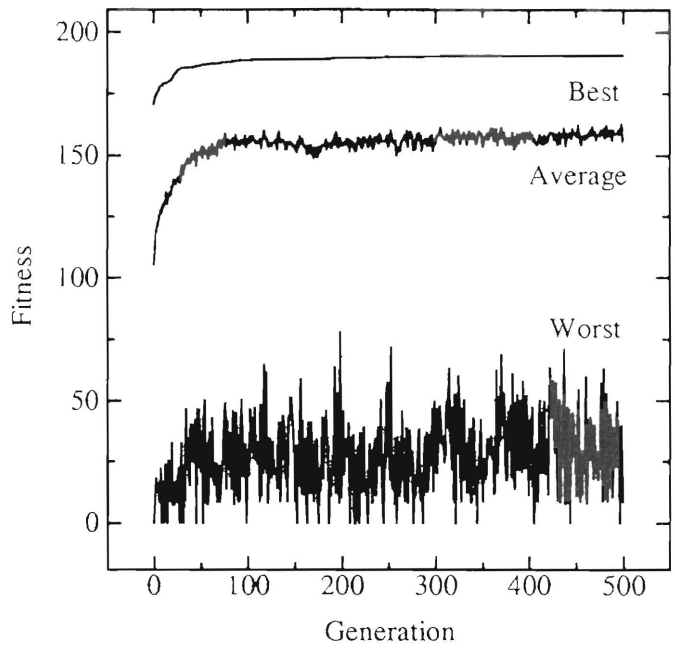


図 5.6　世代と適応度の関係（モデル1）

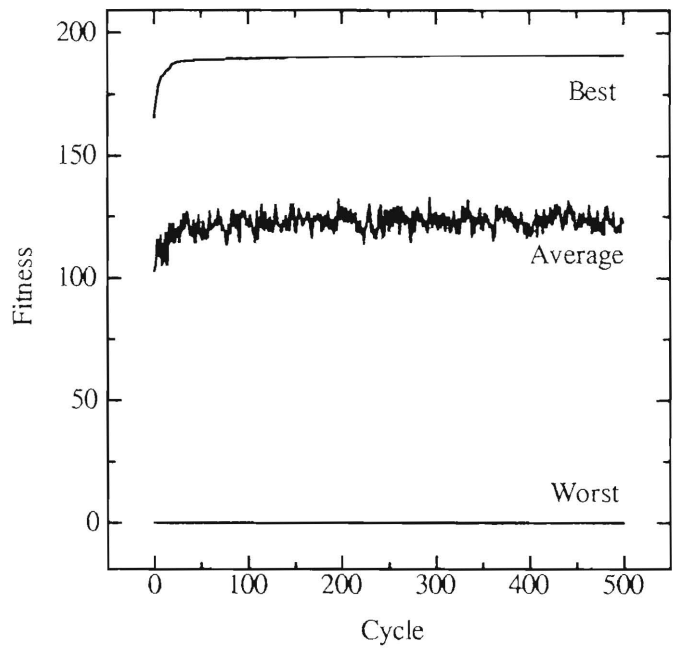


図 5.7　サイクルと適応度の関係（モデル1）

できる，その確率は上がっている．また，モデル2では最適解の適応度は不明だが，欲張り法[5]を用いて得られた近似解の適応度は4617となっている．VEAでは10回すべてでこの値を上回る結果を得ているが，GAでは交叉に関わらずこの値を上回ることはできなかった．以上の結果は，VEAの方が局所探索性に優れていることに起因すると思われる．

なお，図5.6，図5.7において，GAでは最小値もある程度の値を持っているのに対し，VEAでは最小値がほとんどのサイクルで0，すなわち，ナップザックの許容量を超える解になっている．これは，GAではナップザックの許容量を超える解は淘汰され次世代には生き残れないのに対し，VEAでは淘汰にあたる操作は存在しないため，いつまでも集団に存在しつづけることに原因があると考えられる．

5.5.2　巡回セールスマン問題

次に巡回セールスマン問題についても同様に比較実験を行う．巡回セールスマン問題は次のようなものとする．都市数をNとし，都市には1からNまでの番号がつけられているものとする．また，都市i，j間の距離を$d_{ij}(=d_{ji})$とする．都市の訪問順序を（$1\cdots N$）の置換（$t_1\cdots t_N$）で表わしたとき，式（5.2）を最小にする訪問順序を求める．ただし，$t_{N+1}=t_1$とする

$$d_{total}=\sum_{i=1}^{N} d_{t_i,t_{i+1}} \tag{5.2}$$

この問題を解くにあたっては，都市の訪問順序を染色体とすることが一般的である．その場合，染色体が順列であるという条件を満足させるために，コーディングか遺伝子操作に何らかの工夫が必要となり，多くの手法が提案されている[2, 5]．そして，それらの手法を探索能力で比較した場合，巡回路に含まれる枝や部分巡回路に着目した交叉を用いた方が，有効であることは知られている．しかし，本実験では元々の遺伝子操作が異なる二つの手法の比較を目的としているため，GAの交叉という遺伝子操作への工夫をもって，順列という条件を満足させる手法を用いることは，まったく同様の遺伝子操作をVEAの感染として定義できない以上，比較実験としては不適当である．そこで，Grefenstetteらによる方法[19]を用いてコーディングへの工夫を行うことにする．この工夫により，遺伝子型と表現型が一対一に対応づけられ，GAおよびVEAでの通常の遺伝子操作の適用が可能になる．

5.5.2.1　実験条件

ここでは巡回セールスマン問題の実験条件について述べる．まず，コーディングと適応度は次の通りである．

コーディング: 都市の訪問順序を $(t_1 \cdots t_N)$ とし，また都市を適当な順序で並べたリスト W（定数）を用意する．そして，i 番目に訪問する都市を，未訪問都市のリスト $W - \{t_1, \cdots, t_{i-1}\}$ の何番目であるかを表わし，これを遺伝子 l_i とし，リスト $L = (l_1 \cdots l_N)$ を染色体とする．

適応度: 適応度 f は次式で定義する．

$$f = \frac{N}{\sum_{i=1}^{N} d_{t_i, t_{i+1}}} \tag{5.3}$$

これ以外の初期集団，選択，交叉，感染，突然変異などの実験条件は，ナップザック問題のものと同じとする．

5.5.2.2　実験結果と考察

実験は都市数とその配置が異なる次の 2 つのモデルについて行った．

モデル 1 各都市を円上に配置し都市数は 24 とする．

また，集団サイズ（M）を 50，染色体長（N）を 24 とする．

モデル 2 各都市をランダムに配置し都市数は 51 とする．

また，集団サイズ（M）を 1000，染色体長（N）を 51 とする．

上記の集団サイズ，染色体長は GA，VEA に共通とし，その他の実験パラメータはナップザック問題で用いたものと同じ表 5.1 に示すものを用いる．また，実験結果を表 5.6，表 5.7 に示す．この場合も，実験はそれぞれのアルゴリズムについて 10 回行い，その平均を算出した．実験に用いた計算機は，ナップザック問題で用いた計算機と同じものである．なお，ナップザック問題の場合と同様に，以下では特に断りのない限り GA は 2 点交叉の結果を対象として考察する．

1. **計算時間**

   表 5.6，表 5.7 からもわかるように，VEA の計算時間は GA のものよりも短い．その減少率はモデル 1 で 37.9%，モデル 2 で 61.4% となり，ナップザック問題と同様に VEA の有用性が認められる．また，図 5.8 にモデル 1 の計算時間と巡回距離の関係を示す．この図でいう巡回距離も，各 10 回の実験での巡回距離の最短値を世代またはサイクルごとに平均したものである．なお，モデル 2 においても同様の傾向を示す．

表 5.6　巡回セールスマン問題の実験結果（モデル1）

	最短巡回距離		
	遺伝的アルゴリズム		提案手法
	1 点交叉	2 点交叉	
1st	56.1	36.4	**31.3**
2nd	47.1	45.2	51.2
3rd	**31.3**	56.4	49.6
4th	55.7	46.9	**31.3**
5th	56.7	57.6	**31.3**
6th	58.1	**31.3**	**31.3**
7th	48.7	42.8	**31.3**
8th	53.8	56.8	**31.3**
9th	44.9	38.6	54.7
10th	53.3	47.6	33.8
平均	50.6	46.0	37.7
平均計算時間（s）	11.62		7.22

表 5.7　巡回セールスマン問題の実験結果（モデル2）

	最短巡回距離		
	遺伝的アルゴリズム		提案手法
	1点交叉	2点交叉	
1st	230.7	222.2	180.8
2nd	225.3	208.3	165.9
3rd	239.8	211.4	164.5
4th	232.9	226.8	182.9
5th	230.0	216.2	182.0
6th	219.6	219.7	152.9
7th	225.2	227.4	158.9
8th	225.4	224.0	161.6
9th	237.7	232.9	175.2
10th	231.0	222.0	149.4
平均	229.8	221.1	167.4
平均計算時間（s）	1184		457

2. **収束速度**

図5.9，図5.10にモデル2の世代またはサイクルと巡回距離の関係を示す．これらの図の初期段階での最短値の変化から，GA よりも VEA の方が収束が速く，また，その収束した値から，VEA の方が探索能力が高いことがわかる．この収束と探索能力の差は，VEA が行う感染という遺伝子操作が，ある一つの解が持つ情報を集団全体に広く拡散させるという操作のために，GA よりも集団全体がある一つの解の周辺に集中しやすくなり，その結果として局所探索能力が高くなっていることに原因があると思われる．なお，モデル1においても同様の傾向が認められる．

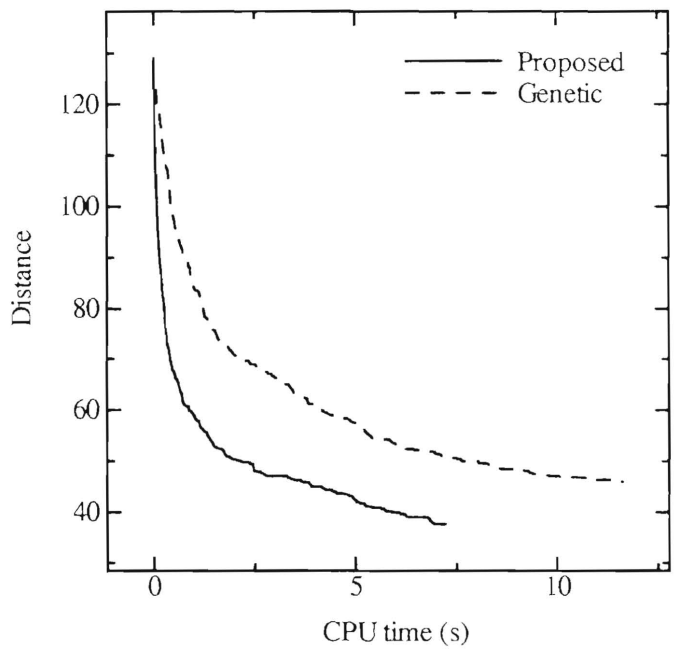


図 5.8　計算時間と巡回距離の関係（モデル1）

3. **最適解を得る確率**

モデル1では都市を円上に配置したため最短巡回路は明らかであり，その距離は31.3であることがわかっている．GA では，どちらの交叉においても最適解を10回中1回しか得られなかったが，VEA では6回得ることができ，その確率はここでも上がっている．これは VEA の方が局所探索性に優れていることと同時に，局所最適解に陥る確率も低いことを示している．このことから，VEA では局所探索と広域探索の両方の探索がうまく機能しているとも考えられる．また，モデル2の最短巡回路は図5.11に示すもので，その距離は86.0であることがわかっている．この場合 GA および VEA のどち

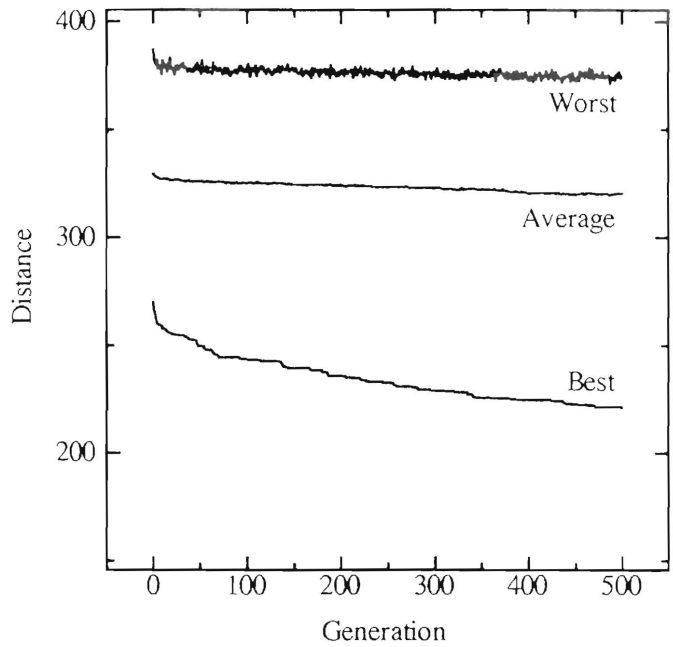


図 5.9　世代と巡回距離の関係（モデル2）

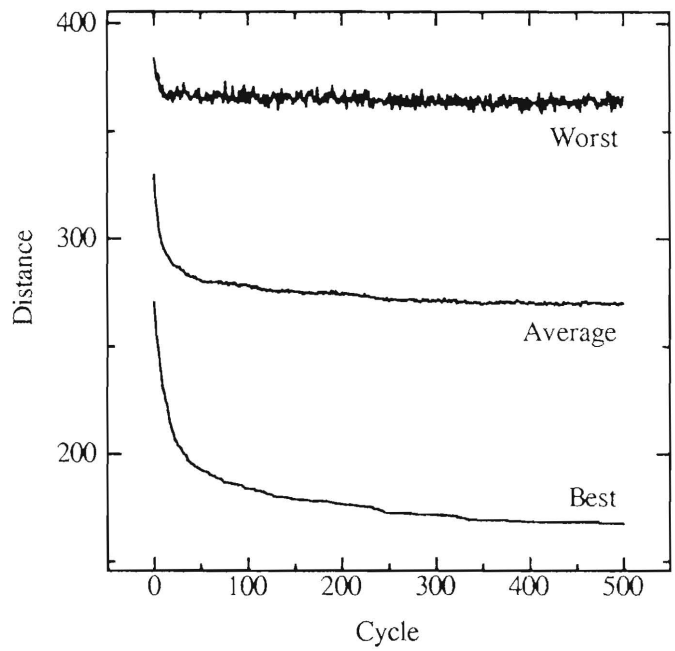


図 5.10　サイクルと巡回距離の関係（モデル2）

らにおいても最適解を得ることはできなかった．これは今回の実験において同条件での比較を優先して採用した，Grefenstetteらによるコーディング法を用いてのGAおよびVEAの構成では，探索能力が元々あまり高くないことに原因があると思われる．しかし，GA（2点交叉）では平均最短巡回距離が221.1と，最適解に対して257%の結果しか得られなかったが，VEAでは平均最短巡回距離は167.4と最適解の195%まで探索を進めることができ，VEAの方がGAより探索能力に優れているといえる．

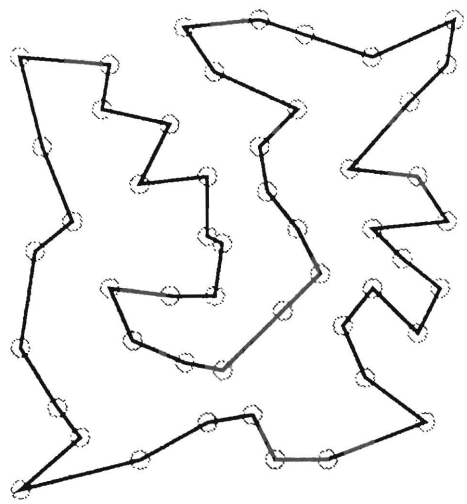


図 5.11　巡回セールスマン問題の例（モデル2）

5.5.3　考察

実験全体を通して，GAに対してVEAは以下の特長を有することが確認された．

1. 計算時間が減少する．
2. 収束速度が上昇する．
3. 最適解を得る確率が高くなる．

なお，巡回セールスマン問題（モデル1）とナップザック問題（モデル1）は共に，集団サイズが50，染色体長が24，世代およびサイクル数が500とまったく同じであるのに，計算時間の減少率に差を生じている．これは，それぞれの問題でのデコーディング時間の差によるものと考えられる．すなわち，既に述べたようにVEAの遺伝子操作はGAのものよりも軽く，この部分ではVEAの計算時間は

確かに短縮される．しかし，遺伝子型から表現型に変換し適応度を求めるというデコーディング処理は，手法によらず共通であり当然のことながら計算時間もまったく同じになる．そしてVEAやGA 全体は，このデコーディング処理と遺伝子操作で構成されており，このことから遺伝子操作における計算時間の比が，そのまま全体の計算時間の比にはならないことは明らかである．つまり，デコーディング処理と遺伝子操作の計算時間の比において，デコーディング処理の割合が大きくなればなるほど計算時間の比は1に近づく．本実験の場合，ナップザック問題のデコーディングよりも，巡回セールスマン問題のデコーディングの方が処理が重いことが，計算時間の減少率の差になって現れていると考えられる．

5.6　むすび

本章ではウイルス進化論に基づく新しい進化型アルゴリズムを提案した．このアルゴリズムではウイルスを遺伝子を運ぶ生物器官の一つと捉え，ウイルスの感染による遺伝子操作を，GA と組み合わせて探索能力の向上を目的とするものではなく，唯一の探索操作とした点が，GA や従来のウイルス進化論に基づく手法とは大きく異なっている．また，ウイルスの感染により集団全体へ遺伝子断片を急速に拡散可能であることから，局所探索能力が向上し，さらに遺伝子操作の処理の軽さから，その計算時間はGA に比べ短くなるという利点がある．

提案手法の有用性を確認するため，組み合わせ問題を解くことにより，提案手法とGA を比較する計算機実験を行った．その結果，GA に比べ提案手法では次のことが確認された．

1. 計算時間が減少する．
2. 収束速度が上昇する．
3. 最適解を得る確率が高くなる．

第 6 章

ウイルス進化型アルゴリズムを用いた配置手法

6.1 まえがき

本章では，“ウイルス進化型アルゴリズム”を用いた配置手法を提案する [28, 29]. ここで対象とする配置問題は“基本的な配置問題”および“ウェハー規模集積回路の再構成問題”であり，これらにおいてウイルス進化型アルゴリズムと，遺伝的アルゴリズムとで比較を行った．また，提案手法の有用性を確認するため計算機実験を行い，その結果，ウイルス進化型アルゴリズムは遺伝的アルゴリズムに比べ，計算時間が短かく，収束も速く，また，最適解を得る確率も高いということが確認された.

6.2 基本的な配置問題

6.2.1 研究の目的

この節では第3章でも扱った基本的な配置問題を対象として，ウイルス進化型アルゴリズムを用いた配置手法の提案と，遺伝的アルゴリズムとの比較を行い，その有用性を示す.

6.2.2　提案手法

本研究で対象とする配置問題は，第3.4節で述べた基本的な配置問題と同じものとする．すなわち，配置領域は正方形の区画が $N \times N$ に並んでおり，そこに配置するモジュールも正方形のものが N^2 個あるとする．前述したように，この基本的な配置問題の場合，単純なGAではなく何らかの工夫をしなければ効果的な探索を行うことは不可能である．そこで，本研究でもコーディングに工夫をすることで，情報の矛盾を回避する方法を取ることにし，そのコーディング法には第3.6.1節で述べた“リストによるコーディング法”を用いることにする．なおコーディング法としては，第3.6.3節で提案した“情報に冗長性を持たせたコーディング法”の方が有用ではあるが，しかし提案したコーディング法は，デコーディングに要する時間が染色体の情報により大きく異なるという特徴を持つ．したがって，本研究の目的であるウイルス進化型アルゴリズムと，遺伝的アルゴリズムとの比較，特に計算時間の比較においては適当ではないと考え，リストによるコーディング法を採用する．

6.2.3　計算機実験

ウイルス進化型アルゴリズムと，遺伝的アルゴリズムとの比較をするため計算機実験を行なった．この節では実験における幾つかの条件および，それらの条件下での実験結果について述べる．

6.2.3.1　実験条件

本実験では3.4節で述べた基本的な配置問題を対象とする．また，VEAおよびGAの諸設計については次のように定める．

まず，VEA,GA両者に共通する実験条件は次の通りである．

コーディング: 第3.6.1節で述べた“リストによるコーディング法”を用いる．

適応度の評価: 適応度 f は，仮想配線長 D とモジュール間の総接続本数 C から

$$f = \frac{1}{1 + \frac{D}{C}} \tag{6.1}$$

として定義する．

初期集団の生成：決められた個体数の染色体 M 個をランダムに生成する．

次に，VEAの実験条件は次の通りである．

選択: 最大適応度の染色体を選択する.
感染: 感染率はp_iとする.
突然変異: 突然変異率はp_{mv}とする.

最後に，GAの実験条件は次の通りである.
選択: ルーレット戦略とエリート保存戦略を用い，エリート保存数は1とする.
交叉: 1点交叉を用い，交叉率はp_cとする.
突然変異: 突然変異率はp_{mg}とする.

以上の条件下で，VEAとGAの比較実験を行う.

6.2.3.2　実験結果と考察

実験におけるパラメータは表6.1に示す値を用いた．実験モデルとしては3.4節で述べた配置問題において$N = 4$のモデルを用い，接続関係は図6.1に示すように格子状に接続されたものとする．したがって，仮想配線長の理論最短値は24になり，その時の適応度は

$$f = \frac{1}{1 + \frac{24}{24}} = 0.5$$

となる.

表 6.1　計算機実験パラメータ

共通		
集団サイズ	M	1000
染色体長	N^2	16
ウイルス進化型アルゴリズム		
サイクル数	T_c	500
感染率	p_i	0.95
突然変異率	p_{mv}	0.1
遺伝的アルゴリズム		
世代数	T_g	500
交叉率	p_c	0.95
突然変異率	p_{mg}	0.05

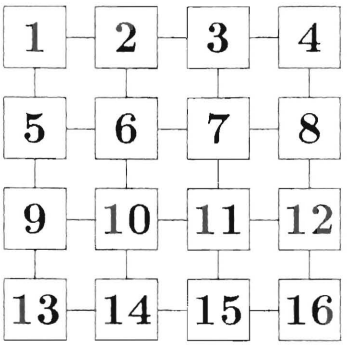


図 6.1　実験モデルの接続関係

実験結果を表6.2に示す．実験はそれぞれのアルゴリズムに対し５回行った．また，$f_{\max}$ はその世代の最大適応度，$\overline{f}$ はその世代の適応度の平均を表す．なお，表6.2に示す結果は最終結果であり，$f_{\max}$，$\overline{f}$ は500世代目の値である．計算時間はHP-9000/755 ワークステーションで要した時間である．

表 6.2　実験結果

	ウイルス進化型アルゴリズム		遺伝的アルゴリズム	
	$f_{\max}$	$\overline{f}$	$f_{\max}$	$\overline{f}$
1st	0.42	0.36	0.38	0.27
2nd	0.47	0.38	0.39	0.28
3rd	0.43	0.36	0.38	0.27
4th	0.40	0.35	0.43	0.28
5th	**0.50**	0.40	0.39	0.28
平均	0.44	0.37	0.39	0.28
平均計算時間 (s)	251.7		757.4	

表6.2からわかるように，VEA の計算時間はGA のものよりも短く，その減少率は

$$\frac{757.4 - 251.7}{757.4} \times 100 = 66.8\ (\%).$$

となり，VEA の有用性が認められる．また，図6.2に示す計算時間と適応度の関係からも明らかなように，VEA はGA に比べ短い時間での探索が可能になってい

る．さらに，図6.3に示すサイクル，世代数と適応度の関係からも，VEAの方が収束が速く，探索能力も高いことが確認できる．図6.4と図6.5には，それぞれのアルゴリズムで最大適応度になった配置を示す．VEAでは図6.4に示す通り最適配置を得ることができているが，GAでは図6.5のように最適配置を得ることができなかった．すなわち，このことからもVEAの方がより高い探索能力を持つということがいえる．

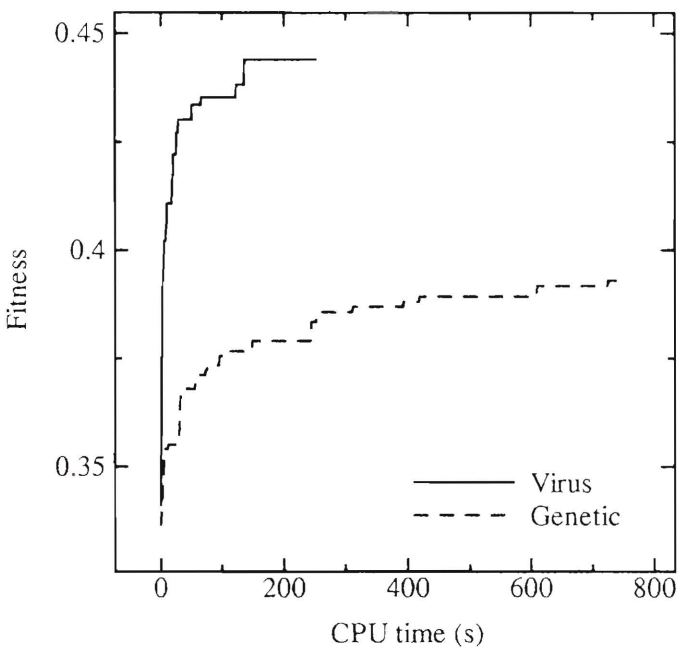


図 6.2　計算時間と適応度の関係

6.2.4　まとめ

この節では第3章でも扱った基本的な配置問題を対象として，ウイルス進化型アルゴリズムを用いた配置手法の提案と，計算機実験によりウイルス進化型アルゴリズムと遺伝的アルゴリズムとの比較を行った．その結果，遺伝的アルゴリズムに比べ，ウイルス進化型アルゴリズムでは次のことが確認された．

1. 計算時間が減少する．
2. 収束速度が上昇する．
3. 最適配置を得る確率が高くなる．

これは第5章の結果を指示するものである．

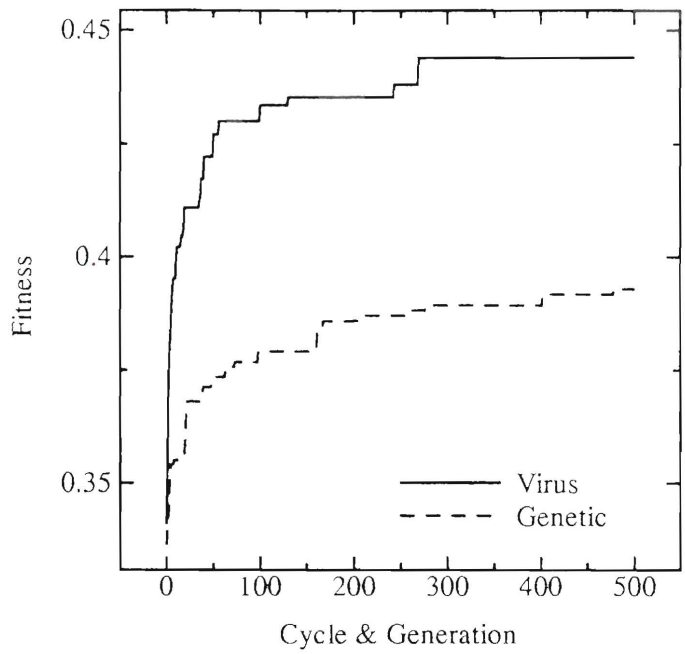


図 6.3　サイクル，世代数と適応度の関係

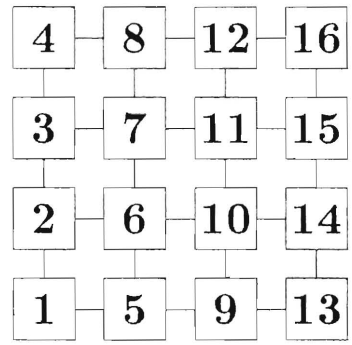


図 6.4　VEA の最適解

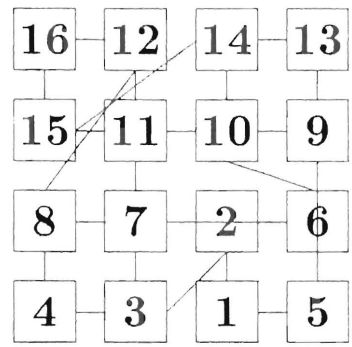


図 6.5　GA の最適解

6.3　ウェハー規模集積回路の再構成

6.3.1　研究の目的

ウェハー規模集積回路（Wafer Scale Integration:WSI）は一枚のウェハーの上に大規模な回路を集積し，通常のチップのように切り離すこと無しに，そのまま一つの大規模回路として利用する技術のことである[20]．したがって，WSI には次のような利点が期待できる．

1. どの LSI よりも大規模な回路を集積できる．
2. システムを 1 ウェハー上に集積することによって，回路間の相互配線長が短くなり，信号遅延時間を短くできる．
3. 組み立て工程削減による実装上の信頼性や，欠陥救済技術によってシステムの信頼性を向上できる．

しかし WSI の場合，必ず欠陥部分を含むような広い範囲に回路を集積するために，冗長構成をもたせ欠陥を迂回してシステムを構成する技術が不可欠となる．この“欠陥を迂回してシステムを構成する”技術は“WSI の再構成”とよばれている．

WSI の再構成では不良部を迂回する配線を形成する必要があるが，この技術は以下の条件を満足しなければならない．

1. ウェハーごとに異なる配線経路を短時間，低コストで形成できる．
2. 形成した配線経路が十分低抵抗で回路のブロック間遅延が少ない．

WSI の再構成手法は多くの研究がなされている[23]が，その中には遺伝的アルゴリズムを用いる手法[21, 22]も提案されている．そして，第5章で提案した“ウイ

ルス進化型アルゴリズム”は遺伝的アルゴリズムよりも高速な最適解探索手法として期待できる．そこで，本研究ではウイルス進化型アルゴリズムを用いたWSIの再構成手法を提案し，計算機実験によりその有用性を示す．

6.3.2　提案手法

対象とする問題は次のようなものとする．チップが$N \times N$に並べられており，その中のいくつかは欠陥チップであり利用できないとする．そして，並べられたチップの端に位置する 1 つから，欠陥チップを避けつつ鎖状に可能な限りチップを繋ぐことを目的とする．このとき連結可能なチップは上下左右にある 4 つのチップに限定する．なお，実際のWSIではチップは図6.6に示すように，正方形に並んでいるわけではないが，本質的には同じものと考えられるので上記の問題を対象とした．

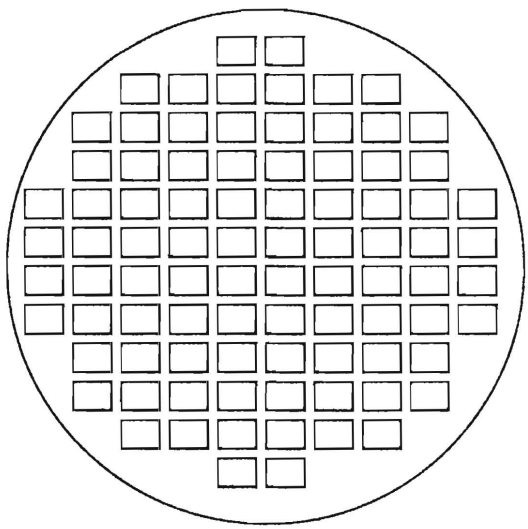


図 6.6　WSIの例

本研究で用いるコーディング法は，基本的に文献[21, 22]において提案されたものを用いるが，若干の変更を加えることにする．すなわち次のようなコーディングを行う．各チップは情報として“次に接続するチップの方向（d_i）”を持っているとし，この情報を一次元配列上に順に並べたものを染色体とする．したがっ

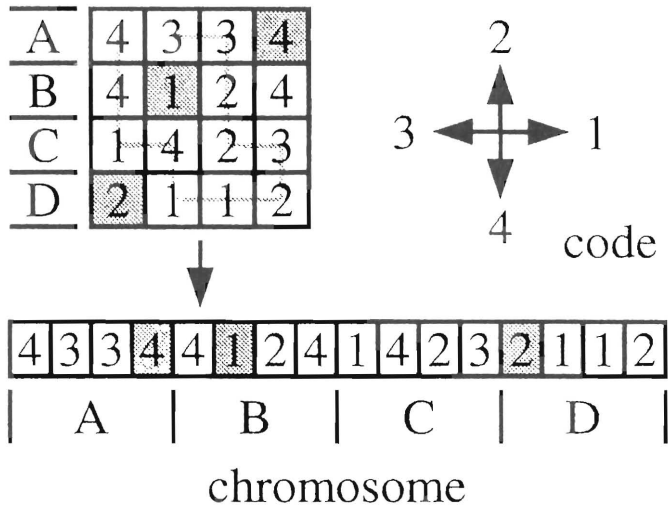


図 6.7　染色体の例

て染色体は

染色体：　d_1　d_2　d_3　$\cdots$　d_{N^2}

となる．そしてd_iの値が“1”のとき次に繋ぐチップの方向は“右”，“2”のときは“上”，“3”のときは“左”，“4”のときは“下”と定義する（図6.7）．なお，文献[21, 22]のコーディング法では，欠陥チップが持つd_iを“0”と定義するとしている．しかし，この“欠陥チップの位置”という情報は，すべてに共通する情報であり個々の染色体に持たせることは意味をなさないと思われる．そこで本研究では，この欠陥チップの位置情報を染色体に持たせるということを行わず，コーディングとは別の手段で情報を持つことにする．ここで，与えられた染色体の情報からチップの連結を決定する手順を述べる．

チップには座標(x,y) $(x,y = 1,2,\ldots,N)$が与えられているとし，座標(x,y)のチップが持つ情報は，遺伝子$d_{x+(y-1)\times N}$にあるとする．そして，座標(x,y)のチップが“欠陥”あるいは“既に連結済み”であることを保存する変数C_{xy} $(x,y = 1,2,\ldots,N)$を導入する．この変数C_{xy}は，座標(x,y)のチップが“欠陥”あるいは“既に連結済み”ならば0を，“利用可能”ならば1を持つものとする．さらに，連結した総チップ数をA_cとする．これらの変数を用いると配置位置決定手順は次のようになる．

ステップ1

$C_{xy} = 1\ (x, y = 1, 2, \ldots, N)$として初期化する．ただし，座標$(x, y)$のチップが欠陥のときは$C_{xy} = 0$とする．

また，$(x_c, y_c) = (0, 0)$，$A_c = 0$とする．

ステップ2

If $C_{x_c y_c} \neq 0$,

Then $A_c = A_c + 1$，$C_{x_c y_c} = 0$.

If $d_{x_c+(y_c-1)\times N} = 1$,

Then $(x_c, y_c) = (x_c + 1, y_c)$.

If $d_{x_c+(y_c-1)\times N} = 2$,

Then $(x_c, y_c) = (x_c, y_c - 1)$.

If $d_{x_c+(y_c-1)\times N} = 3$,

Then $(x_c, y_c) = (x_c - 1, y_c)$.

If $d_{x_c+(y_c-1)\times N} = 4$,

Then $(x_c, y_c) = (x_c, y_c + 1)$.

Else **ステップ4**へ.

ステップ3

If $1 \leq x_c \leq N$かつ$1 \leq y_c \leq N$を満たす,

Then **ステップ2**へ.

ステップ4

手順を終了する．

適応度（f）は，最終的に繋ぐことができた総チップ数とする．したがって，図6.7の例の場合は$f = 12$となる．

6.3.3　計算機実験

ウイルス進化型アルゴリズムと，遺伝的アルゴリズムとの比較をするため計算機実験を行なった．この節では実験における幾つかの条件および，それらの条件下での実験結果について述べる．

6.3.3.1　実験条件

VEAおよびGAの諸設計については次のように定める．

まず，VEA，GA両者に共通する実験条件は次の通りである．

初期集団の生成：決められた個体数の染色体M個をランダムに生成する．

次に，VEAの実験条件は次の通りである．

選択: 最大適応度の染色体を選択する．またエリート保存戦略を用い，エリート保存数は5とする．

感染: 感染率はp_iとする．

突然変異: 突然変異率はp_{mv}とする．

最後に，GAの実験条件は次の通りである．

選択: ルーレット戦略とエリート保存戦略を用い，エリート保存数は5とする．

交叉: 1点交叉を用い，交叉率はp_cとする．

突然変異: 突然変異率はp_{mg}とする．

以上の条件下で，VEAとGAの比較実験を行う．

6.3.3.2　実験結果と考察

実験におけるパラメータは表6.3に示す値を用いた．実験結果を表6.4に示す．実験は各アルゴリズムに対し５つのモデルでそれぞれ10回行った．表6.4に示す各モデルごとの結果は，10回の実験で得られた最も優れた解の適応度である．実験モデルには図6.8に示すような，10×10に並べられたチップの中に25個の欠陥チップがあるものを用いた．なお，図6.8に示したものはモデル１のものである．表6.2中で，f_v，f_g，はそれぞれのアルゴリズムで得られた最大適応度（再構成された最大チップ数）であり，f_oはそのモデルにおいて再構成可能な最大チップ数，すなわち，最適解の適応度である．計算時間はHP-9000/755ワークステーションで要した時間である．

表6.2からわかるように，VEAの計算時間はGAのものよりも短く，その減少率は

$$\frac{208.4 - 70.4}{208.4} \times 100 = 66.2\ (\%).$$

となり，VEAの有用性が認められる．また，VEAの最適解に対する適応度の比の平均は0.85であり，GAの場合は0.73である．これはVEAの方がGAよりも探索能力が高く，より最適解に近づいていることを示しているが，最適解を得ることはできなかった．なお，図6.9と図6.10は，モデル１においてVEAとGAで得られた再構成されたチップ数が最大のものである．さらに図6.11に示す，モデル１のサイクル，世代数と適応度の関係からも，VEAの方が収束が速く，探索能力も高いことが確認できる．

表 6.3　計算機実験パラメータ

共通		
集団サイズ	M	300
染色体長	N	100
ウイルス進化型アルゴリズム		
サイクル数	T_c	500
感染率	p_i	0.95
突然変異率	p_{mv}	0.1
遺伝的アルゴリズム		
世代数	T_g	500
交叉率	p_c	0.95
突然変異率	p_{mg}	0.05

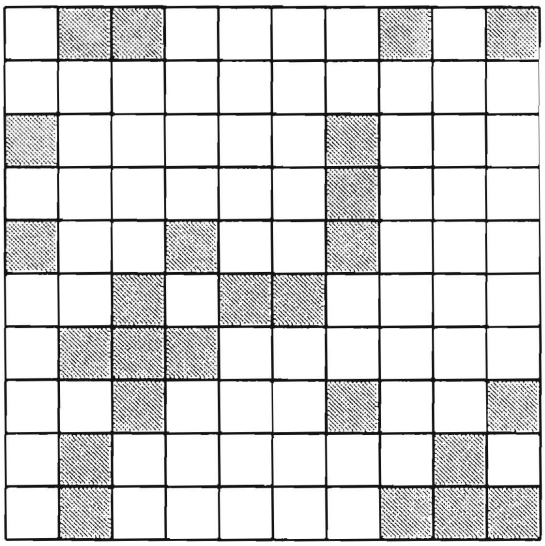
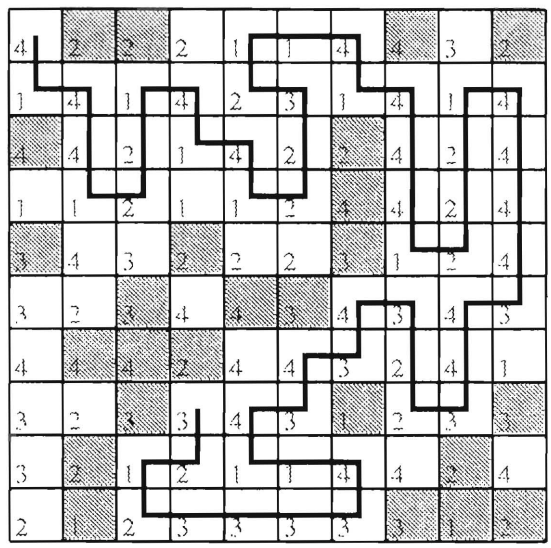


図 6.8　実験モデルの例（モデル 1）

表 6.4　実験結果

	アルゴリズム				
	ウイルス進化型		遺伝的		最適解
	適応度 (f_v)	f_v/f_o	適応度 (f_g)	f_g/f_o	適応度 (f_o)
モデル1	55	0.89	46	0.74	62
モデル2	53	0.82	43	0.66	65
モデル3	52	0.87	47	0.78	60
モデル4	52	0.84	45	0.73	62
モデル5	54	0.83	47	0.72	65
平均	–	0.85	–	0.73	–
平均計算時間 (s)	70.4		208.4		–



55

図 6.9　VEA の実験結果の例（モデル 1）

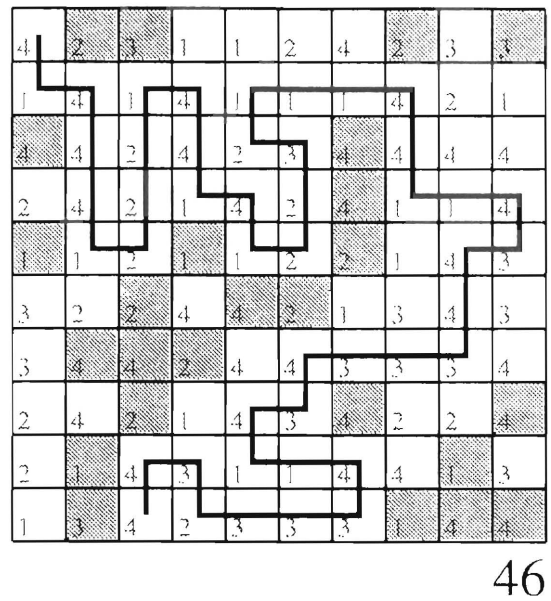


図 6.10　GA の実験結果の例（モデル１）

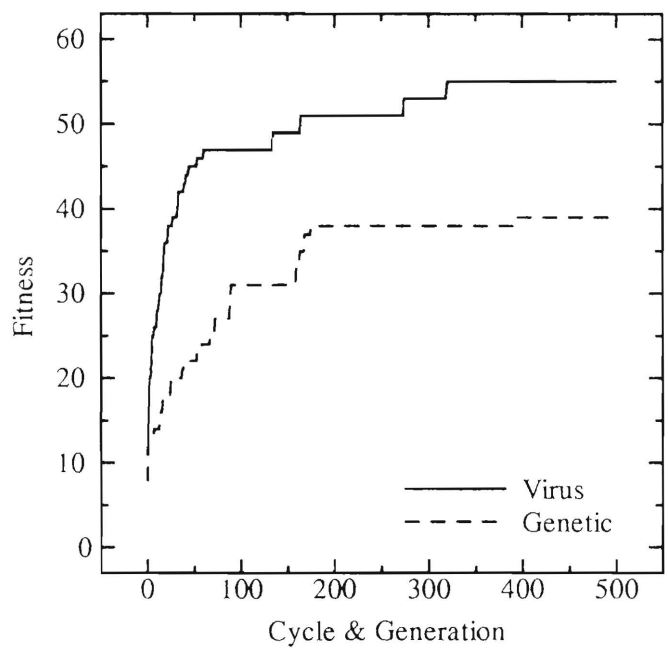


図 6.11　サイクル，世代数と適応度の関係

6.3.4　まとめ

この節ではWSIのためのウイルス進化型アルゴリズムを用いた再構成手法の提案と，計算機実験によりウイルス進化型アルゴリズムと，遺伝的アルゴリズムとの比較を行った．その結果，遺伝的アルゴリズムに比べ，ウイルス進化型アルゴリズムでは次のことが確認された．

1. 計算時間が減少する．
2. 収束速度が上昇する．

これは第5章の結果を指示するものである．ただし，本研究では最適解を得ることはできなかった．この原因はコーディング法にあると思われ，ウイルス進化型アルゴリズムにおいて，“最適解を得る確率が高くなる”ことを否定するものではないと考える．

6.4　むすび

本章では，“ウイルス進化型アルゴリズム”を用いた“基本的な配置問題”のための配置手法，および“WSIの再構成問題”のための再構成手法を提案した．これらの手法は，ウイルス進化型アルゴリズムを用いることで遺伝的アルゴリズムに比べ，以下の点で優れていることが計算機実験により確認された．

1. 計算時間が減少する．
2. 収束速度が上昇する．
3. 最適配置を得る確率が高くなる．

なお，これは第5章の結果を指示するものであり，ウイルス進化型アルゴリズムが，実用的な問題にも利用可能なことを示している．

第 7 章

結論

7.1 本研究の成果

本論文では，従来手法では難しかった凹部を含むブロックの配置や，配置領域を制限しての配置を可能にする任意形状ブロックの配置手法を提案した．提案手法は，ブロックの形状に応じた評価値を定義し，その値を用いることで任意形状ブロックを配置する組立式配置手法と，その配置順序に遺伝的アルゴリズムによる制御を組み合わせたものであり，計算機実験により提案手法の有用性も確認された．

また，ウイルス感染による進化という点に着目して，ウイルス進化論に基づく新しい進化型アルゴリズム，ウイルス進化型アルゴリズムを提案した．計算機実験により提案手法と遺伝的アルゴリズムとを比較した結果，提案手法は計算時間が短かく，収束も速く，また，最適解を得る確率も高いということが確認された．

第１章では本研究の社会的背景と本研究の目的について述べた．また，各章ごとにまとめた論文の概要も示した．

第２章では本研究の対象であるLSIレイアウト設計の現状について述べた．なお，本論文で提案した任意形状ブロックの一配置手法は，セミカスタム・スタンダードセル方式の中の，ビルディングブロック設計方式を対象とした配置およびフロアプラン手法である．

また第３章では遺伝的アルゴリズムを用いた配置手法ということで基本的配置問題を対象に遺伝的アルゴリズムの適用を試みた．しかし，単純にGAを適用したのでは探索効率が低く局所解に陥りやすい．あるいはランダムサーチ的な振る舞いをするという問題があった．そこで本研究では探索効率の低下を回避する，

次の２つの新しいコーディング法を提案した.

1. 配置位置を否定する情報を持たせたコーディング
2. 配置位置情報に冗長性を持たせたコーディング

提案手法の有用性を確認するため，従来手法との比較を目的に計算機実験を行った．“否定情報を持たせたコーディング法の場合”は配置不能な解の生成こそは抑制可能だが，実用に足る結果を得ることはできなかった．“情報に冗長性を持たせたコーディング法の場合”は従来手法の問題点であった配置不能な解の生成と，ランダムサーチ的な振る舞いを共に抑制することにより，従来よりも優れた探索結果を得ることが可能なことが確認された.

第４章ではVLSIレイアウトにおける組立式配置手法として，ブロック形状に応じた評価値を定義することにより，任意形状ブロックの配置を行う一手法を提案した．本手法は以下のような特長を持つことが計算機実験により確認された.

1. 従来手法では難しかった凹部を含む，任意形状ブロックの配置が可能である.
2. 非矩形の領域制限を含む，配置領域を制限しての配置が可能である.
3. 配置順序に遺伝的アルゴリズムによる制御を組み合わせることで，組立式手法に共通する欠点である配置結果の順序依存問題に対応できる.
4. 従来のコンパクション手法のための初期配置を与えることに利用可能である.

そして第５章ではウイルス進化論に基づく新しい進化型アルゴリズムを提案した．このアルゴリズムではウイルスを遺伝子を運ぶ生物器官の一つと捉え，ウイルスの感染による遺伝子操作を，GAと組み合わせて探索能力の向上を目的とするものではなく，唯一の探索操作とした点が，GAや従来のウイルス進化論に基づく手法とは大きく異なっている．また，ウイルスの感染により集団全体へ遺伝子断片を急速に拡散可能であり，また遺伝子操作の処理の軽さから，GAに比べ以下のような特長を持つことが計算機実験により確認された.

1. 計算時間が減少する.
2. 収束速度が上昇する.
3. 最適解を得る確率が高くなる.

第６章では，ウイルス進化型アルゴリズムの基本的な配置問題への適用と，WSIの再構成問題への適用を行った．その結果，これらの応用問題においてもウイルス進化型アルゴリズムは遺伝的アルゴリズムに比べ，以下の点で優れていることが計算機実験により確認された.

1. 計算時間が減少する.
2. 収束速度が上昇する.

3. 最適配置を得る確率が高くなる.

最後に，第 7 章では本研究の総括と今後の展望を述べている.

7.2　今後の展望と課題

LSI レイアウト設計における CAD システムによる自動化は，今後ますます必要性を増すものと思われる．そういった中で本研究で扱った任意形状を対象とした配置手法も，重要な意味を持つものと予測される．今後は形状の自由度をさらにあげて，長さ方向にも十分な自由度を持つ手法の構築を目指す.

今回提案したウイルス進化型アルゴリズムは，いくつかの応用例でも良好な結果を残しており，これは，他の問題においても結果を期待させるものである．今後は，より多くの多種にわたる問題への適用を試みたい．なお現在は，任意形状ブロックの配置手法への適用を進めている．また，提案した手法は極めて単純な遺伝子操作しか行っていないが，遺伝的アルゴリズムにも多くの遺伝子操作が提案されているように，提案手法にも感染などの操作の工夫で改良の余地があるものと考えており，研究を進めている.

謝　辞

本研究の全過程を通じて，懇切なる御指導と御鞭撻を賜わりました埼玉大学工学部情報システム工学科 近藤邦雄助教授に心から感謝の意を表します．

また本論文をまとめるにあたり，貴重な御指導および御助言を頂きました埼玉大学工学部情報システム工学科 前川仁教授，多くの御助言を頂きました埼玉大学工学部情報システム工学科 三島健稔教授，電気電子システム工学科 大嶋健司教授，伊藤和人助教授，金杉昭徳助手に厚く御礼申し上げます．

参考文献

[1] 渡辺孝博: “ＬＳＩレイアウト自動設計の現状と可能性”, 信学誌, Vol.76, No.7, pp.774–782, 1993.

[2] 北野宏明: “遺伝的アルゴリズム”, 産業図書, 1993.

[3] 北野宏明: “遺伝的アルゴリズム２”, 産業図書, 1995.

[4] D. Lawrence: “Handbook of Genetic Algorithms,” Van Nostrand Reinhold, 1991.

[5] 玉置　久, 喜多　一, 岩本貴司, 三宮信夫: “遺伝アルゴリズム－１，２，３”, システム制御情報学会誌, Vol.39, No.6,8,10, 1995.

[6] J. P. Cohoon and W. D. Paris: “Genetic Placement,” IEEE Trans. CAD, Vol.6, No.6, pp.956–964, 1987.

[7] K. Shahookar and P. Mazumder: “A Genetic Approach to Standard Cell Placement Using Meta-Genetic Parameter Optimization,” IEEE Trans. CAD, Vol.9, No.5, pp.500–511, 1990.

[8] J. P. Cohoon, S. U. Hegde, W. N. Martin, and D. S. Richards: “Distributed Genetic Algorithms for the Floorplan Design Problem,” IEEE Trans. CAD, Vol.10, No.4, pp.483–492, 1991.

[9] Y. Inoue, T. Shimamoto and A. Sakamoto: “A Module Placement Using Genetic Algorithms,” Trans. IEICE, Vol.J77-A, 8, pp.1189–1191, 1994.

[10] Wong D.F. and Liu C.L.: “Floorplan Design for Rectangular and L-shaped Modules,” Proc. Intl. Conf. on Computer-Aided-Design, pp.520–523, 1987.

[11] 大村道郎, 宮尾淳一, 菊野　亨, 吉田典可: “ＶＬＳＩのブロック配置における重なり除去問題”, 信学論（Ａ）, Vol.J72-A, No.7, pp.1093–1100, 1989.

[12] Hudson J.A., Wisniewski J.A. and Peters R.C.: “Module Positioning Algorithms for Rectilinear Macrocell Assemblies,” 21st Design Automation Conference, pp.672–675, 1984.

[13] 佐藤真司, 堤　定雄, 後藤源助: “マスタスライスセルの自動発生”, 信学技報, ICD88-70, 1988.

[14] Ｊ.Ｄ.ビースリー: “ゲームと競技の数学”, サイエンス社, 1992.

[15] 中原英臣, 佐川峻: “ウイルス進化論”, 泰流社, 1989.

[16] 狩野均, 長谷川和代, 松本美幸, 西原清一: “ウイルス進化論に基づく制約充足問題の解法”, 計測制御学会第30回知能システムシンポジウム, pp.75–80, 1996.

[17] N. Kubota, T. Fukuda and K. Shimojima: “Virus-Evolutionary Genetic Algorithm for Self-Organizing Manufacturing System,” Computer & Industrial Engineering Journal, Vol.30, No.4, pp.1015–1026, 1996.

[18] 下島康嗣, 久保田直行, 福田敏男: “ウイルス進化論に基づく学習型ファジィコントローラ”, 日本機会学会論文集（Ｃ編）, 63-608, pp.1261–1268, 1997.

[19] Grefenstette, J., Gopal, R., Rosmaita, B. and VanGucht, D.: “Genetic Algorithms for the Traveling Salesman Problem,” Proc. of 1st Int. Conf. on Genetic Algorithms and Their Applications, pp.160–168, 1985.

[20] 山下公一, 金杉昭徳, 土屋真平, 後藤源助: “ウェーハ・スケールLSIの可能性と限界”, 日経エレクトロニクス, Vol.6-1, No.422, pp.141–161, 1987.

[21] 金杉昭徳, 飯島大輔: “遺伝的アルゴリズムを用いたWSIの再構成手法”, 回路実装学会研究報告（実装CAE研究会）, CAE90-6, 1997.

[22] 府川典文, 金杉昭徳: “遺伝的アルゴリズムを用いたウエーハ集積回路の再構成手法”, 情報処理学会研究報告（設計自動化）DA88-2, pp.9–14, 1998.

[23] R. E. Massara ed.: “Design & test techniques for VLSI & WSI circuits,” IEE Computing Series 15, Peter Peregrinus Ltd., 1989.

[24] 中谷直司, 金杉昭徳, 近藤邦雄: “任意形状ブロックを対象とした一配置手法”, エレクトロニクス実装学会, Vol.1, No.6, pp.476-482, 1998.

[25] N.Nakaya, A.Kanasugi, H.Shindo and M.Morisue: “A Genetic Approach to Placement Using a New Coding Technique”, Proc. of ITC-CSCC'96, pp.1003–1006, 1996.

[26] N.Nakaya and K.Kondo: “A Novel Coding Technique for Genetic Placement Method”, Proc. of ITC-CSCC'97, pp.661–664, 1997.

[27] N. Nakaya, A. Kanasugi and K. Kondo: “A Novel Genetic Algorithm Based on the Theory of Virus Evolution”, Proc. of the Third International Symposium on Artificial Life and Robotics, pp.293–296, 1998.

[28] N.Nakaya, A.Kanasugi and K. Kondo: “A Novel Placement Method Using Evolutionary Algorrithm”, Proc. of ITC-CSCC'98, pp.1405–1408, 1998.

[29] N.Nakaya, A.Kanasugi and K. Kondo: “A Reconfiguration Method of WSI circuits using Evolutionary Algorithm”, Proc. of 5th Int. Conf. on Soft Computing and Information / Intelligent Systems IIZUKA'98, pp.845–848, 1998.

[30] 中谷直司, 金杉昭徳, 森末道忠: “任意形状ブロックのレイアウト手法”, 情報処理学会研究報告（設計自動化）DA80-5, pp.29–36, 1996.

[31] 中谷直司, 金杉昭徳, 進藤裕志, 森末道忠: “新しいコーディング法を用いた遺伝的配置手法”, 電子情報通信学会技術報告（VLSI設計技術）VLD96-66, pp.9–16, 1996.

[32] 中谷直司, 近藤邦雄: “ウイルス進化論に基づく新しい遺伝的アルゴリズム”, 情報処理学会研究報告（数値モデル化と問題解決）MPS19-6, pp.31–36, 1998.

[33] 中谷直司, 金杉昭徳, 森末道忠: “任意形状ブロックの一配置手法”, プリント回路学会第８回学術講演大会講演論文集, pp. 85–86, 1994.

[34] 中谷直司, 金杉昭徳, 進藤裕志, 森末道忠: “遺伝的アルゴリズムを用いた配置手法における新しいコーディング法”, 情報処理学会第54回全国大会（平成９年前期）講演論文集（１）, pp.159–160, 1997.

本研究に関する発表文献

本研究に関する論文・国際会議

[1] 中谷直司, 金杉昭徳, 近藤邦雄: “任意形状ブロックを対象とした一配置手法”, エレクトロニクス実装学会, Vol.1, No.6, pp.476-482, 1998.

[2] 中谷直司, 金杉昭徳, 近藤邦雄: “ウイルス進化論に基づく進化型アルゴリズム”, 情報処理学会, 採録決定.

[3] N.Nakaya, A.Kanasugi, H.Shindo and M.Morisue: “A Genetic Approach to Placement Using a New Coding Technique”, Proc. of ITC-CSCC'96, pp.1003-1006, 1996.

[4] N.Nakaya and K.Kondo: “A Novel Coding Technique for Genetic Placement Method”, Proc. of ITC-CSCC'97, pp.661-664, 1997.

[5] N. Nakaya, A. Kanasugi and K. Kondo: “A Novel Genetic Algorithm Based on the Theory of Virus Evolution”, Proc. of the Third International Symposium on Artificial Life and Robotics, pp.293-296, 1998.

[6] N.Nakaya, A.Kanasugi and K. Kondo: “A Novel Placement Method Using Evolutionary Algorrithm”, Proc. of ITC-CSCC'98, pp.1405-1408, 1998.

[7] N.Nakaya, A.Kanasugi and K. Kondo: “A Reconfiguration Method of WSI circuits using Evolutionary Algorithm”, Proc. of 5th Int. Conf. on Soft Computing and Information / Intelligent Systems IIZUKA'98, pp.845-848, 1998.

本研究に関する学会研究会資料

[1] 中谷直司, 金杉昭徳, 森末道忠: “任意形状ブロックのレイアウト手法”, 情報処理学会研究報告（設計自動化）DA80-5, pp.29-36, 1996.

[2] 進藤裕志, 金杉昭徳, 中谷直司, 森末道忠: “ネット形状の保存を考慮した遺伝的配置手法”, 電子情報通信学会技術報告（VLSI設計技術）VLD96-29, pp.25-30, 1996.

[3] 中谷直司, 金杉昭徳, 進藤裕志, 森末道忠: “新しいコーディング法を用いた遺伝的配置手法”, 電子情報通信学会技術報告（VLSI設計技術）VLD96-66, pp.9-16, 1996.

[4] 中谷直司, 近藤邦雄: “ウイルス進化論に基づく新しい遺伝的アルゴリズム”, 情報処理学会研究報告（数値モデル化と問題解決）MPS19-6, pp.31-36, 1998.

本研究に関する口頭発表

[1] 中谷直司, 金杉昭徳, 森末道忠: “任意形状ブロックの一配置手法”, プリント回路学会第８回学術講演大会講演論文集, pp. 85-86, 1994.

[2] 中谷直司, 金杉昭徳, 進藤裕志, 森末道忠: “遺伝的アルゴリズムを用いた配置手法における新しいコーディング法”, 情報処理学会第54回全国大会（平成９年前期）講演論文集（１）, pp.159-160, 1997.