メタヒューリスティック手法を用いた
最適配置問題の解法に関する研究

1999年 7月

高 原 茂 幸

目　次

第１章

序論

最適配置問題として，ある部材を有効に利用するために，取りたい部品をどのようにその部材上に配置すればよいのかを考える問題がある．この最適配置問題では，取りたい部品として様々な形状の部品が考えられるため，その組み合わせ方法が非常に難しい．また，部品数が多くなると組み合わせの数が爆発的に増えるため，解くことが非常に困難な問題である．このような問題は，電子部品の配置であるとか板金加工や裁断の工程など様々な産業分野において現れ，一般的にネスティング問題，もしくはレイアウト問題と呼ばれている．この問題のイメージを簡単に**Fig.1.1**に示す．この配置問題（packing problem）[1]は，スケジューリング問題や巡回セールスマン問題に代表されるような組合せ最適化問題[2]の一種であり，厳密な最適解を求めることが非常に困難であるNP-完全（NP-Complete）であることが知られている[3],[4]．特にこの最適配置問題では，問題の定式化の方法自体が一意的ではなく，厳密な最適解はその問題の表現形式に依存する．そのため，厳密な最適解がどのようなものであるのかがはっきりせず，たとえあったとしても，それが一意であるとは限らない．よって，ある最適解が得られたとしても，それが真の最適解であるのかどうかについては分からない．これまでに，この問題に対して，２次元矩形の配置（bin packing）問題に特化した形で，線形計画法（linear programming）や動的計画法（dynamic programming）などを用いて様々な検討が行われてきた[5]-[9]．これらの手法が対象としているのは，原材料切断の問題(cutting stock problem)において，比較的簡単な２次元矩形形状を扱う場合に限られていた．しかし一般的には，配置対象となるのはそのような簡単な形状の部品ばかりではない．そのため，矩形以外の部品も対象とした最適配置問題に取り組んでいる報告もあるが，扱える形状に制約があるなど問題点も

ある[10)-13)].

現場の立場からすれば，このような最適配置問題に対して厳密な最適解が得られるのに越したことはないが，納期など時間的な制約もあり，ある程度満足な解が得られればそれで十分メリットがある場合もある．また，部材が高額であるため，時間をかけても厳密な最適解に近い解が欲しい場合もある．このように，状況に応じて様々な対応が迫られる．このように対応が困難な問題に対して，これまではエキスパートと呼ばれる人達が経験によって柔軟な対応をしてきていた．そのことから，このようなエキスパートの人達の知識（ヒューリスティックス）をルールベースとして用いたシステムの開発がこれまでに試みられたりもしている[14)-16)].

また最近では，計算機の処理速度の急速な進歩により，計算能力に余裕がなく解くことが諦められていたような大規模かつ複雑な問題に対しても計算機を利用したアプローチが可能になってきた．特に近似最適解が得られれば十分であるような問題に対しては，メタヒューリスティックスと呼ばれる手法が有効であり，様々な問題に適用されている．このメタヒューリスティックスとして代表的な手法は，局所探索，シミュレーテッド・アニーリング，タブー探索，遺伝アルゴリズムなどが上げられる[17)-19)]．本研究で扱う最適配置問題は，このような大規模かつ複雑な問題の一種であり，メタヒューリスティック手法を用いた解法も報告されている[20)-22)].

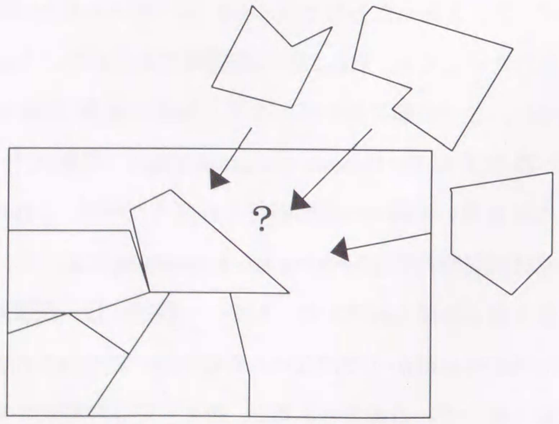


Fig.1.1　Optimal Allocation Problem

このようなヒューリスティックスやメタヒューリスティックスを用いたアプローチを取っているものとして次のようなものが上げられる．その１つの方法として，Albanoら[14)]やCaiら[16)]が取っている方法がある．これはヒューリスティック手法を用いたもので，先にも述べたエキスパートの知識ベースを用いて任意形状部品の配置まで考慮に入れた方法である．この手法では，配置位置と配置順序という捉え方ではなく，部品の組み合わせのパターンを経験的なヒューリスティックスを用いて決定するようになる．よって，ユーザーの仕様に応じて知識ベースを更新する必要があるため，問題への依存度が高くなり，問題によって得られる解の良否が分かれるという問題がある．

さらにもう１つ別のアプローチ方法として，藤田ら[22)]が取っている方法がある．この方法では，配置する部品の代表点座標や代表軸を設計変数として，この設計変数をメタヒューリスティック手法によっていろいろ変えることで，最適な配置を求めるというものである．この手法では，部品配置の組み合わせをメタヒューリスティック手法によって決定することになる．最初にある配置パターンが与えられており，そこから部品を揺らして動かしながら最適な配置を探索するようなイメージである．この手法では，あらかじめ配置パターンを与えるため，探索よりもその配置パターンへの依存度が大きくなることと，配置パターンとして様々な組み合わせを試すことができないアプローチ方法となっていることが問題である．また，配置する部品数が多くなれば，最適化を行うための設計変数が多くなり，効率的な方法とは言えない．

このようなアプローチでは，知識をまとめたルールベースの作成が困難であり，なおかつ問題に特化したシステムとなってしまったり，最適化を行うためのパラメータが多くなり，問題へのコーディングが非常に難しくなったりした．よって，簡単に問題に対するコーディングが行え，なおかつ様々なタイプの問題に対応できるような最適化の手法が望まれている．

そこで本論文では，この最適配置問題を次のような２つの部分問題に分解して考察する．

(1) 部品を配置する順序をどのように決定すればよいのか．

(2) 部品を配置する候補位置を選択するアルゴリズムをどのようにするのか.

これと同様のアプローチを取っているものとして, Smith[20]やKrögerら[21]がある. これらの手法では, (1)の配置順序の決定方法として遺伝アルゴリズムを適用しているだけである. また(2)の配置位置選択のアルゴリズムにおいて, 配置する部品を矩形に限定している. よって, メタヒューリスティック手法の中でどのような手法が有効であるかは検討されていない点と, 配置対象も矩形に限定されているため実用的に満足できるものではない点が問題として上げられる. しかし, このように問題を配置位置決定の問題と配置順序決定の問題というように2段階構造とすることで, 問題の見通しが良くなり, それぞれ独立に考えることで改善も行いやすく, 最適化の相乗効果が期待できるというメリットがある. よって, ユーザーの仕様が変わっても配置順序を決定するときに評価を行う目的関数を変えるだけで対応可能である. また, メタヒューリスティック手法で最適化を行う設計変数として部品の配置順序を用いることで, 問題に対してのコーディングも行い易くなる.

そこで本論文では, 最適配置問題に対して有効な解を与えるための手法として様々なメタヒューリスティック手法を取り上げ, 配置対象が柔軟に選択可能である効果的

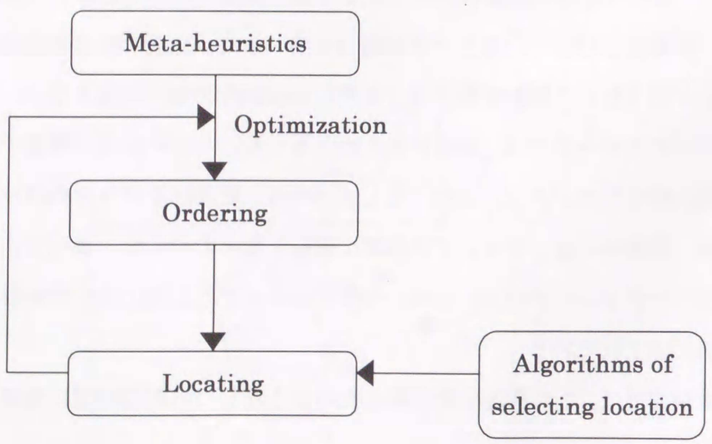


Fig.1.2 An approach to the optimal allocation problem

かつ実用的な解探索の手法を構築することを目的とする. ここでは, (1)の配置順序の決定に各種メタヒューリスティック手法を適用する. さらに, 問題に依存しない適応的な手法を提案する. そして, (2)の配置位置選択のアルゴリズムとして, 任意多角形状部品の配置までを考慮に入れたアルゴリズムを提案する. ここでは, 配置順序により配置が決定できるようなアルゴリズムを提案する. 本論文における, この問題に対するアプローチをFig.1.2に示す.

第2章では, 最適配置問題の解法として, (2)の配置位置選択のアルゴリズムについて説明する. ここでは, 配置順序から部品の配置が決定できるようなアルゴリズムを提案する. まず, 最適配置問題の記述を行い, この問題の定式化を行う. そして, 矩形部品を配置する場合の位置選択のアルゴリズムとして, 稜線法(ridgeline method)を提案する. この手法の有効性を示すために, シミュレーションによりこれまで提案されていた手法との比較を行う. また, 配置する部品数によって, その配置効率にどのような影響があるのかについても検証する. 次に, この稜線法を拡張した形で, 配置対象として任意多角形状部品にまで対応できる配置位置選択のアルゴリズムについて説明する. そして稜線法の場合と同様に, シミュレーションによりその効果を検証する. 配置する部品の形状によって配置効率にどのような影響があるのか, また配置する部品数によってどのような影響があるのかについて検証する.

第3章では, (1)の配置順序を探索する問題に対して, 各種メタヒューリスティック手法を適用する. これまでに, 局所探索(local search), シミュレーテッド・アニーリング(simulated annealing), タブー探索(tabu search), 遺伝アルゴリズム(genetic algorithm)など様々なメタヒューリスティック手法が提案されているが, これらを比較することで, この最適配置問題に適した手法を見いだすことを目的とする. 一般的に組合せ最適化問題では, シミュレーテッド・アニーリング, タブー探索, 遺伝アルゴリズムがよく用いられ, その有効性が報告されている[18]. ここで取り上げる代表的な4つのメタヒューリスティック手法の中で遺伝アルゴリズムと他の3つの手法とでは少し枠組みが違う. そこで, これらを2つのグループに分け, それぞれ探索に重要な役割を果たしていると思われる項目に着目して比較を行う. 遺伝アルゴリズムに

関しては，集団を進化させるための手続きについて，また他の３つの手法に関しては，解近傍の定義についてその効果を検証する．また，最初に２つの問題として分けた(1)の配置順序を探索する問題と(2)の部品を配置する位置選択の問題との関係についても考察を行う．ここでは，部分アルゴリズムである配置位置選択のアルゴリズムを変えた場合に，メタヒューリスティック手法を用いた最適解の探索にどのような影響があるのかについて調べる．

第４章では，第３章の結果から，本論文で扱う最適配置問題に最も適した手法であると考えられるシミュレーテッド・アニーリングを取り上げ，この手法をベースとしてさらに問題に対する適応性を持たせた適応型シミュレーテッド・アニーリング(adaptive simulated annealing)を提案する．従来のシミュレーテッド・アニーリングでは探索ステップ数が増えると探索効率が低下するという問題がある．しかしこの手法では，探索戦略に適応性を持たせることで，このような問題についても解決できる．ここでは，適応度評価テーブル(evaluation table)を利用して過去の探索履歴を保持し，その履歴を後の探索で活用することで，適応的な探索を実現する．つまり，学習のスキームを探索の中に取り入れることにより，探索の効率を上げ，さらに様々な問題に対して適応的に解決策を見出すような探索を実現することが目的である．この手法についてアルゴリズム等の説明を行い，またその効果をシミュレーションによって示す．

第５章では，組合せ最適化問題で評価の高いタブー探索を取り上げ，この手法に適応性を持たせた適応的タブー探索(adaptive tabu search)を提案する．タブー探索は非常に様々なオプションが利用できることが特徴である．一般的に，多様なメモリーを用いることで探索効率を上げようとする手法であるが，これが逆に問題であり，問題に対するコーディングを難しくしている．そこで，非常にシンプルなメモリーを用いることで，これまでタブー探索の特徴であるとされてきた柔軟性に富んだ探索が実現できることを示す．ここではそのメモリーの働きをするものとして適応度評価メモリー(adaptive memories)を導入する．このメモリーを用いた手法について説明し，またその効果をシミュレーションによって示す．ここでは，その効果を見るために，

適応度評価メモリーを適応型シミュレーテッド・アニーリングの適応度評価テーブルの代わりに用いた場合の評価や局所探索に適用した場合の評価も行う．また，タブーリストによる効果も併せて検証する．

第６章では，本論文のまとめと今後の課題について述べる．

第２章

最適配置問題の解法

2.1　最適配置問題の記述

ここではまず，最適配置問題の定式化を行う．ここで配置の対象とする部品は，n個の部品$(b_1,...,b_n)$であるとし，この部品に対してそれぞれ整数1,2,.....,nを対応づける．つまり，整数$k(k=1,.....,n)$がある部品b_kを表すことになる．そうすると，それらの順列の集合として，

$$\rho=\left\{\sigma=\left(i_1,i_2,.....,i_n\right)\right\} \tag{2.1}$$

が考えられる．この順列$\sigma\in\rho$によって決定される順序を元にし，あるアルゴリズムに従って部品を配置していくことで配置のパターンが決定される．つまり，配置位置を選択するアルゴリズムALによって解となる順列σの評価が行われる．このアルゴリズムを用いて決定される順列の評価値を$F(\sigma,AL)$とする．この順列を様々に変えることで，最適な配置を見つけようとするものである．よって，最適配置問題は，

$$\min_{\sigma\in\rho}F(\sigma,AL) \tag{2.2}$$

と記述される．この$F(\sigma,AL)$がこの問題の目的関数となる．またここでは，アルゴリズムALを明示する必要のないときには，単にこの目的関数を$F(\sigma)$と表す．

ここで，この配置位置を選択するアルゴリズムALについて考える．最適配置問題では様々なアルゴリズムが考えられるが，ここで考えるアルゴリズムは，配置順序によって逐次的に配置を決定するようなグリーディ法[23]である．そのため，このアルゴリズム自体で最適化を行う必要はなく，配置順序に対して一意に配置が決定できればよい．矩形部品を配置対象としたアルゴリズムとして，配置する部品の長さ，高さなどを入力情報として，その組合せを線形計画法や動的計画法によって最適化するアル

ゴリズムの開発がある[5]-[9]．これらは古くから行われており，そのアルゴリズム自体を用いて最適配置を求めようとするものであった．ここで考えているような配置順序によって配置位置を決定するようなアルゴリズムはあまり研究されていない．これまでの研究では，skyline pack法[20]が提案されているが，部品を配置する選択候補の位置が限定されているため，十分に満足できる解を得ることは難しい．さらに，矩形以外の形状部品を配置対象とした手法もこれまでに提案されてはいる[10]-[16]．しかし，それらは配置する形状がある特定のものに限定される手法であったり，ルールベースを用いる手法であったりし，それ自体で配置位置を求めようとするものであり，配置順序によって配置位置をコントロールするようなアルゴリズムは提案されていない．よって以下の節で，矩形部品の配置に限定したものとして，部品を配置する選択候補の位置を広げた稜線法[24]について説明を行う．また，この稜線法を拡張し，任意多角形状の部品の配置を可能にするアルゴリズム[25]についての説明も行う．

また，そのアルゴリズムを用いて配置された状態に対する評価を与える関数Fであるが，これは適用される問題によってその目的は様々であるため，いろいろと考えられる．例えば，部品を配置する部材がロール状のものであるのか，１枚の板状のものであるのかでも変わってくる．また，残材を活用しやすいように部品取りをするのか，カッティングが行いやすいように部品取りをするのかといったことによっても違ってくるであろう．ここで行うシミュレーションでは，部品を２次元矩形の１枚の板に配置するものとし，配置した結果，すべての部品を含む矩形領域の面積が最小となるようにすることを目的とする．よって，σに対してあるアルゴリズムALを適用したときに，すべての部品を含む最小の矩形領域の面積を$F(\sigma, AL)$として，σを変えることによって$F(\sigma, AL)$を最小化することになる．そしてシミュレーションによる評価は，用いる部品集合のデータが様々であることから，配置後の余材率$r(\sigma)$を用いて良い配置であるかどうかを判断する．ここで言う余材率は，

$$r(\sigma) = \left. F(\sigma, AL) - \sum_{k=1}^{n} S_k \right/ F(\sigma, AL) \tag{2.3}$$

と定義する．ここでS_kとは，整数kに対応する部品b_kの面積である．よって余材率と

は，すべての部品を含む最小の矩形領域に占める余材の比率である．この余材率が小さくなるように配置することが求められる．

その他に，部品を配置する際に部材のどこを基準点として配置するのかを決める必要がある．ここでは，その基準点Oを２次元矩形の１枚の板上にある左下の頂点とする．よって，この頂点を含む矩形領域を最小になるように部品配置を行う．

2.2 矩形部品の配置

矩形部品を配置する場合に限定した配置位置選択のアルゴリズムについて説明を行う．その新たな手法として，ここでは稜線法(ridgeline method)を提案する．そして，このアルゴリズムの有効性を示すために，実際に矩形部品を配置するシミュレーションを行う．そして，これまでに提案されているskyline pack法との比較を行い，稜線法の有効性についても示す．また，配置する部品数によって，結果にどのような影響があるのかについても考察する．

2.2.1 稜線法（ridgeline method）

既に配置されている部品群の稜線をたどることによって最適な配置位置を選択するアルゴリズムである稜線法について説明する．このアルゴリズムでは，最初にある配置順序$\sigma = (i_1, \ldots, i_n)$が決められ，その順序に従って配置を行う．$\sigma$の前から$j$個を取った部分順列を$\sigma[j] = (i_1, \ldots, i_j)$とすると，$j$番目の部品を配置するときにその評価値は$F(\sigma[j])$と記述できる．ここでは，配置する部品$i_j$に基準点を設けて，その基準点が稜線上をたどることによって，最適な配置位置を選択するものとする．ここでは，その基準点o_jを部品のある頂点とする．さらに，既に配置された部品群の稜線には，稜線の頂点にラベル（$A_{i,j}$, $A_{i,j}$など）を付け，稜線を表現する．この稜線は，部品を配置した形状によって，１本ではなく複数生じることが考えられる．よって頂点ラベル$A_{i,j}$は，i本目の稜線に含まれるj番目の頂点を表す．

ここで，稜線法のアルゴリズムを以下に示す．

Step0 配置順序$\sigma=(i_1,...,i_n)$を決定する．

Step1 部品を配置させる部材である矩形領域において，初期稜線として，$(A_{1,1},A_{1,2},A_{1,3})$を生成する．そして，$k=1$，最大稜線数$l=1$とする．

Step2 配置するk番目の部品b_{i_k}をすべての稜線上$((A_{1,1},....,A_{1,n_1}),...,(A_{l,1},.....,A_{l,n_l}))$で連続的に移動させたとき，$F(\sigma[k])$が最小となる位置を探索する．ここで配置する部品は矩形部品であるので．部品を縦方向に配置した場合と横方向に配置した場合の両方向を探索する．このときに，配置した部品b_{i_k}の基準点o_{i_k}の位置が稜線の頂点上（$A_{i,j}$など）にあればStep3に進み，稜線の頂点上以外にあればStep4に進む．

Step3 Step2で選択された位置に部品の配置を行う．そして，配置する位置として選択された位置が含まれる稜線の更新を行う．$k=n$であれば終了し，そうでなければ$k=k+1$としてStep2に戻る．

Step4 Step2で選択された位置に部品の配置を行う．そして，配置する位置として選択された位置が含まれる稜線の更新を行う．この場合，この位置に部品を配置することで必ず中空領域が生じる．そこで，$l=l+1$とし，この中空領域に新たな稜線$(A_{l,1},A_{l,2},A_{l,3})$を生成する．$k=n$であれば終了し，そうでなければ$k=k+1$としてStep2に戻る．

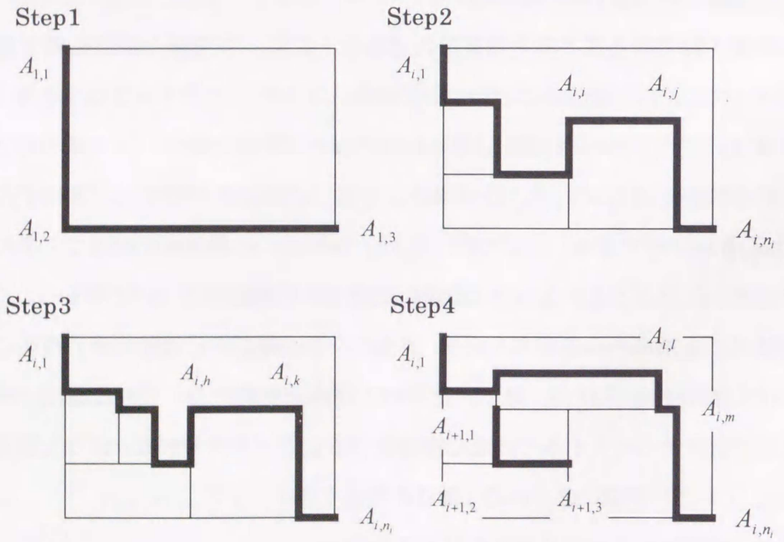


Fig.2.1 Illustration of ridgeline method

この簡単な流れを**Fig.2.1**に示す．ここで検討している部品は矩形部品であるので，配置位置に対して部品の配置パターンが２つ考えられる．Step2では，この２つのパターンを考慮する必要がある．また，このアルゴリズムを用いて配置を行う過程で，配置した部品が既に配置されている部品と重なったり，配置したい部材内に収まらなかったりして，その部品が配置不能である場合には，目的関数$F(\sigma[k])$は無限大の値を取るものとし，その順序列σは実行可能解ではないものとする．

ここで重要なのは，Step2において目的関数$F(\sigma[k])$が同じ値となる配置が複数個存在した場合にどうするのか，ということである．この選択の方法によっては，同じ配置順序であっても，最終的な配置がまったく違った配置となってしまう．ここでは，目的関数値が同じとなるそれぞれの配置で，その時配置された部品上で，部材の基準点Oから最もユークリッド距離の遠い位置の座標を$(p_i,q_i)(i=1,...,m)$としたときに，

$$\min_{i=1}^{m} p_k q_k \tag{2.4}$$

となる位置を選択することとした．また，Step4において中空領域が出現した場合の処理を加えていることが，skyline pack法との大きな違いである．

2.2.2 シミュレーション結果

稜線法を用いて，シミュレーションを行った結果例を**Fig.2.2**，**Fig.2.3**に示す．用

Table 2.1 Comparison of rectangular location algorithm

	Average[%]	Best result[%]
Skyline pack method	9.39	5.48
Ridgeline method	7.67	3.77

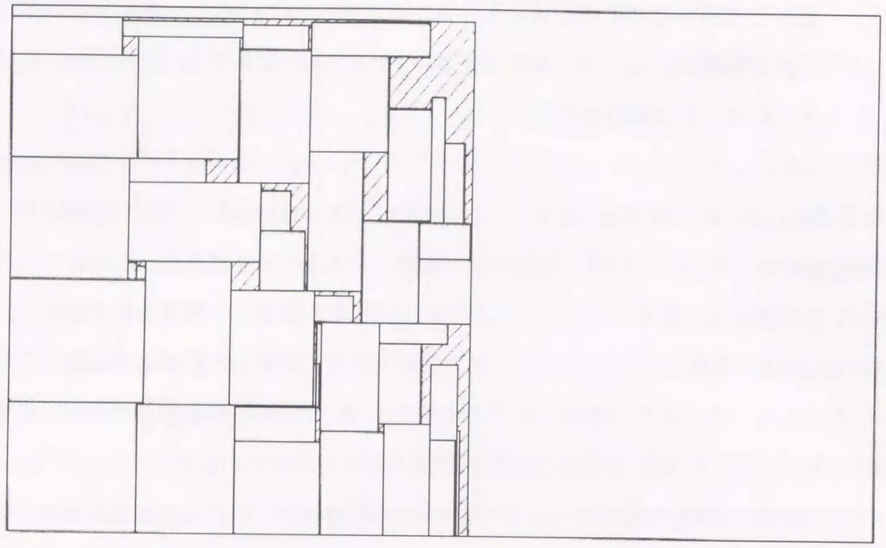


Fig.2.2 Layout example of rectangular parts (1)

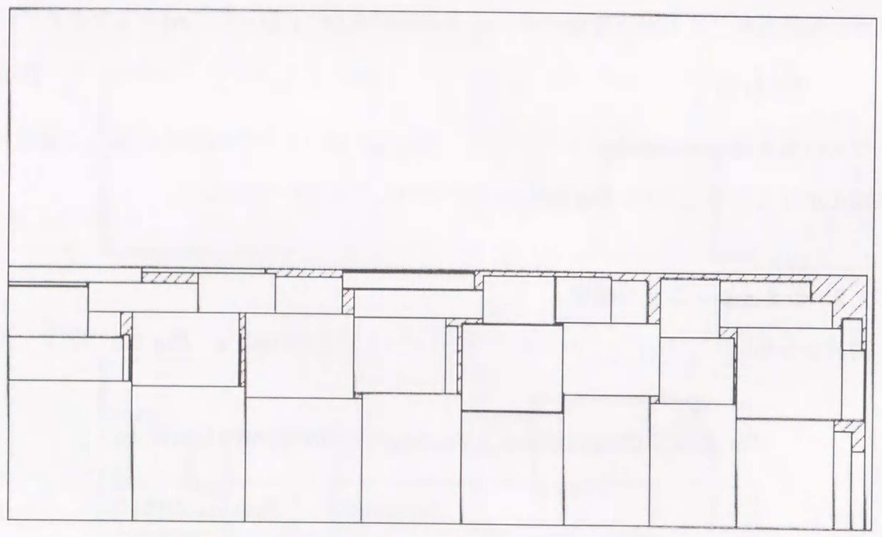


Fig.2.3 Layout example of rectangular parts (2)

いたデータは，部品数$n=40$である．また配置順序であるが，簡単なヒューリスティックスとして，部品の面積の大きい順にソーティングを行い，配置している．この例の場合，Fig.2.2の余材率は7.60%であり，Fig.2.3の余材率は4.34%であった．ここで行ったシミュレーションでは，どちらの方向を基準として配置を行うかという配置制約を加えていないので，最初に配置された部品の方向によってFig.2.3のように横に延びた形に配置される場合とFig.2.2のように縦に延びた形で配置される場合がある．ロール状のシートを部材として用いる場合には，ロールの長さが短くなるように部品を配置するため，そのための制約が付加される．

これと同様の部品数$n=40$のサンプルデータをランダムに100通り生成し，それを用いて面積の大きい順にソーティングを行い，稜線法及びskyline pack法で配置を行い，その結果の比較を行った．そこで得られた余材率の平均値と最良値を**Table 2.1**に示す．これより，平均値，最良値共に稜線法が優れていることが分かる．これは，中空領域の処理が効果的に働いていることを示している．

また，配置する部品の数によって，配置効率にどのような影響があるのかについ

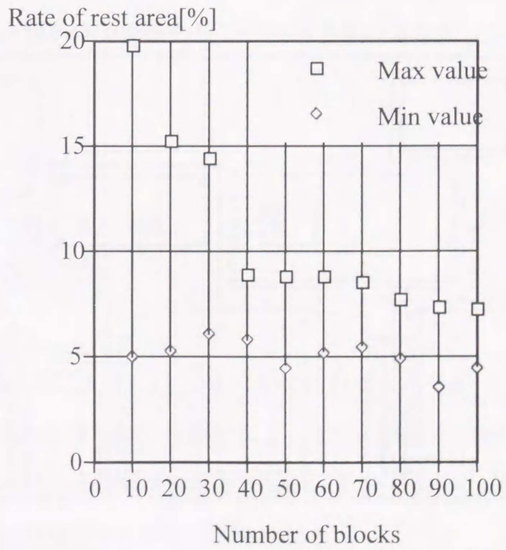


Fig.2.4 Effects of number of blocks

て検証する．ここでは，部品数$n=10$から100までのサンプルデータをそれぞれ10個用意し，稜線法を用いて配置を行った．そのときの余材率の最大値と最小値を**Fig.2.4**に示す．これによると，部品の数が少ない場合には，データによってうまく配置できる場合とできない場合の差がはっきり出てしまうが，部品数が多くなると，データによらずある一定のレベルにまで余材率を抑えることができることが分かる．ここでは，配置順序をただ面積の大きい部品から配置しているだけであるが，部品数が多い場合には，このような簡単な規則でも効果が得られることも分かる．

これらのことから，矩形部品を配置するためのアルゴリズムとして，稜線法は有効な手段であることが分かる．また，実際問題への適用例として，既にこのアルゴリズムを板金加工工程における展開部品のネスティングシステムに適用している．そのシステム上での配置例を**Fig.2.5**に示す．この場合は，配電盤や分電盤の板金加工部品の形状が矩形に近いものが多いということで，矩形近似することによって稜線法の適用が可能であった．しかし，屋根のあるような盤になると，矩形近似するのでは無駄が多くなるような展開部品も出てくる．よって，限定された形での利用しかできないことが問題である．

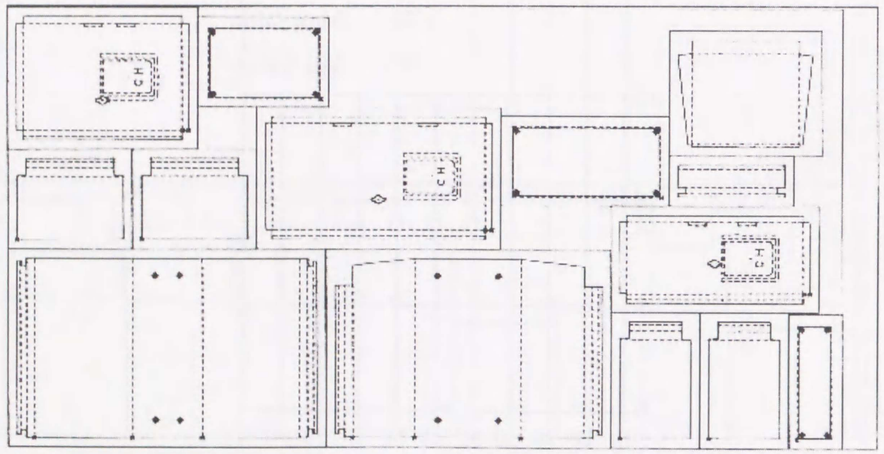


Fig.2.5 Application example to a sheet metal process

2.3 任意多角形状部品の配置

配置する部品が任意多角形状部品でも可能となる配置位置選択のアルゴリズムについて説明を行う．ここでは，任意の凸多角形状部品について考える．ここで考える手法も矩形の場合と同様に，部品を決められた配置順序$\sigma=(i_1,i_2,.....,i_n)$に従って逐次的に配置を行うある種のグリーディ法を用いる．ここで提案するアルゴリズムの基本的な考え方は，既に配置されている部品群の凹部分である頂点と，配置しようとする部品の凸部分の頂点を組み合わせられるような部分を探索することで，配置位置を決定しようとするものである．また，稜線法で用いたように，中空部分の処理も行う．そして，このアルゴリズムの有効性を示すために，実際に任意多角形状部品を配置するシミュレーションを行う．この手法を用いれば，任意形状部品についても多角形近似を行うことで配置を行うことが可能となる．

2.3.1 形状情報

部品を配置するにあたって，その形状の情報を定義しておく必要がある．ここでは，既に配置されている部品群の稜線の形状情報と配置しようとする部品外形の形状情報が必要である．部品群の稜線P_h及び部品外形線の頂点列Qを，それぞれ

$$P_h=\left(A_{h,1},A_{h,2},.....,A_{h,n_h}\right)$$

$$Q=\left(B_1,B_2,.....,B_m\right) \tag{2.5}$$

とすると，それぞれの頂点では，

$$A_{h,k}=\left(p_{h,k},\theta_{h,k}\right)$$

$$B_l=\left(p_l,\theta_l\right) \tag{2.6}$$

という情報を持つ．ここで，$p_{h,k}$，p_lは位置情報であり，$\theta_{h,k}$，θ_lは角度情報である．これにより，部品の配置状態が把握できる．また，稜線P_hの端点である$A_{h,1}$，A_{h,n_h}は，開放されているので角度情報は持たないものとする．さらに，部品の外形形状情報Qは，時計回りか反時計回りに閉じたパスである．

2.3.2 配置位置の探索

形状情報の角度情報をもとにして，既に配置された部品群の稜線 P_h の凹部と配置しようとする部品の外形線 Q 上にある凸部との組み合わせを求める．このとき配置しようとする部品が j 番目の部品であれば，その組み合わせの中で，目的関数 $F(\sigma[j])$ を最小とするような配置位置を選択する．このとき，組み合わされた凹部と凸部の形状情報が $A_{h,k}$，B_l であり，その角度情報がそれぞれ $\alpha_{h,k}$，β_l であるとすると，次の３つの場合が考えられる．

(1) $\alpha_{h,k} > \beta_l$

(2) $\alpha_{h,k} = \beta_l$

(3) $\alpha_{h,k} < \beta_l$

ここで，凹部と凸部をどのように組み合わせたときに目的関数を最小にできるかについての探索を行う必要がある．この手法でも矩形の場合と同様に，配置する部品 b_{i_j} に基準点 o_{i_j} を設ける．この点は，組み合わせようとする部品の頂点であるとする．この場合には組み合わされる頂点 p_l が基準点 o_{i_j} となる．よって，矩形の場合のように各部品に１つ持たせるのではなく，組み合わせを行う過程の中で基準点は変更される．探索手順としては，最初に部品群と部品との干渉が起こらないように位置決めを行う．この位置決めを行うときに，まず部品群の凹部の頂点 $p_{h,k}$ と部品の凸部の頂点 p_l が一致するように位置決めを行い，そこで干渉が起こらないようであればその一致した頂点を基点とする．(1)，(2)の場合には，一致させることができる可能性がある．干渉が起こるようであれば，基準点 o_{i_j} が部品群の稜線 $\overline{p_{h,k-1}p_{h,k}}$，$\overline{p_{h,k}p_{h,k+1}}$ 上をたどるように移動することで，干渉が起こらない位置を見つける．これを移動操作と呼ぶ．そして，この見つかった位置を基点とする．もし，稜線 $\overline{p_{h,k-1}p_{h,k}}$，$\overline{p_{h,k}p_{h,k+1}}$ 上で干渉が起こらない位置が見つからなければ，この組み合わせでは配置不可能と見なす．基点が見つかれば，次にその基点を中心として部品の回転を行うことによって，さらに探索を行う．これを回転操作と呼ぶ．(3)の場合には，この基点を中心とした回転操作を行うと必ず干渉が起こるためこの回転操作はできない．そこで，移動操作によって移

った位置で部品が稜線 $\overline{p_{h,k-1}p_{h,k}}$，$\overline{p_{h,k}p_{h,k+1}}$ と接している点を支点として部品を横にずらすことでさらに探索を行う．これをずらし操作と呼ぶ．このような操作を組み合わせることで，目的関数が最小となる位置を見つける．

また，この任意多角形状部品の場合に問題となるのが，部品の表裏を反転させることによって，その部品が違う形状の部品となってしまうことである．これは，矩形部品を扱う場合には問題とはならない．一般的に，特に制約のない場合には，部品の表裏の区別は付けないことが多いが，ここでは簡単化のために部品の表裏の区別があるものとする．よって，探索を行う操作としては，先に述べた移動操作，回転操作及びずらし操作によって表現できる．このように部品の表裏の区別がない場合には，部品の形状情報 Q を定義するところで，時計回りと反時計周りの２つの形状情報 Q_f，Q_b を定義し，その２つの部品の形状情報と稜線の形状情報 P_h との組み合わせを考えることで対応可能である．

これまでに提案されている方法[22]では，最初にすべての部品の位置関係が決まっており，そこから各部品の外形線の左側及び下側が接するように位置決めを行っている．これでは，様々な配置パターンを網羅できているとは言えない．ここで提案している手法では，逐次的に各部品が稜線上を探索することによって，より広い探索空間を持つことができる．

2.3.3 配置アルゴリズム

以下に任意多角形状部品の配置位置選択のアルゴリズムを示す．シーケンシャルに部品を配置していくという考え方は，稜線法と同じである．

Step0 配置順序 $\sigma = (i_1, i_2, ..., i_n)$ を決定する．

Step1 部品を配置していく部材である矩形領域に，形状情報として初期稜線情報 $P_1 = (A_{1,1}, A_{1,2}, A_{1,3}, A_{1,4})$ を生成する．そして，最大稜線数 $mr = 1$ とする．

Step2 配置しようとする k 番目の部品 i_k の形状情報 $Q = (B_1, B_2, ..., B_m)$ を生成する．

Step3 既に配置された部品群の稜線の形状情報 $P_h = (A_{h,1}, A_{h,2}, ..., A_{h,n_h})\ (h = 1, ..., mr)$

に含まれる凹部分と部品の形状情報Qに含まれる凸部分のすべての組み合わせを求め，その中で目的関数が最小となる位置を探索する．

Step4 Step3で選択された位置に部品の配置を行う．そして，配置する位置として選択された位置が含まれる稜線の形状情報の更新を行う．ここで，その位置に部品を配置することで中空領域が出現すれば$mr = mr + 1$とし，その領域に新たな稜線の形状情報$P_{mr} = (A_{mr,1}, A_{mr,2}, ..., A_{mr,n_{mr}})$を生成する．$k = n$であれば終了し，そうでなければ$k = k + 1$としてStep2に戻る．

このアルゴリズムの簡単なイメージを**Fig.2.6**に示す．ここでは凸多角形状部品のみを配置の対象として考えたが，凹部分を含んだ多角形状部品を対象とするアルゴリズムに拡張することは容易である．Step3において，配置する部品の形状情報に含まれる凹部分である頂点と稜線の形状情報に含まれる凸部分である頂点の組み合わせも配置可能位置であるとして探索領域に含めれば対応可能である．また，外形に曲線

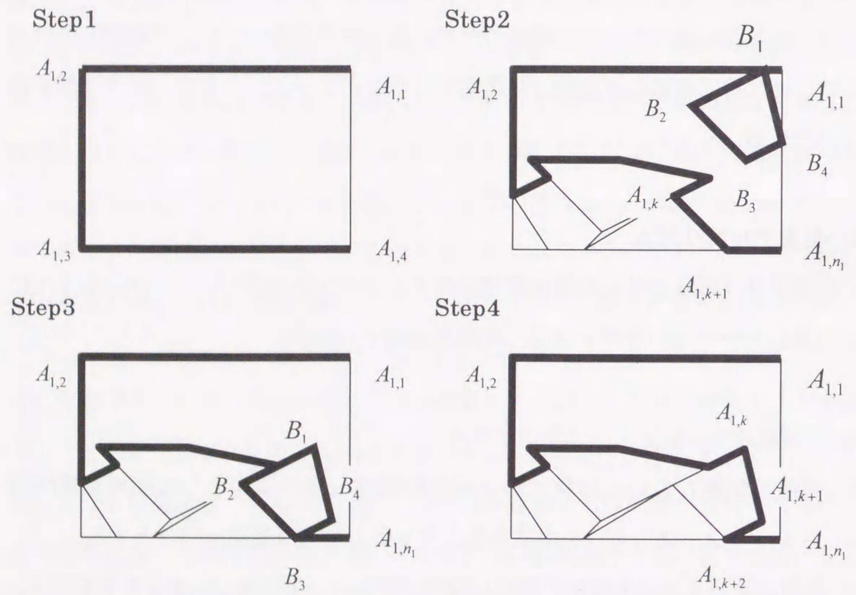


Fig.2.6 Illustration for the irregular shape allocation algorithm

が含まれるような部品に対しても，多角形近似を行うことである程度対応は可能である．

2.3.4 シミュレーション結果

これまで説明した手法を用いて部品を配置したシミュレーション例を示す．**Fig.2.7**は部品数$n = 20$で面積の大きな順に配置した例である．この例では大きさの違う様々な部品を配置しており，この場合の余材率は24.8%であった．また，**Fig.2.8**では同一の形状の部品を配置しており，この場合の余材率は11.1%であった．この例では配置した部品数$n = 14$である．ここで示したように同一形状の部品を配置した場合には，このアルゴリズムを用いると一般的に人が考えるのと同様に２つの部品がペアを作るように組み合わせた形で部品の配置を行う．このように，人の思考に近い配置が可能である．

また，矩形の場合と同様に配置する部品数によって配置効率にどのような影響があるのかを検証する．ここでは，部品数$n = 20$の場合と$n = 30$の場合とで比較を行う．それぞれの場合で，ランダムに10種類のデータを用意し，面積の大きな順に配置したシミュレーションを行った．その結果を**Table 2.2**に示す．これは，10回のシミュレーションで得られた余材率の平均とその中での最良値を示している．これを見ると，$n = 30$の場合には$n = 20$の場合と比べて平均値も最良値も悪い結果となっている．つまり，配置する部品数が多くなると，余材率は高くなる傾向にあるように思われる．任意の多角形状部品であるので，配置する部品の形状が様々であり，余材率を抑えることに関してもある程度限界があることも理解できる．部品数$n = 30$の場合の配置例

Table 2.2 Influence to allocation by number of blocks

	Average[%]	Best result[%]
Number of parts $n = 20$	26.8	23.6
Number of parts $n = 30$	33.1	29.4

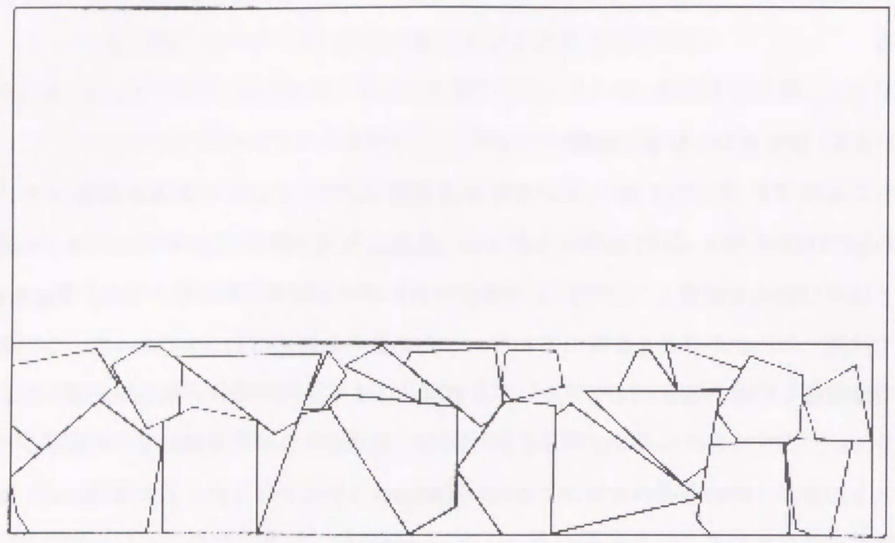


Fig.2.7 Layout example of irregular parts (n=20)

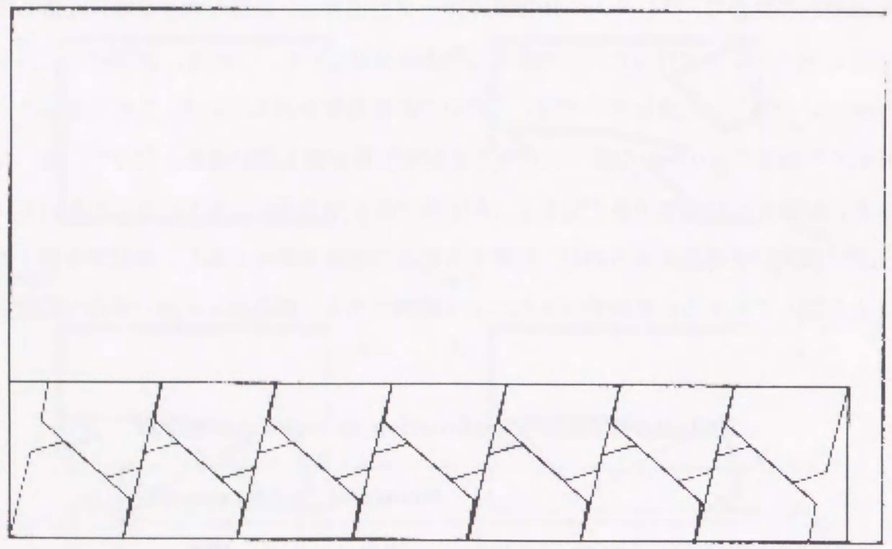


Fig.2.8 Layout example of irregular parts (same shape)

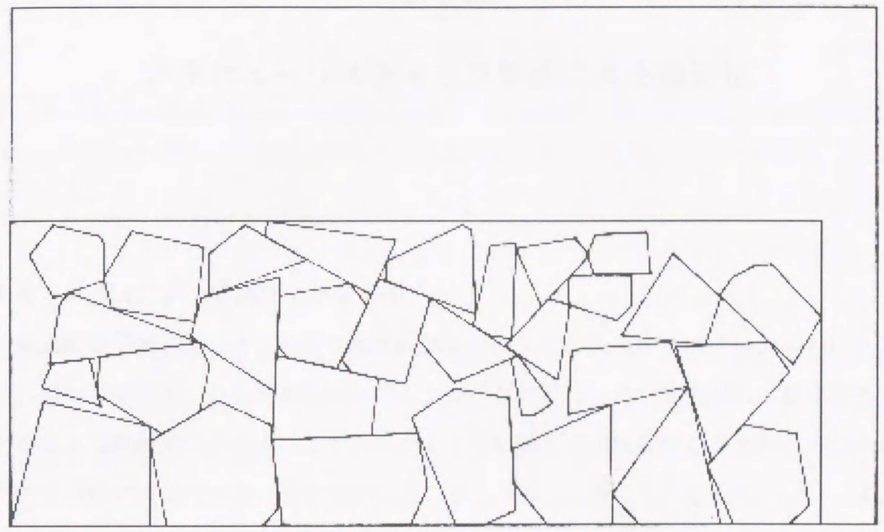


Fig.2.9 Layout example of irregular parts (n=30)

をFig.2.9に示す．この例の余材率は30.4%である．これを見ても，かなり無駄な領域が生じていることが分かる．

ここで示した配置位置選択のアルゴリズムは，これ自体で最適化を行うものではないが，ある配置順序に対して良い配置結果を返すに越したことはない．しかし，この処理に時間がかかっていては，ここでのアプローチ方法の有効性が失われてしまう．このトレードオフの関係も考慮する必要がある．

第３章

メタヒューリスティック手法による最適化

3.1　メタヒューリスティック手法

最適配置問題に対して，部品の配置順序をどのように決定すればよいのかという問題について考える．この問題に対して，ヒューリスティックな知識がある程度有効であることは経験的に分かる．だが，そのような知識がどの程度有効なものであるのか，またその他に有効な知識はないのか，ということははっきりと分かっていない．またそのような知識は，配置する部品の集合によって異なるものであると考えられる．

ここでは，良い配置順序を求めるために一般的な知識という枠を超えた手法であるメタヒューリスティック手法を取り上げる．一口にメタヒューリスティック手法と言っても，様々な手法がこのカテゴリーに含まれる．その代表的な手法として，シミュレーテッド・アニーリング[26],[27]，タブー探索[28],[29]，遺伝アルゴリズム[30],[31]などがある．また，これらが基本的に抱えている様々な問題を解決するために，これらをハイブリッド化した手法[32]-[35]なども提案されている．これまでに最適配置問題に対して，シミュレーテッド・アニーリング[35]-[38]，遺伝アルゴリズム[20][21][39]，ハイブリッド化した手法[22],[40],[41]を適用した報告はあるが，それぞれ違ったアプローチ方法を取っており，どの手法が有効であるのかが客観的に評価されていなかった．そこでこれらの中で，どの手法が最も最適配置問題に対して効果的であるのかを調べるために，それぞれの手法の比較を行う．ここでは，局所探索，シミュレーテッド・アニーリング，タブー探索及び遺伝アルゴリズムを用いる．また，メタヒューリスティック手法ではないが，ランダム探索とヒューリスティック手法も比較のために用いる．

ここで，４つのメタヒューリスティック手法を用いるが，これらの中で遺伝アルゴリズムだけは少し他の３つの手法と枠組みが違う．局所探索，シミュレーテッド・ア

ニーリング，タブー探索の３つは，ある解xが暫定解として与えられた時，近傍$N(x)$を定義し，その近傍内を何らかの戦略を用いて探索を行うものである．よって，その近傍内に存在する解の評価を評価関数$F(x)$で行い，その中での最良解として暫定解を更新し，更新された暫定解の近傍内の探索を行う次のステージへ進むというスタイルで最適解を求める．この手法を用いた最適化の概略を**Fig.3.1**に示す．ここでのALは，最適配置問題においては部品の配置位置選択のアルゴリズムに当たる．よって，近傍内の探索を行う戦略も重要ではあるが，解の近傍が探索のベースであるため，その近傍が非常に大きな役割りを果たすと考えられる．しかし，遺伝アルゴリズムの場合にはこのような近傍を用いるのではなく，ある解集団$G=\{x_i|i=1,....,n\}$が与えられた時，その集団の中で何らかの戦略を用いて変形を行い，その変形された解集団をさらに変

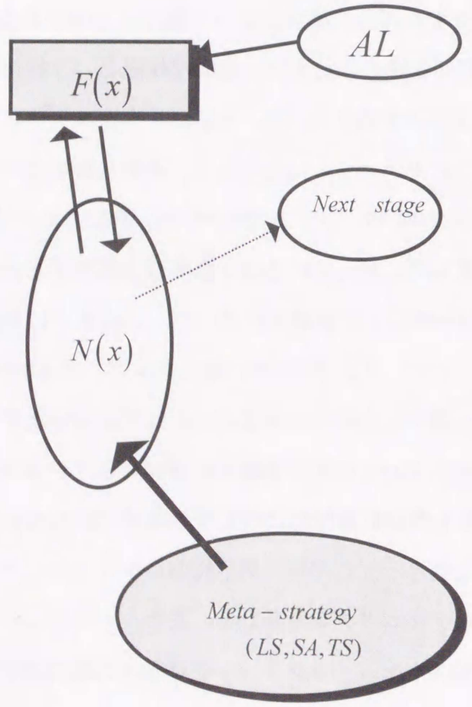


Fig.3.1 Meta-strategy (LS, SA, TS)

形することで次のステージへ進むというスタイルで最適解を求める．この場合も，解集団の中の解を評価するための評価関数$F(x_i)$は必要である．この遺伝アルゴリズムでは，この変形を進化と呼んでいる．よってこの手法では，解集団をどのように変形させるのかという戦略にあたる部分が重要であり，そこでこの手法の良否が決まってしまう．

これらのことから，ここでは局所探索，シミュレーテッド・アニーリング，タブー探索については近傍の定義について検討を加えながら効果の評価を行う．また，遺伝アルゴリズムについては，集団を進化させる戦略について検討を加えながら効果の評価を行う．

以下でそれぞれの手法について，その一般的な特徴と探索のために必要なパラメータについて述べる．また，解近傍の定義など最適配置問題に対して適用した手法についても説明する．

3.1.1 局所探索（local search）

局所探索はメタヒューリスティック手法の中で最も基本的な手法であり，反復改善法とも呼ばれる．これは，ある解xを変形して得られる解集合$N(x)$を解xの近傍として，その近傍内に解xよりも良い解があれば，その解を暫定解として，その暫定解の近傍の探索を行う．そのような解がなければ，その解を局所最適解として計算を終了するというものである．よって，どのステップかで探索は終了すると思われるが，計算時間の制約などがある場合には探索の終了条件も決定しておくことも必要である．ここで，局所探索の簡単なアルゴリズムを示す．

Step1 初期解xを決定し，その解を暫定解とする．

Step2 近傍$N(x)$内にxよりも良い解が存在すれば，その解を新たな暫定解xとして更新する．暫定解の更新が行われないかもしくは終了条件が満たされれば，現在の暫定解xを準最適解として終了し，そうでなければStep2へ進む．

この手法では，暫定解の更新がなければそこで終了してしまう．そのため，探索の結果良い解が得られるかどうかは初期解に大きく依存する．よってこの手法では，良い初期解を見つけることが非常に重要である．しかし，探索前にそれを知ることは非常に困難である．よってこの局所探索法では，このような問題を解決するために，マルチスタート局所探索法(multi-start local search)などが提案されている．これは，初期解を複数持たせることで，これまでの１つの初期解に探索の効果が依存してしまうというリスクを回避するための手段である．

局所探索で決定しなければならないことは，探索の終了条件と解の近傍の定義である．探索の終了条件については，必ずしも設定する必要はないが，本論文で行うシミュレーションでは，各手法との比較を行うという観点から，探索の実行時間をほぼ同じにするために探索の終了条件として最大探索回数 lsn を用いる．よって，暫定解の更新が続いても，最大探索回数に達すれば探索は終了するものとする．

次に，解 x の近傍 $N(x)$ の定義について説明する．この近傍の定義は，シミュレーテッド・アニーリングやタブー探索でも用いられる基本的な要素である．一般的に解 x は次のように定義できる．

$$x=\{s_1,s_2,\ldots\ldots,s_i,s_{i+1},\ldots\ldots,s_j,s_{j+1},\ldots\ldots,s_n\} \tag{3.1}$$

この解 x に対して，要素 s_j を要素 s_i の前に移すような操作が考えられる．その結果，解 x は，

$$x'=\{s_1,s_2,\ldots\ldots,s_j,s_i,s_{i+1},\ldots\ldots,s_{j+1},\ldots\ldots,s_n\} \tag{3.2}$$

のように変形される．このような操作をすべての i と j の組み合わせに対して行う．それらが近傍という集合を形作る要素となる．この操作は挿入操作(insert operation)と呼ばれる．同様にこの解 x に対して，要素 s_j と要素 s_i を交換するような操作も考えられる．その結果，解 x は，

$$x'=\{s_1,s_2,\ldots\ldots,s_j,s_{i+1},\ldots\ldots,s_i,s_{j+1},\ldots\ldots,s_n\} \tag{3.3}$$

のように変形される．このような操作をすべての i と j の組み合わせに対して行い，近傍 $N(x)$ とする．これは，交換操作(exchange operation)と呼ばれる．これらは非常

に一般的な近傍の定義である[18]．これら以外にこれまでに提案されている近傍として知られているのものに，λ 交換(λ - interchange)[42]と呼ばれる手法がある．これは，移転処理(shift process)と交換処理(interchange process)という２つの処理によって近傍を形成するものである．ある意味では，挿入操作と交換操作を組み合わせたような近傍である．解である順列を複数の部分順列に分割し，その部分順列間で要素の移転や交換を行う．

ここで最適配置問題を考えた場合に，当然部品の組み合わせとして良い組み合わせというものが存在する．ここでは，良い組み合わせとはつまり，良い配置順序のことを意味する．これは，全体の順列の中での部分列という形で表れる．よって，そのような部分列が壊されないような近傍の定義を考える．これまでの挿入操作，交換操作のように，各要素単位で挿入したり交換したりするのではなく，あるブロック単位（部分列単位）で挿入したり交換したりする操作を取り入れる．これをそれぞれ n ブロック挿入操作，n ブロック交換操作と呼ぶ．ここでの n は操作する単位であるブロック数である．(3.1)式の解に２ブロック挿入操作を行うと，

$$x=\{s_1,s_2,\ldots\ldots,s_j,s_{j+1},s_i,s_{i+1},\ldots\ldots\ldots,s_n\} \tag{3.4}$$

となり，２ブロック交換操作を行うと，

$$x=\{s_1,s_2,\ldots\ldots,s_j,s_{j+1},\ldots\ldots,s_i,s_{i+1},\ldots\ldots,s_n\} \tag{3.5}$$

となる．

これらの近傍を用いて，どのような近傍が最適配置問題において効果的であるのかについて調べる．

3.1.2 シミュレーテッド・アニーリング（simulated annealing）

局所探索の場合には，近傍内をすべて探索するため，非効率な探索が多くなる．しかし，シミュレーテッド・アニーリングは近傍内を決められた反復回数 L だけの要素しか探索を行わない．つまり，近傍内をすべて探索するわけではなく，ランダムに近傍内の要素を一定数選択して探索を行うのである．よって，同じ初期解から探索を始

めると，常に同じ最適解しか得られない局所探索とは違い，逆に同じ初期解から探索を始めても，同じ最適解に辿り着くことはほとんどない．このようなランダム性が導入されていることはシミュレーテッド・アニーリングの大きな特徴である．

この手法の基礎となる考え方は，焼きなまし(annealing)という物理現象をシミュレートするメトロポリス法[43]というアルゴリズムにあり，温度パラメータTによって探索が制御される．また，評価の低い解も確率的に暫定解として採用するといった要素も含まれている．このシミュレーテッド・アニーリングでは，理論的にある冷却スケジュールに対しては収束性が保証されるという報告[44]もなされている．これは，他のメタヒューリスティック手法では保証されていないことであり，最適解探索において信頼性を与えている．

シミュレーテッド・アニーリングを用いる場合に必要なパラメータは，凍結温度$t_0(>0)$，初期温度$T(0)(>t_0)$及び温度減少率$\gamma(0<\gamma<1)$である．このとき，温度パラメータTの系列は，

$$T(k)=\gamma^k T(0) \tag{3.6}$$

であり，探索の終了条件は，

$$T(k)\le t_0 \tag{3.7}$$

である．また，確率的に暫定解として採用するための確率関数は，解xの目的関数を$F(x)$とし，現在の暫定解をx，評価する解をyとすると，

$$e^{-F(y)-F(x)/T(k)}=e^{-\Delta/T(k)} \tag{3.8}$$

となる．ここで，シミュレーテッド・アニーリングの簡単なアルゴリズムを示す．

Step0 凍結温度t_0，初期温度$T(0)$，温度減少率γ及び反復回数Lを設定する．

Step1 初期解xを決定し，暫定解$p=x$とする．そして，$k=0$，$l=1$とする．

Step2 $y\in N(x)$をランダムに選択する．

Step3 $\Delta=F(y)-F(x)$とする．ここで，$\Delta\le 0$であれば，$x=y$とし，$\Delta>0$であれば，確率$e^{-\Delta/T(k)}$で$x=y$とする．ここで，$\Delta<0$，$F(y)<F(p)$であれば，$p=y$とする．そして，$l=l+1$とする．$l>L$であれば$l=1$，$k=k+1$としStep4へ進み，そうでなければStep2に進む．

Step4 $T(k)<t_0$であれば，現在の暫定解pを準最適解として終了し，そうでなければStep2に進む．

この手法では，温度が高い初期段階では，高い確率で近傍内の悪い解でも探索対象となるようなランダム性が保持されるが，温度が低い最終段階に近づくほど，確率的に悪い解を探索対象とするようなことはほとんど起こらず，局所探索に近い形となる．よって，局所探索で問題であった初期解にかなり依存してしまうという問題点はかなり改善されるが，探索が進めば局所探索と変わらなくなるといった問題がある．この問題を解決するための様々な手法が提案されている．この手法では(3.6)式に従って，徐々に冷却されるが，ある一定以上冷却しても意味がないと判断できれば，再加熱を行うような手法も提案されている[36],[45]．

この手法で決定しなければならないことは，Step0にあるような各種温度パラメータの設定，解近傍の定義などがある．本論文では，先に示した基本的なアルゴリズムを用いてシミュレーションを行う．このことによって，このアルゴリズムが最適配置問題に対して効果的であるかどうかが判断できる．

3.1.3 タブー探索（tabu search）

タブー探索の基本的な手順は局所探索に非常に近い．違っている点は，1つには終了条件を設けることである．局所探索の場合には，暫定解の更新がなければ探索を終了していたが，タブー探索の場合には暫定解の更新がなくても，解近傍の中で最も良い解を用い，その解近傍を探索することで探索を続ける．よって，局所探索のように解の改善が見られないからといって探索をあきらめてしまうのではなく，辛抱強く良い解が見つかることを期待して探索を進める．違っている点のもう1つは，解近傍の中にタブー領域（タブー集合）TLを設定し，近傍内であってもその領域の探索は行わないようにできることである．この目的としては，これまでに探索を行った解の探

索は避けることでサイクリックな探索を行わないようにすることが上げられる．よって，局所探索で行っていた無駄な近傍探索を減らすことが可能になる．このように，タブー探索もシミュレーテッド・アニーリングと同様に局所探索で問題であった点を改善しようとした手法である．

これらが基本的な考え方であるが，タブー探索にはさらに大きな特徴があり，実際に最適解を導き出す上で大きな役割を果たすと思われる．その特徴とは，基本的な枠組みに捕らわれずに，有益なものは何でも取り入れて行こうという考え方である．よって，様々なオプションが用意されている[46],[47]．例えば，アスピレーション基準(aspiration criteria)というものがある．これは，基本的な枠組みの中ではタブー領域内の探索は行わないということであったが，何らかの基準を越え，解として良さそうなものであればタブー領域内の解であっても探索対象とできるというものである．またその他にも，中期記憶(集中化，intensification)や長期記憶(多様化，diversification)などがある．中期記憶とは，探索の過程において有効な解がありそうな領域を集中的に探索するというものである．また長期記憶とは，探索の過程において十分に探索が行われていない領域へ移動し，探索を行うというものである．

このように，非常に様々なオプションを選択できることが特徴である．ここで，タブー探索の基本的なアルゴリズムについて示す．

Step1 初期解 x を決定し，暫定解 $p=x$ とする．そして，タブー集合 $TL=\varnothing$ とする．

Step2 $N(x)-TL$ 内の最良解を求め，その解を x とする．ここで，$F(x)<F(p)$ であれば，$p=x$ とする．

Step3 終了条件が満たされれば，現在の暫定解 p を準最適解として終了し，そうでなければ，タブー集合 TL を更新して，Step2に進む．

ここで，F は目的関数である．この手法は，このアルゴリズムをベースとして様々なオプションが付くことは先に述べた．それを含めて，タブー集合の決定，アスピレーション基準の決定，中期記憶及び長期記憶の利用といったところで，探索が進んできた過程を記憶しておくメモリーが必要となる．そのメモリーによって，これらの戦略が決定されるのである．このそれぞれに関与するメモリーの構成をどのようにするのかも問題によって考える必要がある．

このようにタブー探索は，様々なオプションの利用，それに対するメモリー構成の決定などが非常にアルゴリズムを複雑にしている．そのため，どのオプションが探索に効果的に働いているのかが見えにくい．よって，問題に対するコーディングが非常に難しいという欠点を持っている．

この手法で決定しなければならないことは，タブー集合の構成，解の近傍の定義，探索の終了条件，そしてどのようなオプションを利用し，その構成をどのようにするのかということである．本論文では，タブー集合 TL を次のように定義する．

$$TL=\bigcup_{i=k-sl+1}^{k}C_i,\quad k=1,\ldots,tsn$$

$$C_i=\begin{cases}(p_i,q_i) & (i=1,\ldots,tsn)\\ \varnothing & (i\le 0)\end{cases} \tag{3.9}$$

ここで，タブー集合 TL はリストの形で表現され，sl はその長さであるとする．また C_i には，i 番目のステップでの近傍探索において，最良解として選択された解を得るために使われたムーブが記憶されるものとする．よって，近傍として(3.2)式で表されるような挿入操作を利用した場合，その操作によって移された要素と直前に挿入された要素が記憶されることになる．つまりそれぞれ $p_i=s_j$，$q_i=s_i$ と記憶される．また交換操作を用いた場合も，その時に交換した要素が記憶されるものとする．このように，ここではムーブを代表する要素が記憶される．また，探索の終了条件は，最大探索回数 tsn を用いる．つまり，探索するステップは tsn までということである．

次に，探索に用いるオプションであるが，本論文ではアスピレーション基準を用いる．これは探索の固定化を防ぐためのものである．近傍操作を表すムーブを mi として，アスピレーションレベルを $a(x,mi)$，閾値を A とすると，アスピレーション基準は，

$$a(x,mi)<A \tag{3.10}$$

と表される．このアスピレージョン基準が満たされれば，タブー集合内の要素でも解

として取ることが可能となる．このアスピレーション基準は，暫定解よりも良い解は無条件で選択することを意味している．ここでは，問題に対して影響が大きいと思われる面積の大きな部品に対応した要素が含まれるムーブはタブー集合から外すようにした．

これまでの説明では，タブー探索の中にシミュレーテッド･アニーリングに見られたランダム性は導入されていないが，そのようなことを導入することも許されるのがタブー探索である．いろいろな要素を導入して改良していくのは良いが，これではそれぞれの手法の本質が見えなくなる恐れがある．よって本論文では，先に説明したように，タブー探索の非常に基本的な要素だけで構成した手法を用いて，比較を行う．

3.1.4 遺伝アルゴリズム（genetic algorithm）

遺伝アルゴリズムは，ここで取り上げた４つのメタヒューリスティック手法の中でも少し枠組みが違う．この手法は，自然界のシステムの適応過程である生物の進化のメカニズムを模倣した手法である．これまで述べてきた３つの手法は，基本的に１つの解を近傍操作によって変形させることによって，より良い解を見つけようとするものである．よって，マルチスタート局所探索のように，複数の解を同時に変形させることはあっても，それらの間で相互に何らかの影響を及ぼし合うことはなかった．しかし遺伝アルゴリズムでは，複数の解（個体，individual）を同時に持たせた集合（集団，population）の中で，相互の関係で影響を及ぼし合いながら集団の中でそれぞれの解が変形（進化，evolution）して，これを何回（世代，generation）か繰り返してより良い解を見つけようとするものである．

この遺伝アルゴリズムの中で，解の表現(genotype)は(3.1)式と同様にできる．この中の要素であるs_iは，通常遺伝子(gene)と呼ばれ，その位置を遺伝子座(locus)と呼ばれる．最適化問題の解法として用いられる場合，この遺伝子として取られる値は，0または1の整数（0-1表現）か，1からnまでの整数（順序表現）がほとんどである．解集団を進化させていく操作は，このような個体の表現型に依存する．その集団を進化させていく操作としては，生物の進化のメカニズムと同様に，交叉，突然変異，再生

（淘汰）などが用意されている[48),(49)].

ここで，集合の中に持たせた複数の解の数（個体数，population size）をgmとし，最大探索回数（世代数）をgnとし，簡単に遺伝アルゴリズムのアルゴリズムについて示す．

Step1 gm個の初期解の集団を生成し，その中で最も良い解を暫定解とする．そして，$k=1$とする．

Step2 解の集団の中で，一定の確率で交叉や突然変異を行い，新しい解（子）を生成する．

Step3 得られた新しい解の適応度を求め，それを用いて解の再生（淘汰）を行い，新しい世代の解の集団を作る．この集団の中で，最も良い解が暫定解よりも良い解であれば，その解を新たな暫定解とする．そして，$k=k+1$とする．

Step4 $k>gn$であれば，現在の暫定解を準最適解として探索を終了し，そうでなければ，Step2に進む．

ここで，Step2で交叉，突然変異の操作を行い，Step3で再生の操作を行っている．この中の再生という操作では適応度(fitness)を定義し，基本的にその適応度の高い解が多く集団に残るようにする．よって，低い解は集団から除く操作を行うのが一般的である．そのため，集団の中での解のばらつきが少なくなる傾向がある．このことにより初期収束(premature convergence)と呼ばれる問題が起き，最適解に辿り着く前に局所解に収束してしまう．よって遺伝アルゴリズムでは，その進化の過程でいかにして集団の多様性を維持するのかが問題となる．自然界においても，集団が均質化されてしまうと，進化が進みにくくなるのと同じである．よって，交叉，突然変異等の操作をどのように行うかで，良い探索を実現できるかどうかが決まる．

a 交叉（crossover）

交叉の操作は，遺伝アルゴリズムの最も大きな特徴であり，非常に重要な役割を果

たす操作である．この操作によって，集団の中でそれぞれの解が相互に影響を及ぼし合う．この基本的な考え方は，集団の中から任意の２つの解を親として選択し，それぞれ相手の解を参考にして遺伝子の並び換えを行い，２つの新しい解を子として生成するというものである．この操作を交叉確率 $P_{Crossover}\left(0 \le P_{Crossover} \le 1\right)$ の元で行う．交叉確率が高くなるほど，多くの新しい子が生まれることになる．交叉の操作では，親として選ばれる２つの解が同じ場合には，子として生成される新しい２つの解は親とまったく同じになる．このように，解の同質性を保持する特徴を持っている．

ここで代表的な交叉の操作について説明する．この操作は解の表現型に大きく依存している．解の表現型として0-1表現を用いている場合には，１点交叉(one-point crossover)や多点交叉(multi-point crossover)がある[49]．これらは，解の上でランダムに切れ目をいくつか選び，その切れ目によってできた部分列を２つの解の間で入れ換える操作である．その他に一様交叉(uniform crossover)と呼ばれるものもある．これは，あるマスクパターンを作成し，それを元にして親である２つの解の遺伝子情報を組み合わせるものである．また，解の表現型として順序表現を用いている場合には，重複を許さない順列として解が表現されていることが多い．よって，0-1表現の場合と同様にある切れ目でただ単純に入れ換えると，新たに生成された解の中で同じ遺伝子の重複が起こる可能性がある．この場合，このような解は実行可能解とはならない．よって順序表現を用いている場合には，順序交叉(ordered crossover)がある．これは，ランダムに切れ目を１つ選び，切れ目の左側はそのまま引き継ぎ，右側は相手の並びを参照して並び換えるようなものである．その逆で，切れ目の右側をそのまま引き継ぐような操作もある．

最適配置問題に適用する場合には，解の表現型が1からnまでの整数で表現される順序表現となる．よって，本論文では順序表現で用いる交叉として一般的な順序交叉を用いる．そして，配置に大きな影響がある最初に配置される部品を表現する切れ目の左側がそのまま引き継がれる操作と，逆に後に配置される部品を表現する切れ目の右側が引き継がれる操作を行うとする．

b　突然変異（mutation）

突然変異は，自然界では生物の多様性を維持するために起こるものである．遺伝アルゴリズムの上でも同様であり，集団の中での解がすべて同質のものとならないようにするための操作である．この基本的な考え方は，ランダムにある１つの解を選択し，その解の遺伝子情報に変化を与え，新しい解を生成するというものである．この操作を突然変異確率 $P_{Mutation}\left(0 \le P_{Mutation} \le 1\right)$ の元で行う．一般的に，この突然変異確率を高くするとランダム探索に近くなると言われている．

ここで代表的な交叉の操作について説明する．この操作としては，交換(exchange)，移動(shift)または逆位(inversion)の操作[49]などがある．この逆位の操作は，解の上で部分的に遺伝子の並びを入れ換えるような操作である．0-1表現では，解のある部分の遺伝子座の情報を逆にする操作であり，順序表現では，解のある２つの遺伝子座の間で情報を入れ換える操作である．この操作を突然変異と別の枠組みで捉える考え方もある．

これまでにも述べたが，最適配置問題では解の表現型として重複を許さない順序表現を用いる．よって，挿入や欠損など解の長さが変わるような操作は適当でない．そこで，本論文では突然変異として交換の操作を用いる．一般的には，ランダムに選択した２つの遺伝子を入れ換える操作である．

このように解が順序表現で扱われる最適化問題として代表的な問題に巡回セールスマン問題(TSP)がある．この問題では，どの遺伝子座も全体の解に与える影響は同じであると考えられる．基本的に得たい解は巡回するパスであるので，解としてはどこで切って順序列として表現するかということだけだからである．しかし，最適配置問題では遺伝子座によって解に与える影響が違うと考えられる．一般的に，最初に配置される部品の配置が後に配置される部品の配置に大きな影響を与えると考えられる．そこで，それぞれの遺伝子座において，突然変異が起こる遺伝子座として選ばれる確率 $Pm_i\left(i=1,\ldots,n\right)$ を考え，それらが違っているような操作を考える．このとき，i 番目の遺伝子座が突然変異の起こる遺伝子座として選ばれる確率 Pg_i は次のようになる．

$$Pg_i = P_{Mutation} Pm_i, \quad i = 1,\ldots,n$$

$$\sum_{i=1}^{n} Pm_i = 1 \tag{3.11}$$

ここで最適配置問題では，

$$Pm_i \geq Pm_j \quad (i < j) \tag{3.12}$$

となるように設定することが考えられる．このようにすることで，最初に配置する部品に当たる遺伝子座で突然変異が起こる確率が高くなり，解の構造を大きく変えることが可能になる．よって結果として，集団全体の多様度も大きくなり，初期収束に陥りにくくする(50)．

c　再生（reproduction）

再生は，適応度の高い個体が生き残るという自然淘汰のことであり，この考えを遺伝アルゴリズムでも取り入れている．この操作は，集団内のすべての解に対してその適応度を求め，その適応度を元にしてその解を集団から除くのか，そのまま残すのか，その解を増やすのかを決定するものである．これらは，再生の規則に則って行われる．このとき一般的に，集団内の解の数が変わらないように再生される．この規則としては，ルーレット方式(roulette selection)やランキング方式(rank selection)などがある．ルーレット方式とは，適応度に応じて確率的に再生を行う手法であり，適応度の低い解は集団から除かれるようになる．よって，解の再生確率と適応度とが比例関係にある．またランキング方式とは，適応度により解の順序付けを行い，その順序に応じて解の再生確率が決まるという手法である．よって，適応度が低い解でも集団に残るようにすることが可能である．また，エリート保存戦略(elitism)という方式もあり，これは適応度が集団の中で最大の解は必ず残るようにする戦略である．このことにより，交叉や突然変異などの操作で良い解が破壊されることはなくなる．

再生を行うためには，規則の元となる適応度と解の再生確率を決定する必要がある．そして，オプションとしてどのような例外処理を導入するのかも考えなければならない．本論文では，一般的なルーレット方式を用いることとする．そこで，解xの適応

度$Fi(x)$を次のように定義する．

$$Fi(x) = \left(FW - F(x)\right)^2$$

$$FW = \max_{x \in U}\left(F(x)\right) \tag{3.13}$$

ここで，Uは解の集団（母集合）であり，Fは目的関数である．また，FWはその集団内で最も悪い解である．これを元にして，各解の再生確率$E(x)$を次のように定義する．

$$E(x) = Fi(x) \Big/ \sum_{y \in U} Fi(y) \tag{3.14}$$

ここでは，ルーレット方式を採用したために，適応度の高い解が集団の中で増えすぎる懸念があるが，交叉，突然変異の操作によってできるだけ解集団の多様度が失われないようにする．また，エリート保存戦略は用いないものとする．

3.1.5　その他の手法

大規模な組合せ最適化問題の解法として，これまでは経験的な知識すなわちヒューリスティックスを用いて解こうとする試みが主流であった．これを実現するためには，その問題に対して精通している必要があり，その問題の特徴をうまく利用することができなければならなかった．よって，そのような手法は，その特定の問題に対しては効果を発揮するが，別の問題をそれで解こうとするとうまくいかないことが多い．メタヒューリスティック手法は，ヒューリスティックスを越えた枠組みであり，ある問題に対して効果があれば，別の問題に対しても同様の効果が期待できるというメリットがある．よって，それほど問題に精通していなくても，問題を解くことが可能となる．このような手軽さということも，利用者の側から見れば必要なことであり，複雑な問題に対処する際に，このような問題の性質が分からなくても使えるツールに対する要望が強くなってくると思われる．実際にエンジニアリングの世界においても，設計者が気軽に誰でも使える解析ツールというものが市場に出てきており，そのようなニーズが生まれている．

そこで，メタヒューリスティック手法の間だけでなく，このヒューリスティックスを用いた場合との比較も行うが，ここで用いたヒューリスティックスは，“部品の面積の大きい順に配置を行う”という簡単なものである．よって，ヒューリスティック手法とメタヒューリスティック手法との間のしっかりした比較とはならないが，問題に対して精通していない人が組合せ最適化問題をような複雑な問題を解く場合に，簡単なヒューリスティックスしか分からない場合とこのメタヒューリスティック手法を用いた場合という観点での比較は可能であろう．これまでの説明の中で，ここで用いるメタヒューリスティック手法は標準的なものがほとんどである．遺伝アルゴリズムのパラメータ設定の中で，“最初に配置する部品が後に配置する部品の配置に与える影響が大きい”という知識を用いているが，この知見はヒューリスティック手法で用いる知見と同義のものである．

またその他に，ランダム探索(random search)との比較も行う．ここでは，ランダムに一定数の解を生成し，その中から最も良い解を準最適解とする．生成する解の数はランダム探索数rsnであるとする．

3.2 メタヒューリスティック手法の評価

これまでに説明した手法を最適配置問題に適用して結果を示す．最初に，比較対象とした４つのメタヒューリスティック手法の中で，近傍探索としてその枠組みが似ている局所探索，シミュレーテッド・アニーリング，タブー探索において，大きな役割を果たすと思われる解近傍の評価を行う．次に，集団探索とも言える遺伝アルゴリズムについて，交叉，突然変異など集団の進化に関与するパラメータを変えることで，その探索に与える効果の評価を行う(50)．

最後に，先に行った解近傍や遺伝アルゴリズムの評価をもとにして，これら４つのメタヒューリスティック手法の間で比較を行う．そしてさらに，ヒューリスティック手法及びランダム探索といった手法との比較も行い，最適配置問題の解法として最も有効な手法及び操作を検証する(51)．

Table 3.1 Search effects of neighborhood (1)

a) Case 1

	LS	SA	TS
Insert	7.14	5.37	6.72
Exchange	7.08	5.07	5.86

b) Case 2

	LS	SA	TS
Insert	5.24	4.54	4.62
Exchange	4.88	4.05	5.00

3.2.1 解近傍の評価

近傍の定義は，近傍探索において探索を進める上でベースとなる部分である．そこで，どのような近傍を用いれば探索効率が上がるのかを見る．ここでは近傍として，まず挿入操作と交換操作で比較を行う．この近傍を局所探索，シミュレーテッド・アニーリング及びタブー探索において利用した．部品数$n=20$の矩形部品のデータを２種類用意し，それぞれ10回のシミュレーションを行った．シミュレーションに用いた各種パラメータとして，局所探索では，挿入操作の場合$lsn=5$，交換操作の場合$lsn=10$とした．またシミュレーテッド・アニーリングでは，挿入操作も交換操作も$t_0=5$，$T(0)=100$，$\gamma=0.85$，$L=100$とした．そしてタブー探索では，タブーリストの長さ$sl=3$，最大探索回数tsnは，挿入操作の場合$tsn=8$，交換操作の場合$tsn=15$としている．そのときの余材率の平均値を**Table 3.1**に示す．

この表を見ると，挿入操作よりも交換操作のほうが効果的であることが分かる．挿入操作と交換操作を比べると，交換操作のほうが近傍内の要素数（近傍サイズ）は小さい．ここで行ったシミュレーションでは，どの手法でも計算時間がほぼ等しくなるようにパラメータを決定している．よって同じ計算時間であれば，シミュレーテッド・アニーリングを除いて交換操作のほうが最大探索回数を多く取ることが可能であった．よって，近傍サイズを小さくして探索回数を多くし，できるだけ多くの近傍探索を行うほうが効率は良いことが分かる．

Table 3.2 Search effects of neighborhood (2)

a) Data1

	n=1	n=2	n=3	n=4	n=5	n=6
LSI	19.554	19.409	20.700	19.837	19.903	20.111
LSC	18.901	19.358	21.327	21.650	22.346	23.584
SAI	18.962	19.097	18.972	19.334	18.688	19.255
SAC	17.670	20.632	20.673	22.326	21.367	22.991

b) Data2

	n=1	n=2	n=3	n=4	n=5	n=6
LSI	20.488	20.996	19.405	19.304	20.277	20.219
LSC	20.415	20.000	20.218	21.761	22.367	24.285
SAI	19.716	18.762	19.283	18.992	19.302	19.471
SAC	19.228	20.016	21.370	22.332	22.802	23.830

次に，nブロック挿入操作とnブロック交換操作を用いて比較を行う．この近傍を局所探索及びシミュレーテッド・アニーリングにおいて利用した．部品数$n=20$のデータを２種類用意し，それぞれ10回のシミュレーションを行った．ここで用意した部品は矩形部品ではなく任意多角形状部品である．局所探索の最大探索回数lsnは，挿入操作の場合$lsn=10$，交換操作の場合$lsn=15$とした．またシミュレーテッド・アニーリングでは，挿入操作も交換操作も$t_0=40$，$T(0)=1000$，$\gamma=0.85$，$L=100$とした．そのときの余材率の平均値をTable 3.2に示す．また用いた近傍は，移動するブロックの単位が1から6までを用いた．

この表を見ると，挿入操作と交換操作とでは少し傾向が違う．交換操作の場合には，交換するブロックの単位が大きくなれば，それに従って結果が悪くなっている．しかし，挿入操作の場合には移動するブロックの単位に関係なく同じような結果となっている．これは，メタヒューリスティックスの手法に依らず，局所探索の場合もシミュ

レーテッド・アニーリングの場合も同様の傾向にある．これは，交換操作の場合には近傍として交換するブロックの単位を大きくすると，まったく違った解となってしまい，近傍としての機能が失われてしまうと考えられる．しかし挿入操作の場合には，移動するブロックの単位を大きくしても，近傍としては機能しているが，それほど大きな改善が見られるわけではない．よってこの最適配置問題では，ある解の並びの中では配置の組合せとして良い性質を持った部分列でも，それとは違う解の並びの中でもその部分列が良い性質を示すとは限らないと考えられる．よって，良い性質を持った部分列を生かすようにするためには，ここで取るようなアプローチとは違った方法を考える必要がある．

一般的に，局所探索，シミュレーテッド・アニーリング，タブー探索における近傍は，局所探索が可能となることが期待されている．よって，最適配置問題のように解の性質が判断しにくい問題に対しては，解の構造を大きく変えない近傍を定義する必要がある．さもないと，各手法でのそれぞれの戦略が生きてこないことになる．Table 3.1，Table 3.2の結果から，これらの近傍定義の中で最も効果的であったのは1ブロッ

Table 3.3 Comparison of crossover rate and mutation rate

a) Case 1

	GA1 $P_{Crossover}=0.9$ $P_{Mutation}=0.1$	GA2 $P_{Crossover}=0.1$ $P_{Mutation}=0.9$	GA3 $P_{Crossover}=0.5$ $P_{Mutation}=0.5$	GA4 $P_{Crossover}=0.7$ $P_{Mutation}=0.3$
Average[%]	6.06	5.68	6.00	5.87
Best result[%]	4.07	4.83	4.73	3.87

b) Case 2

	GA1 $P_{Crossover}=0.9$ $P_{Mutation}=0.1$	GA2 $P_{Crossover}=0.1$ $P_{Mutation}=0.9$	GA3 $P_{Crossover}=0.5$ $P_{Mutation}=0.5$	GA4 $P_{Crossover}=0.7$ $P_{Mutation}=0.3$
Average[%]	4.59	4.549	4.552	4.58
Best result[%]	4.17	4.00	4.00	4.33

ク交換操作であり，このことは配置する部品が矩形部品であっても任意多角形状部品であっても同じである．

3.2.2 遺伝アルゴリズムの評価

遺伝アルゴリズムでは，いかに集団に多様性を持たせながら効果的に進化させるのかが重要である．その役割を担う交叉と突然変異の操作について，最適配置問題での働きを評価する．まず，交叉と突然変異との関わりを見るために，交叉確率 $P_{Crossover}$ と突然変異確率 $P_{Mutation}$ をいくつかのパターンを調べる．部品数 $n=20$ の矩形部品のデータを２種類用意し，それぞれ10回のシミュレーションを行った．シミュレーションに用いたパラメータであるが，世代数 $gn=50$，個体数 $gm=40$ とし，突然変異が起こる遺伝子座として選択される確率として，１番目の遺伝子座が選択される確率 $Pm_1=0.3$，その他をすべて同じ確率とした．交叉については順序交叉を用い，切れ目の左側がそのまま引き継がれる操作を用いた．ここでは，交叉確率と突然変異確率を変えた4つの組み合わせを調べた．その結果として，余材率の平均値とその中での最良値をTable 3.3に示す．また，探索がどのように進んでいるのかを見るために，各世代において集団内の解の平均値とその中での最良値の変化について，それぞれの場合をFig.3.2, Fig.3.3, Fig.3.4, Fig.3.5に示す．

この結果，特にTable 3.3を見ると，ここで行ったコーディングでは，交叉の操作によって大域的な探索が行われ，突然変異の操作によって局所的な探索が行われていると思われる．突然良い解を見つける可能性が高いのは交叉確率が高い場合であり，探索が効率的に行われているのは突然変異確率が高い場合であることが分かる．この理由は，ここで扱っている最適配置問題の解のコーディングが順序表現によって行われていることにある．局所探索等における近傍の定義による比較のところでも述べたが，解の構造を大きく変えないようにしなければ局所的な探索は実現できない．また，解の中で部分列の単位で情報を保持し，次の世代へ引き継ごうとしても，その周辺の並びが変わってしまうことで，その部分列の情報が引き継がれた解では意味を変えてしまう．ここで行っている交叉の操作は，そういった意味で解の構造を大きく変えて

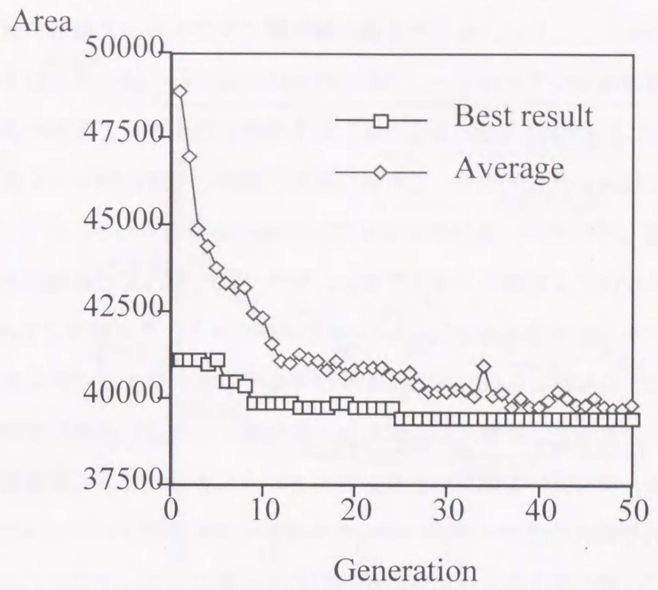


Fig.3.2 Dynamics of GA1

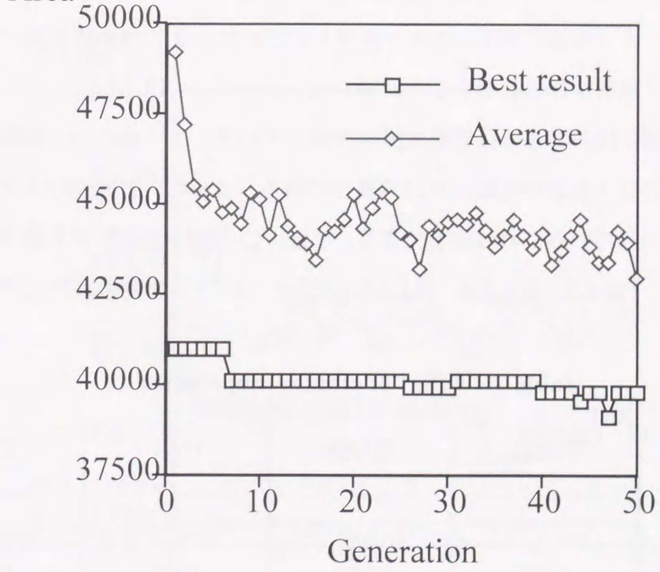


Fig.3.3 Dynamics of GA2

しまう可能性がある．これは，まったく違う解の間で交叉を行った場合に言えることであり，同じ解の間で交叉を行うと，これとは逆に生成される解の構造はまったく変わらない．よってFig.3.2に見られるように，交叉確率を高くすると集団の多様性は世代を重ねるごとに失われてしまう．これは，再生の操作としてルーレット選択を用いたことにも起因しているが，遺伝アルゴリズムの悪い面が出ている．

次に，交叉の操作による違いを見るために，順序交叉で切れ目の右側がそのまま引き継がれる操作を用いた場合のシミュレーションを行った．その結果をTable 3.4に示す．この表で，GAR1とGAR2が切れ目の右側をそのまま引き継ぐ交叉の操作を用いている．これを見ると，結果として切れ目の左側をそのまま引き継いだ操作よりも悪くなっている．切れ目の右側が引き継がれるということは，部品の配置順序として最初に配置される部品の並びの情報はまったく変わってしまうということである．よって，これまでに述べてきたように解の構造が大きく変わってしまう．よってランダム探索に近くなり，結果としてそれほど効果が得られないということになる．このように，ここでの交叉の役割が大域的な探索の役割に近いといっても，解の構造を大きく変えてしまうような操作を用いると，近傍としてブロック交換操作を用いた場合と同様にまったく効果が得られないことが分かる．

突然変異の操作は，ここでは交換の操作を用いている．この操作は，局所探索等で用いた近傍の定義である交換操作と同じ操作である．よって，この操作によって局所的な探索が行われていることは理解できる．また，この操作で突然変異確率をある程度高く取れば，Fig.3.3，Fig.3.4，Fig.3.5に見られるように集団の多様度が維持され

Table 3.4 Effect of crossover operation

(Case 1)	GAR1 $P_{Crossover}=0.9$ $P_{Mutation}=0.1$	GAR2 $P_{Crossover}=0.1$ $P_{Mutation}=0.9$	GA1 $P_{Crossover}=0.9$ $P_{Mutation}=0.1$	GA2 $P_{Crossover}=0.1$ $P_{Mutation}=0.9$
Average[%]	6.69	6.48	6.06	5.68
Best result[%]	5.12	6.01	4.07	4.83

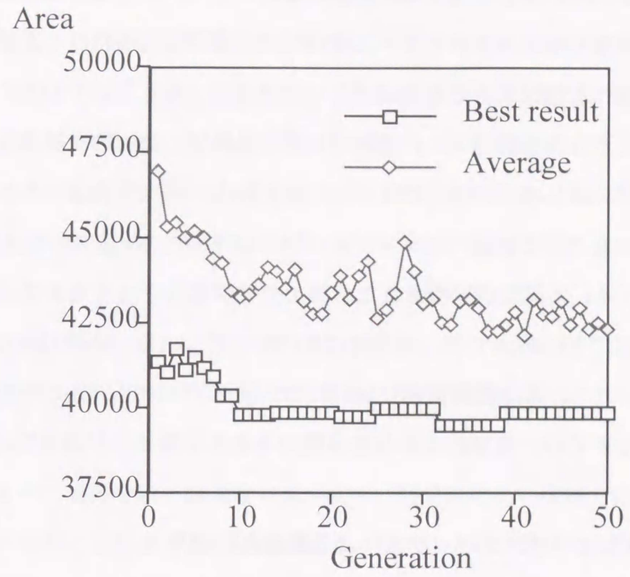


Fig.3.4 Dynamics of GA3

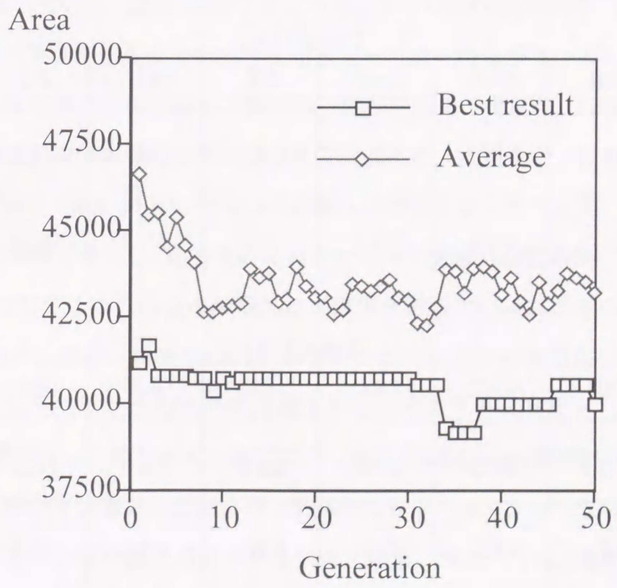


Fig.3.5 Dynamics of GA4

ることも分かる．このことから，突然変異の操作としていろいろな操作があるが，このような順序表現で解が表されるような問題では，近傍探索における近傍として定義できるような操作を選択することが望ましいと言える．もしここで大きく解の構造を変えてしまうような操作を選ぶと，交叉でも突然変異でも局所的な探索を実現できなくなり，遺伝アルゴリズムの性質はまったく見られない探索となる．これらのことから判断して，結果として遺伝アルゴリズムの中で効果的であったのはGA2である．他の手法においても，多様度が維持された状態で交叉の操作を行うことによって突然良い解を見つけることはあっても，平均的に良い解を見つけているのはGA2である．

これらのことから，最適配置問題においては，遺伝アルゴリズムの特徴である交叉の操作が機能しにくく，最適化がかなり困難であるということが言える．

3.2.3 各メタヒューリスティックスによる最適化の効率

メタヒューリスティック手法の中で，どの手法が最適配置問題に有効であるのかを，シミュレーションで比較することにより検証する．部品数$n=20$の矩形部品データを２種類用意し，それぞれ10回のシミュレーションを行った．このデータは，3.2.2で遺伝アルゴリズムの評価を行ったデータと同じデータである．そして，ここでは効率を比較するために，計算終了までの時間がほぼ同じとなるように次のようにそれぞれの探索でのパラメータを設定した．局所探索(LS)の場合，最大探索回数$lsn=5$とした．シミュレーテッド・アニーリング(SA)の場合，$t_0=5$，$T(0)=100$，$\gamma=0.85$，$L=100$とした．タブー探索(TS)の場合，タブーリストの長さ$sl=3$，最大探索回数$tsn=8$とした．これらの手法において近傍を定義する必要があるが，ここでは1ブロック挿入操作を用いた．遺伝アルゴリズムについては，3.2.2におけるGA1とGA2を比較に用いた．そして，メタヒューリスティック手法以外の方法として，ランダム探索(RS)では，ランダム探索数$rsn=2000$とした．ここではヒューリスティック手法(HM)も比較対象としている．シミュレーションの結果，得られた余材率の平均値とその中の最良値をTable 3.5に示す．また，遺伝アルゴリズムを除く他の３つのメタヒューリスティック手法において，計算時間と見つける最良解との関係を検証するために，２つのデータそれぞれの場合について，探索の過程をFig.3.6，Fig.3.7に示す．

この結果を見ると，最適配置問題においてメタヒューリスティック手法として効果があるのは，SA，GA2，GA1，TS，RS，LS，HMの順であることが分かる．SAが最も効果的であった理由は，このような非線形性の強い問題において，温度パラメータによる確率的変動とランダム性をうまく用いた効果が顕著に現れたためだと思われる．特にCase 1の場合には明らかに他の手法に比べてSAが優れている．探索効率においても，最良値を見つける可能性についてもSAが最も優れている．またCase 2の場合には，SA，GA2，GA1，TSの間でそれほど大きな差が表れていない．これは，Table 3.2で行った遺伝アルゴリズムの評価の中でも，データとしてCase 2を用いた場合に，それぞれの手法の間であまり違いが表れなかった．このことから，与えられたデータによって問題を解く難易度に差があり，難易度の低い問題では手法間での差があまり表れなかった結果であると思われる．このことは，HMでもかなり良い結果が得られていることからもそのように判断できる．

Table 3.5 Results of computational experiments

a) Case 1

	LS	SA	TS	GA1	GA2	RS	HM
Average [%]	7.14	5.37	6.72	6.06	5.68	6.79	9.36
Best results[%]	5.30	2.41	5.85	4.07	4.83	6.13	9.36

b) Case 2

	LS	SA	TS	GA1	GA2	RS	HM
Average [%]	5.24	4.54	4.62	4.59	4.55	5.01	6.94
Best results[%]	4.09	3.83	2.98	4.17	4.00	4.33	6.94

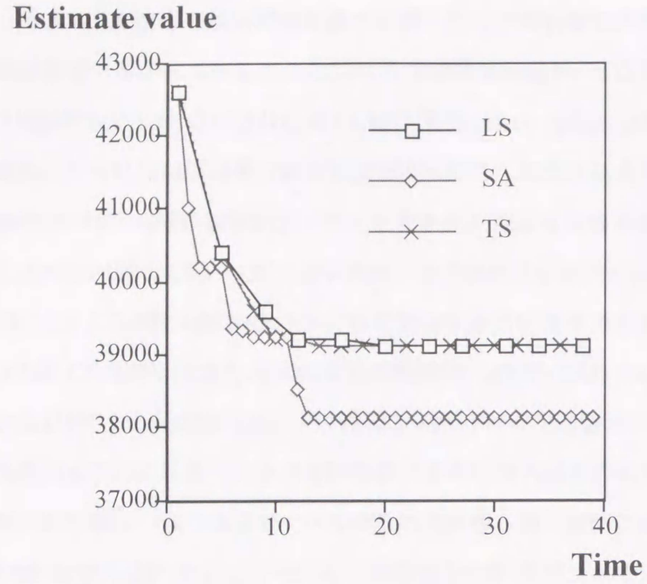


Fig.3.6 Dynamics of Case 1

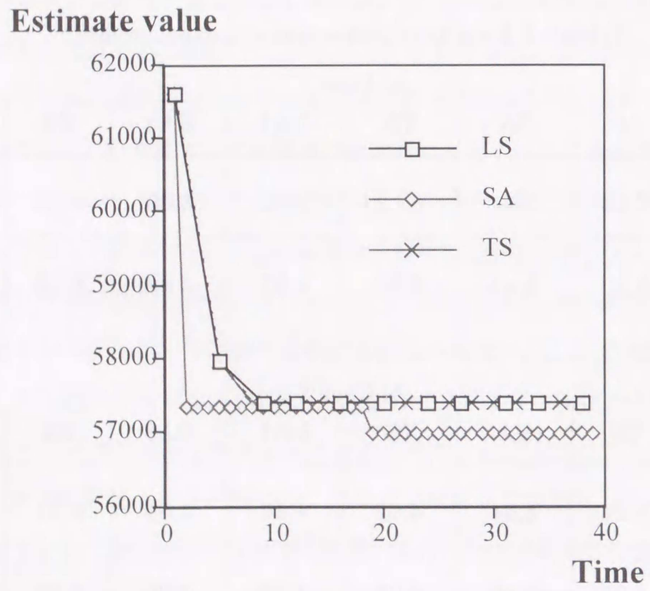


Fig.3.7 Dynamics of Case 2

SAに次いでGA2も良い結果を出している．しかし遺伝アルゴリズムは，3.2.2での評価でも述べたが，この最適配置問題の性質から交叉の操作による効果が得にくいという問題があることは既に述べた．この手法では集団内で様々な操作が可能ではあるが，これらの操作を最適配置問題に特化してうまく利用しないと，十分な効果は得られない．よってここで見られるように，基本的なシミュレーテッド・アニーリングで得られる性能にも達しないということになる．つまり，最適配置問題で効果を発揮するようにするためのコーディングが難しいということになる．

またTSについてであるが，ここで行ったシミュレーションではタブー探索の基本的な要素であるタブー集合とアスピレーション基準を利用しただけである．よってタブー探索は，シミュレーテッド・アニーリングのように基本的な要素を利用しただけで効果が得られるものではないことは明らかである．タブー探索はどのような操作でも有効なものは取り入れようとする考えがあり，どのようにでも問題に特化した形で探索操作を選択したり変形したりできる．これまでに，組合せ最適化問題に対して適用した報告でも，タブー探索が効果的であるという報告がある[52],[53]．しかし，ここで得られた結果より分かることは，そこに辿り着くまでにかなり高いハードルがあるということである．よって，遺伝アルゴリズムと同様にコーディングの難しさに問題がある．

最後に，LSはRSよりも結果が良くなかった．このことは，近傍探索だけでは大きな効果は得られないということを示している．逆に言えば，ある程度ランダム性を考慮した探索が効果的であることを示している．SAが優れていたのも，このランダム性が探索の中で行かされているためであることが予想できる．

また，Fig.3.6，Fig.3.7を見ると，最も効果的であったシミュレーテッド・アニーリングについては，一般的に言われている問題である探索が進むにつれて局所的な探索の要素が強くなり，探索効率が悪くなるという傾向が表れている．ここでは，標準的な手法しか用いていないためにこのような結果に終わっている．よって，その点を改善すれば，さらに効果的な探索が実現できると思われる．また，局所探索とタブー探索についてはほとんど似たような探索過程を辿っている．これは，アスピレーショ

ン基準という戦略はそれほど有効でなく，基本的な枠組みである局所探索の効果しか出ていないことを物語っている．よって，この近傍探索から抜け出るような戦略が必要であることが分かる．

ここで得られた結果は，タブー探索による結果を除けば，スケジューリング問題において示された結果[54]とほぼ一致している点が特に注目される．いずれの場合も，メタヒューリスティック手法として，シミュレーテッド・アニーリング，遺伝アルゴリズム，局所探索の順に良い解が得られている．このように異なる問題について，同様の傾向が見られることは非常に興味深い点である．

3.3 部分アルゴリズムを用いた最適化

組合せ最適化問題では，TSPのように比較的評価の簡単なものから，スケジューリング問題のように評価の複雑なものまで様々である．ここで扱っている最適配置問題は，非常に評価の複雑な問題に属する．このような問題では，評価を決定するためにさらに部分アルゴリズム AL を必要とする問題も多く，そのようなアルゴリズムも複数考えられる場合がある．

本論文では，最適配置問題へのアプローチ方法として，この問題を配置位置選択の問題と配置順序決定の問題という２段階構造の問題として捉え，その解法を考察している．この中では配置位置選択のアルゴリズムが部分アルゴリズム AL に当たる．現実に存在する問題としては，このような複雑な問題が大半を占めると考えられる．しかし，最適化の評価に用いられる目的関数である部分アルゴリズムの選択によって，メタヒューリスティック手法を用いた最適解探索にどのような影響があるのかについては，明確にされていない．

そこで，これまでに部分アルゴリズムとメタヒューリスティック手法をそれぞれ個々の問題としてその解法や特徴の説明を行ってきたが，ここでは部分アルゴリズムである配置位置選択のアルゴリズムとメタヒューリスティック手法を用いた最適化との間でどのような関係があるのかについて調べる．

3.3.1 部分アルゴリズム（partial algorithm）

第２章では，配置する部品が矩形部品に限られたアルゴリズムと任意多角形状部品の配置まで可能なアルゴリズムについて説明した．ここでは，配置する部品が矩形に限られた場合において，ある配置順序 σ を評価する部分アルゴリズムを３種類用意する．本論文で最適配置問題の解は(2.1)式のように $\sigma=(i_1,i_2,\ldots,i_k,\ldots,i_n)$ と表され，この順に部品を配置していく．ここで考えるアルゴリズムは，部品 i_k を配置するときには，既に部材上に i_1 から i_{k-1} までの部品が配置されていることになり，そのときに部品 i_k をどの位置に配置すればよいかを決定する部品配置位置選択のアルゴリズムである．その大きな流れを以下に簡単に示す．

Step 0: Init (H);

$k=1$;

Step k: if $k \le n$ then begin

Set (i_k);

Update (H);

$k=k+1$;

end;

ここで，H は部品を配置する候補位置の集合であり，Init (H)によって候補位置の初期化を行い，Set (i_k)によって，部品 i_k を候補位置 H 内で最も望ましい位置に配置する．そしてUpdate (H)によって，候補位置の集合の更新を行う．このアルゴリズムによって解 σ に基づく配置が決まる．ここで用いたアルゴリズムを AL とすると，その解 σ は目的関数 $F(\sigma,AL)$ によって評価される．

その中で，最も重要なのは配置候補位置の集合 H であり，この違いにより以下の３種類の部分アルゴリズムを考える．

(1) AL1　既に配置されている部品群の頂点のみを配置候補位置の集合 H として考え，部品の配置を行う．

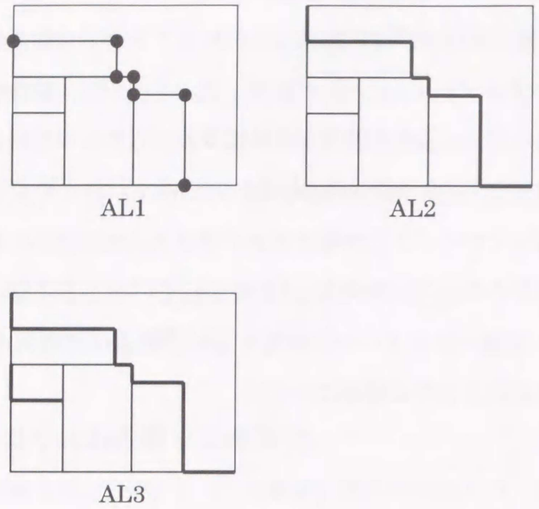


Fig.3.8 Series of allocation algorithm

(2) AL2　既に配置されている部品群の稜線上すべてを配置候補位置の集合Hとして考え，部品の配置を行う．

(3) AL3　AL2を拡張して，部品を配置することによって表れる中空領域内も配置候補位置の集合Hとして考え，部品の配置を行う．

ここで，AL2はskyline pack法であり，AL3は稜線法である．これらの探索空間の違いをイメージとしてFig.3.8に示す．これを見ると，AL3が配置候補位置の範囲としては最も広いことが分かる．

3.3.2　部分アルゴリズムの性能

ここでは3種類のアルゴリズムの性能を比較する．2.2.2において，skyline pack法(AL2)と稜線法(AL3)との比較は簡単に行ったが，確認の意味も含めて再度比較を行う．矩形部品$n=20$のデータを3種類用意し，それぞれランダムに解を6,000個生成し，その解に対して各アルゴリズム適用した．その結果の余材率の平均値をTable 3.6に示す．

Table 3.6 Performance of each algorithm

	Sample 1	Sample 2	Sample 3
AL1	26.96	26.93	30.41
AL2	22.89	20.95	29.49
AL3	22.43	20.40	29.36

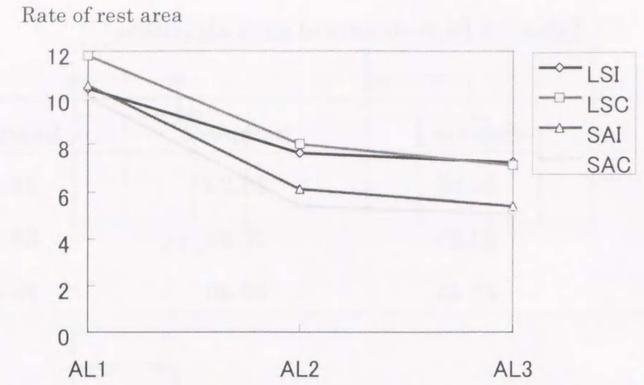
この結果を見ると，各アルゴリズムの性能はAL3>AL2>AL1の順であることが分かる．ここでは，AL2とAL3はわずかの差であるが，AL1は他の2つよりもかなり劣っていることが分かる．このことは，解の探索空間の広さが違っていることから予想されることである．

3.3.3　部分アルゴリズムの探索への影響

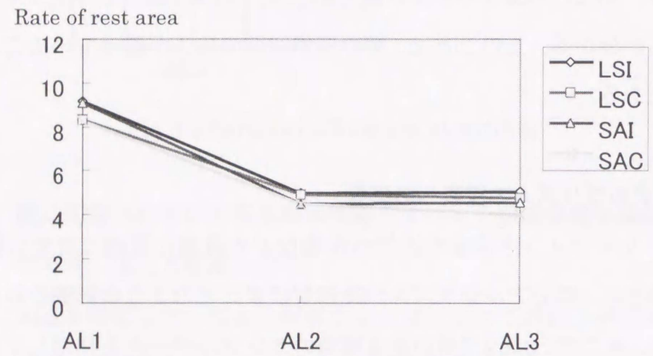
メタヒューリスティック手法や近傍の定義などの最適化問題に対する戦略的な部分を変えた場合に，部分アルゴリズムが最適解探索にどのような影響を与えるのかについて調べる．そこで，3.3.2で用いた3種類のサンプルデータを利用してシミュレーションを行った．メタヒューリスティック手法として局所探索とシミュレーテッド・アニーリングを用い，近傍として挿入操作と交換操作を用い，それぞれの組み合わせで計算を行った．その結果をFig.3.9に示す．それぞれ10回の計算を行い，見つけた最適解の余材率の平均を示している．

これを見ると，Table 3.6で示したアルゴリズムの性能とほぼ同じような結果が得られた．AL2とAL3は僅差であるがAL3が良い解をみつけており，AL1は他の2つとは見つける解にかなり開きがある．これは，近傍の定義やメタヒューリスティック手法を変えても同様の結果が得られた．

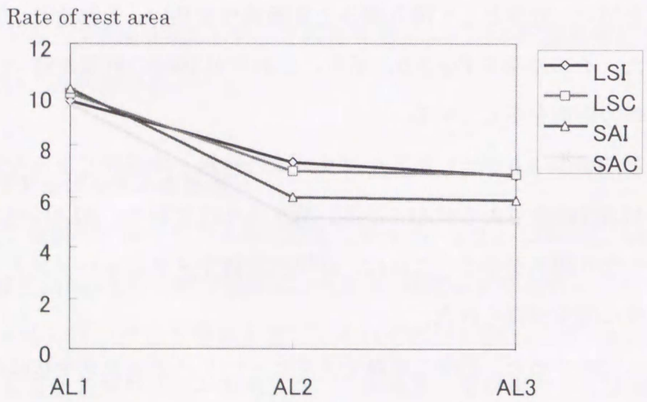
このことから判断すると，近傍の定義やメタヒューリスティック手法にほとんど影響されずに，部分アルゴリズムの性能が探索の効率にほぼ線形的に反映されることが



a) Sample 1



b) Sample 2



c) Sample 3

Fig.3.9 Search effects of partial algorithm

分かる．よって，その部分アルゴリズムの性能とそれに要する計算時間とのトレードオフをうまく考慮することで，問題の探索効率をコントロールすることが可能になる．

このことから，最適配置問題のような複雑な問題に対して，メタヒューリスティック手法を用いた最適化を考える場合に，ここで取っているように２つの部分問題に分けてアプローチすることにより，両方の側面からそれぞれ独立に効率化を図ることが可能である．そのことによって，より良い解決方法を見つけ易くできる．

第４章

適応型シミュレーテッド・アニーリングによる最適化

4.1　適応型シミュレーテッド・アニーリング (adaptive simulated annealing)

第３章では，最適配置問題に対して各種メタヒューリスティック手法を用い，最も適した手法はどれかについて考察を行った．適用した問題は違うが，これらメタヒューリスティック手法を比較した報告がこれまでになされており，概してシミュレーテッド・アニーリングとタブー探索の評価が高い[52)-56)]．第３章の結果では，最適配置問題において次のような問題点が浮かび上がってきた．シミュレーテッド・アニーリングについては，探索ステップ数が増えてくると局所探索的な要素が強くなり，探索効率の低下が起こるということが問題となった．タブー探索については，様々な探索オプションがあるが，どのオプションが問題に対して有効であるのかが分かりにくく，問題に対するコーディングが難しいということが問題となった．また遺伝アルゴリズムについては，交叉の操作が探索の中で機能しづらいことから，かなり問題に特化した操作を用意する必要があることが問題となった．よって，タブー探索及び遺伝アルゴリズムを用いる場合には，かなり問題に精通する必要がある．そして結果として，これらの中でも最も最適配置問題に適していると考えられる手法が，シミュレーテッド・アニーリングであった．

そこで，探索的手法であるメタヒューリスティック手法で，コーディングが比較的簡単でなおかつ性能の高いシミュレーテッド・アニーリングをベースにして，知識ベースを用いることなく，かつ様々な問題への適応性を持たせることが可能な適応型シミュレーテッド・アニーリングを提案する[25),57)]．この方法では，与えられた問題の性質を過去の探索から判断し，探索戦略を適応的に変えていく方法をとる．これによ

り，問題に依存せず，なおかつその特徴を生かした有効な探索が可能となる．これまでにもシミュレーテッド・アニーリングをベースとして改良を加えた手法として，アニーリングのプロセスを並列化させる手法[52]や探索近傍を縮約させる手法[58]などが提案されている．これらはただ計算の処理速度を上げることが目的であり，ここで提案する手法のように問題に対する適応性を持たせるといった観点には立っていない．

一般的にメタヒューリスティック手法で扱われる問題の解の表現形式は，

$$x=(i_1,i_2,.....,i_k,.....,i_n) \tag{4.1}$$

となる．この中の要素は文字列や数字列であるが，基本的に順序列として表現される．この順序列によって表される解$x,y(x,y\in\rho)$に様々な戦略を用いて解集合ρ内で変化させることによって，目的関数$F(x)$を最小，もしくは最大とするような最適解pが探索される．ここで，この要素i_kはおのおの実問題における対象を表現している．例えば，スケジューリング問題であれば，それは行わなければならない“仕事(work)”であり，巡回セールスマン問題であれば，それは巡回しなければならない“都市(city)”

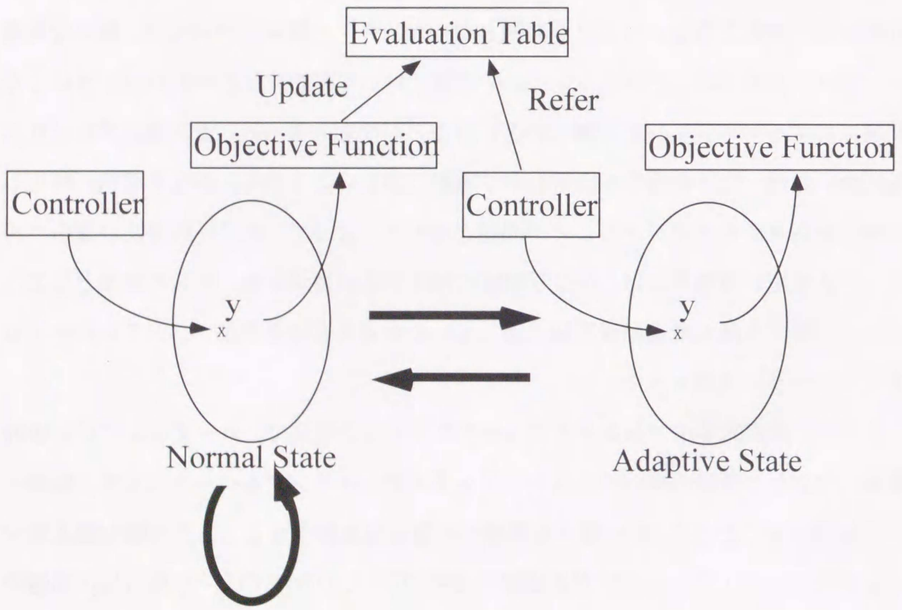


Fig.4.1 Conceptual diagram of adaptive simulated annealing

である．よって，それぞれの要素が問題に対して与える影響というものはおのおの違っていると考えられる．

通常，シミュレーテッド・アニーリングやタブー探索では探索を局所化するために近傍$N(x)$の定義を行う必要がある．一般的に，近傍の定義としては順列操作である挿入操作や交換操作などが用いられる．この近傍の中で，解候補はランダムに選択されるが，ここでは，その近傍内操作をランダムに行うのではなく，過去の探索履歴から各要素の問題における重み付けを行い，その結果を用いることで適応的に探索を行う手法を考える．基本的な探索の進め方はシミュレーテッド・アニーリングをベースとし，その枠組みの中で随時この適応的な探索を行う．この簡単な概念図を**Fig.4.1**に示す．

この図を見ると分かるように，この手法では通常状態(normal state)と適応的状態(adaptive state)の２つの状態が存在する．この状態は，探索における１つのステップに対応しており，１ステップ間は同一の状態に留まる．ここでは，どちらの状態でも探索解を決定するコントローラが存在するが，通常状態では特に指標はなく通常のシミュレーテッド・アニーリングと同様にランダムに探索解を決定する．しかし，ある条件が満たされると，通常状態から適応的状態に遷移し，このコントローラは過去の探索履歴の情報が蓄積されている適応度評価テーブル(evaluation table)を参照することで有効な探索が実現できるような探索解の選択を行うようになる．適応的状態に遷移すると，１ステップ間そこで留まり，次のステップには無条件で通常状態に戻る．通常状態では，再度ある条件が満たされるまで数ステップの間その状態で留まる．また，この間に目的関数の値を参照することによって，適応度評価テーブルが随時更新される．このように適応的状態に遷移することで，従来シミュレーテッド・アニーリングで問題であった点を改善する．

4.1.1 適応度評価テーブル (evaluation table)

最適化問題の中で扱われる各要素i_kが，その問題の中でどのような役割を果たしているのかを短い探索過程の中から明確に捉えるのは困難である．しかし，その要素が

問題の中での重要度といったものは探索履歴から朧気ながら抽出することは可能である．スケジューリング問題であれば，機械を占有する時間の長いような負荷の高いジョブや複数の機械を使用するようなジョブなどはその問題の中での影響度は高いであろうし，ここで考える最適配置問題であれば，組み合わせにくい部品や面積の大きい部品なども問題の中での影響度は高いと思われる．

そこで，このように問題の中で影響度が高いと思われる要素を探索履歴の中から抽出し，解探索に生かすようにする．そのための記憶領域として，適応度評価テーブル ET を用意する．このテーブルを次のように定義する．

$$ET[i_k] = w_k, \quad k = 1,.....,n \tag{4.2}$$

ここでは，問題の要素 i_k に対して，テーブル内の要素 w_k が１対１に対応している．ここでは，この値を適応度と呼び，この適応度が高い要素ほど問題の目的関数に対する影響度が高いものと考える．

また一般的に，探索を始める前にはそれぞれの要素が問題に対してどのような影響度を持っているのかは分からないことが多い．よって，すべての要素を対等に扱うのが普通である．そのため，初期状態では適応度評価テーブルの要素は等しくしておくことが望ましい．しかし，事前に何らかの条件が与えられているとか，経験的に分かっているようなことがあれば，それを考慮に入れてテーブルの初期化を行うことも考えられる．

4.1.2 適応度評価テーブルの更新

この手法はシミュレーテッド・アニーリングがベースとなっているが，通常状態においてコントローラはシミュレーテッド・アニーリングと同様にランダムに探索解を決定する．この通常状態において，目的関数による探索解の評価を元にして，適応度評価テーブル ET の更新を行う．更新の方法は様々なものが考えられるが，次の２点は有効な情報となる．

(1) 解を変化させたときに，暫定解が更新されるようなそれまでにない有効な解が得られた場合には，その操作に関与した要素の適応度を変化させる．

(2) 解を変化させたときに，現在得られている暫定解と同じ解が得られるような場合には，その操作に関与した要素の適応度を変化させる．

このように，探索の過程で解を変化させたときに，その解がどのような性質のものに変わったかを目的関数の評価から判断し，それを元に適応度評価テーブルの各要素の値を更新する．この場合，解を変化させる操作に関与した要素の適応度が更新されることになる．

また，目的関数の評価が極端に悪い結果となるような操作に対しても，関与した要素の適応度を変化させることも考えられる．これらのように，探索過程において特徴的な解が見つかった場合の操作は貴重な情報となる．

4.1.3 適応的状態への遷移条件

通常状態から適応的状態へと遷移することが，この手法の最も重要な部分である．適応的状態では，これまでの探索過程で作成された適応度評価テーブル ET を参照して適応的な探索を行うのであるが，その状態へ遷移するための条件を決める必要がある．そのための条件として，次のような条件が考えられる．

(1) 定期的に適応的状態へ遷移する．

(2) 何らかの条件が満たされれば，適応的状態へ遷移する．

適応的状態へ遷移するタイミングは，探索の効率に大きく影響すると考えられる．この手法で目的とすることは，シミュレーテッド・アニーリングで問題であった局所的な探索に陥りやすいという点を，この状態に遷移することで解決することである．そのために(2)の条件として，探索に進展が見られなくなった場合に遷移するようなことが考えられる．

4.1.4 適応的状態での探索解の決定

適応的状態に遷移すると，適応度評価テーブル ET 内の各要素の適応度を参照することによってある要素を選択し，コントローラが探索解を決定する．つまり，その要素を用いて解を変化させるのである．ここでは，適応的状態でどのような探索を実現

したいのか，つまり局所的な探索を実現するのか大域的な探索を実現するのかによって解の選択方法が変わってくる．また，適応度評価テーブルがどのような更新条件で作成されたのかによっても選択方法が変わってくる．

一般的に考えられるのは，適応度に応じて確率的に選択する方法や，適応度に応じて各要素に順序付けを行い，その順序に応じて選択する方法などが考えられる．ここでは，要素i_kが選択される確率$SL(i_k)$として次のような手法を考える．

(1) 昇順選択

$$SL(i_k)=\left(ET[i_k]-ET[L]\right)\Big/\sum_{j=1}^{n}\left(ET[i_j]-ET[L]\right) \tag{4.3}$$

ここで，$ET[L]$は適応度評価テーブルの中で最も適応度の低い要素の適応度である．適応度の高い要素が高い確率で選択される．

(2) 降順選択

$$SL(i_k)=\left(ET[H]-ET[i_k]\right)\Big/\sum_{j=1}^{n}\left(ET[H]-ET[i_j]\right) \tag{4.4}$$

ここで，$ET[H]$は適応度評価テーブルの中で最も適応度の高い要素の適応度である．適応度の低い要素が高い確率で選択される．

(3) ランク選択

$$SL(i_k)=RA\left(rnk\left(ET[i_k]\right)\right) \tag{4.5}$$

ここで，$rnk\left(ET[i_k]\right)$は要素i_kの適応度のランキングである．あらかじめランキングによって選択される確率はRAによって決定されており，それに従って選択される．よって，適応度の高い要素と低い要素を同時に選択確率を高くするようなことも可能である．

このようにして，ランダムに探索解を選択するのではなく，適応度に応じて要素を選択し，その要素を動かすような近傍操作を行うことで探索解が決定される．また，常にある一定の選択方法を取るのではなく，解探索の状況に応じて選択方法を変化させることも考えられる．局所状態に陥り，探索がうまく進まなくなった場合には，問題

に対する影響度の高い要素を優先的に選択し，解を大きく変動させることでその状態から抜け出そうとしたり，探索の最終段階では，問題に対する影響度の低い要素を優先的に選択し，その周辺を集中的に探索したりすることも考えられる．

4.1.5 適応型シミュレーテッド・アニーリングのアルゴリズム

ここまでで，適応型シミュレーテッド・アニーリングを行うために必要な構成要素とその役割について説明した．以下にその具体的なアルゴリズムを示す．ここではPascal風の記述を用いる．ここでの目的は，目的関数Fを最小化することである．

```
Step 0: Set(t0, T(0), γ, L, N, S);
        Init(ET);
        x:= select(ρ);
        p:= x;
        k:= 0;
Step k: if S is False then begin
        {* normal state *}
            repeat
                y := select(N(x));
                Δ := F(y) − F(x);
                if Δ ≤ 0 then begin
                        x := y;
                        if F(y) < F(p) then p := y;
                        Update(ET);
                    end
                else if prob( e^(−Δ/T(k)) ) then x := y;
            until L;
        end
```

```
else begin
{* adaptive state *}
        repeat
            y := select(N(x) referring to ET);
            if F(y) < F(x) then x := y;
            if F(y) < F(p) then p := y;
        until L;
    end;
k := k + 1;
T(k) = γT(k − 1);
if T(k) ≤ t_0 then return p;
else Confirm(S);
```

ここで，$Init(ET)$は適応度評価テーブルの初期化であり，$Update(ET)$はテーブルの更新である．また，探索の状態をSで表しており，$Confirm(S)$によってその状態の確認が行われ，適応的状態への遷移条件が満たされればSはTrueとなり，そうでない時もしくは適応的状態を１ステップ終えた時にはSはFalseとなる．そして，$select(N(x))$では近傍内からランダムに探索解を選択する通常状態のコントローラの役割を表しており，$select(N(x)\ referring\ to\ ET)$では適応度評価テーブル$ET$を参照して探索解を決定する適応的状態のコントローラの役割を表している．またその他のパラメータは，一般的なシミュレーテッド・アニーリングのパラメータである．

4.2 シミュレーション結果

適応型シミュレーテッド・アニーリングの有効性をシミュレーションにより検証する．第２章で説明した最適配置問題で，配置する部品が任意多角形状部品に対して適用する．また，適応型シミュレーテッド・アニーリング，局所探索，シミュレーテッド・アニーリング及びタブー探索の各シミュレーションで用いる探索近傍として，第

３章で評価の高かった１ブロック交換操作を用いる．ここでは，まず適応型シミュレーテッド・アニーリングの構成について検討する．どのような構成にすれば，最も効率の良い探索が実現できるかを調べる．その後で，他のメタヒューリスティック手法との比較を行う．また，シミュレーテッド・アニーリングで問題であった局所的な探索に陥りやすいという点についても解決できるかどうかを検証する．

4.2.1 適応型シミュレーテッド・アニーリングの設定

本手法では，適応度評価テーブルをどのように構成するか，つまりテーブルの更新をどのように行うかによって，適応的動作がうまく機能するかどうかが決まる．いかにうまくそれぞれの要素が問題の中で持っている特性を掴むかがカギとなる．また，適応的状態への遷移条件をどのように設定するかによっても，探索がうまく進むかどうかが決まる．よってここでは，適応度評価テーブルの更新方法と適応的状態への遷移条件の違いによって最適解探索にどのような影響があるのかについてシミュレーションにより見る．適応的状態での探索解の決定方法として，昇順選択を用いる．よって，テーブル内で適応度の高い要素が確率的に選択される．

そこで，以下の５つの場合について比較を行う．

SABM: テーブルの更新手法として，現在の暫定解よりも良い解が得られた場合に，交換した２つの要素の適応度を１上げ，暫定解と同じ解が得られた場合には交換した２つの要素の適応度を１下げるように更新を行う．また適応的状態への遷移条件として，暫定解が４ステップ更新されなければ適応的状態へ遷移するようにする．

SABS: テーブルの更新はSABMと同様に行い，適応的状態への遷移条件として，暫定解が２ステップ更新されなければ適応的状態へ遷移するようにする．

SAUM: テーブルの更新手法として，現在の暫定解よりも良い解が得られた場合に，交換した２つの要素の適応度を１上げるように更新を行う．また適応的状態への遷移条件はSABMと同じ条件を用いる．

SADM: テーブルの更新手法として，暫定解と同じ解が得られた場合には交換した２

つの要素の適応度を１上げるように更新を行う．また適応的状態への遷移条件はSABMと同じ条件を用いる．

SANM: テーブルの更新手法として，暫定解と同じ解が得られた場合には交換した２つの要素の適応度を１下げるように更新を行う．また適応的状態への遷移条件はSABMと同じ条件を用いる．

また，初期条件としての情報は何もなく，探索の過程の中でそれぞれの部品の適応度を決定する．よって，初期状態ではどの部品も適応度は均一であり，ここでは０であるとする．そして，ここで述べている暫定解と同じ解とは，目的関数の評価値が同じ解のことであり，配置のパターンは異なってもかまわないものとする．

4.2.2 適応型シミュレーテッド・アニーリングの評価

4.2.1で設定した条件でシミュレーションを行った．シミュレーションに用いたデータは，部品数$n=20$のデータを２種類である．またベースとなるシミュレーテッド・アニーリングのパラメータは，$t_0=40$，$T(0)=1000$，$\gamma=0.85$，$L=100$とした．これらの設定で，それぞれのテーブルを用いた場合で10回のシミュレーションを行った．ここで探索を始める解の初期値はその都度ランダムに生成することとした．そして，得られた最適配置の余材率の平均値とそれらの中での最良値を見る．また，解探索の効率を見るために，その10個の標本に基づく標準偏差についても見る．ここで得られた結果を**Table 4.1**に示す．

この結果を見ると，適応度評価テーブルの更新手法としては，SABMの方法が最も良い結果が得られることが分かる．ここで扱っている最適配置問題では，面積の小さな部品は配置順序を変えても全体の配置に与える影響がほとんどない場合が考えられる．逆に形状の複雑な部品や面積の大きな部品は，その順序を変えると全体の配置がまったく変わってしまうことが考えられる．ここでは，新たな暫定解が見つかった場合には操作した要素の適応度を上げ，暫定解と同じ解が得られた場合には操作した要素の適応度を下げることで，その配置順序を変えることによって問題に与える影響が大きな要素をうまく捉えることに成功している．よって，局所的な探索に陥りそう

Table 4.1 Results of adaptive simulated annealing

a) Data 1

	SABM	SABS	SAUM	SADM	SANM
Average	17.150	18.258	19.475	18.293	18.449
Best value	14.533	13.375	16.258	16.226	15.078
Deviation	1.142	2.028	1.966	1.570	2.074

b) Data 2

	SABM	SABS	SAUM	SADM	SANM
Average	18.198	18.785	19.087	18.446	18.841
Best value	15.733	16.676	14.536	16.730	16.185
Deviation	0.934	1.461	2.403	1.330	1.390

なときに，適応的状態に遷移し，近傍内で問題に対して影響が大きな要素を動かすことで解をより大きく変動させることによって，そのような状態から抜け出すことに成功していると考えられる．また，SADMも比較的良い結果が得られているが，これはSABMとは逆にあまり問題に対して影響力のない要素を捕らえ，暫定解の周辺を集中的に局所的な探索が行えている結果であろう．このことは，最良値としてはそれほど良い解を見つけているわけではないが，平均的に良い解を見つけていることからそう判断できる．また，SAUMはそれほど良い結果が得られていないが，これはテーブルを更新する条件として用いた暫定解が更新されるような良い解が見つかるということがそれほど頻繁に起こることではないため，十分な適応度評価テーブルが作成できなかったためだと考えられる．それとは逆にSANMでは，影響が小さいであろうと思われる要素の適応度を下げるだけで十分な適応度評価テーブルが作成できるかどうか試みたが，結果としてそれだけでも不十分であることが分かった．そういった意味で，探索の過程でどのような要素を捉えたいのかをはっきりさせ，それを１つの更新条件だけでなく，いろいろな条件を組み合わせて適応度評価テーブルを作成することで，効果的なテーブルが作成できるものと考えられる．そのときに，その更新条件が

満たされる頻度についても考慮する必要がある．

次に，適応的状態への遷移条件であるが，SABSのように適応的状態へ頻繁に遷移すると，本来のシミュレーテッド・アニーリングの良さが消されてしまい，偶然良い解を見つけることはあっても，偏差の値を見ても分かるように全体的なパフォーマンスは低下する．ここで行ったシミュレーションでは，設定したパラメータを用いると全体のステップ数が20となる．そのため，SABMの場合に4ステップ暫定解の更新がなければ適応的状態へ遷移するように設定したが，適用する問題に応じて妥当な条件を見つける必要がある．このことは，どこで計算を止めるのかという温度パラメータの設定とも関係してくるであろう．

4.2.3 他の戦略との比較

メタヒューリスティック手法として様々な戦略が存在するが，それらと適応型シミュレーテッド・アニーリングとの比較を行う．ここでは局所探索(LS)，シミュレーテッド・アニーリング(SA)，タブー探索(TS)，遺伝アルゴリズム(GA)，ランダム探索(RS)との比較を行う．部品数$n=20$のデータを6種類用意し，適応型シミュレーテッド・アニーリングとして4.2.2で最も結果の良かったSABMを用いて比較を行った．そして各メタヒューリスティック手法で用いたパラメータであるが，局所探索では最大探索回数$lsn=15$とした．シミュレーテッド・アニーリングは，4.2.2と同じパラメータを用いた．タブー探索では，タブーリストの長さ$sl=3$，最大探索回数$tsn=15$とした．タブーリストの構成は，各ステップで見つけた最良解を得るためのムーブ（交換した要素）をタブーリストに加える．よってタブーリストのサイズが3であるということは，最大6個の要素がタブーリストに加えられることになる．また，第3章では効果が見られなかったが，アスピレーション基準も導入する．この戦略では，タブーリストの中で問題に対して影響が大きいと考えられる面積の大きな部品に対応する要素を2つリストから外すようにしている．遺伝アルゴリズムについては，2種類のパターンを比較に用いた．その1つであるGA1として，世代数$gn=50$，個体数$gm=40$とし，交叉確率$P_{Crossover}=0.9$，突然変異確率$P_{Mutation}=0.1$とした．もう1つのGA2は，

Table 4.2 Comparison of other meta-heuristics

a) Data 1

	LS	SA	SABM	GA1	GA2	TS	RS
Average	18.887	17.670	17.150	18.849	18.359	19.489	22.796
Best value	16.664	16.417	14.533	16.688	16.282	16.036	19.045
Deviation	1.139	0.954	1.142	1.383	1.479	1.808	1.691

b) Data 2

	LS	SA	SABM	GA1	GA2	TS	RS
Average	20.417	19.203	18.198	19.047	18.567	20.324	20.254
Best value	17.469	17.035	15.733	17.215	16.512	17.163	18.545
Deviation	1.720	1.502	0.934	1.458	1.521	2.039	1.160

c) Data 3

	LS	SA	SABM	GA1	GA2	TS	RS
Average	17.825	17.491	16.743	18.464	17.940	18.845	19.774
Best value	14.683	15.526	15.415	16.854	17.483	16.920	18.576
Deviation	1.705	1.544	0.830	0.857	1.888	1.455	0.808

d) Data 4

	LS	SA	SABM	GA1	GA2	TS	RS
Average	16.773	15.443	14.956	15.696	16.301	15.956	17.573
Best value	15.124	14.223	14.144	13.486	14.550	12.920	16.067
Deviation	1.376	0.731	0.677	1.391	0.832	2.047	0.835

e) Data 5

	LS	SA	SABM	GA1	GA2	TS	RS
Average	19.476	17.806	17.357	17.328	18.142	19.572	19.031
Best value	16.221	16.059	15.183	14.690	17.006	17.449	17.346
Deviation	1.838	1.014	1.193	1.455	0.879	1.650	1.195

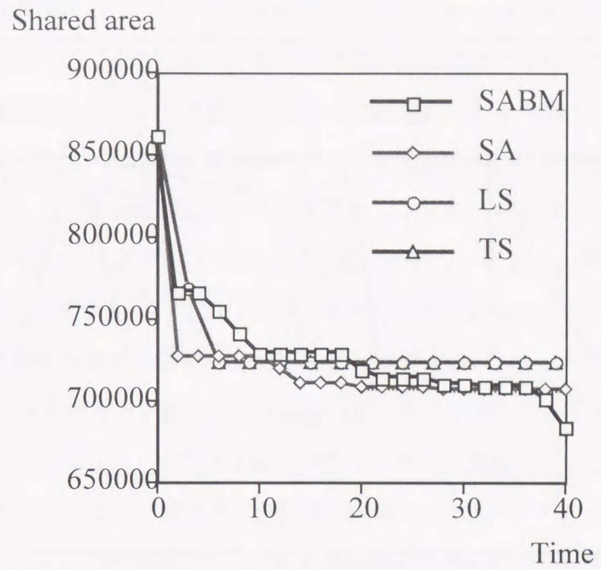
f) Data 6

	LS	SA	SABM	GA1	GA2	TS	RS
Average	19.096	17.127	15.446	16.823	17.711	18.430	18.845
Best value	17.426	13.717	13.434	14.104	15.927	15.847	16.276
Deviation	1.498	1.553	1.635	1.745	1.292	2.005	1.260

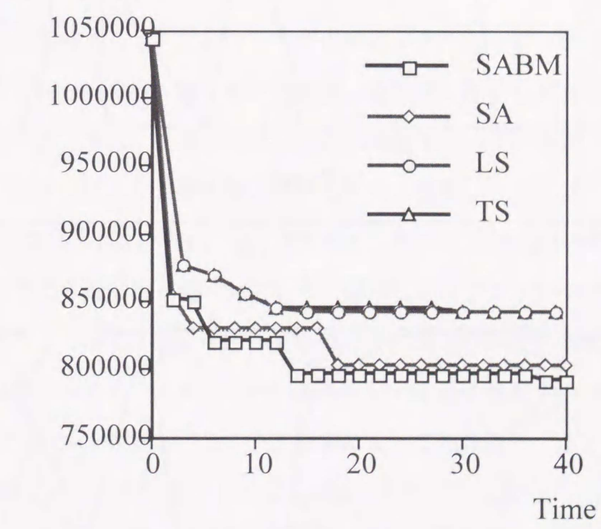
世代数，個体数はGA1と同じで，交叉確率 $P_{Crossover} = 0.1$，突然変異確率 $P_{Mutation} = 0.9$ とした．そして，交叉の操作として，順序交叉を取り，突然変異の操作として交換の操作を取った．また，再生の操作の中でエリート保存戦略も用いている．そして，メタヒューリスティック手法以外の方法として，ランダム探索(RS)では，ランダム探索数 $rsn = 2000$ とした．また，それぞれの手法で探索を始める解の初期値であるが，その都度ランダムに生成することとした．ここで設定しているパラメータ値は，どの手法でも終了までの計算時間がほぼ同じとなるようにしている．局所探索だけは，解が更新されなくなるとそこで探索を終了するという定義のため，この手法だけは探索によって計算時間にばらつきが生じている．

ここで，シミュレーションで得られた結果を**Table 4.2**に示す．また，それぞれの手法で同じ初期解から探索を開始した場合の計算時間から見た解探索の過程を**Fig.4.2**に示す．ここでは，探索の枠組みが似ている局所探索，シミュレーテッド・アニーリング，タブー探索，そして適応型シミュレーテッド・アニーリングとの比較を行っており，見つけた最良解の遷移を示している．局所探索については，探索が途中で終了したが，最後に得られた解を延長して表示している．

Table 4.2において，それぞれの手法の有効性を示しているのは見つけた最良解の平均値及びその標準偏差であると考えられる．その項目を見ると，適応型シミュレーテッド・アニーリング(SABM)が他の手法と比較して最適解探索に最も有効であることが分かる．10回の探索で見つけた最良値も評価の対象となるが，ここでは初期値をランダムに選択しているため，どのような初期値から出発するかという偶然性もある．

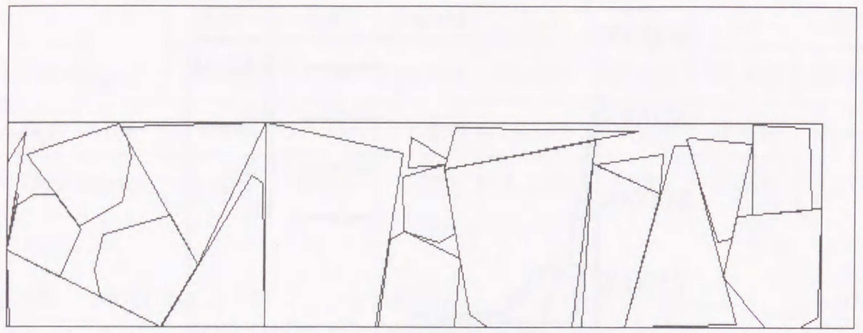


a) Data1

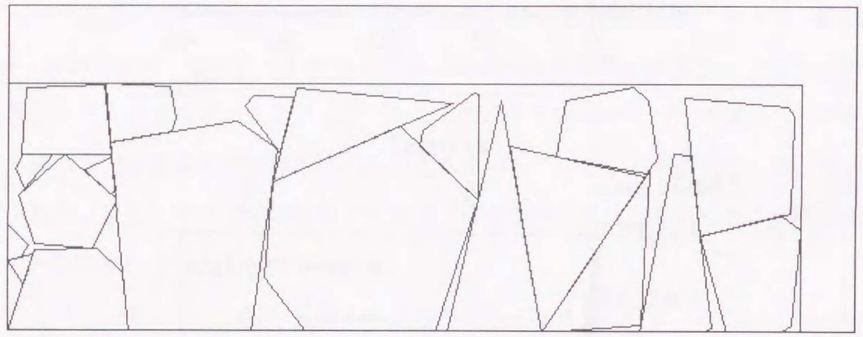


b) Data2

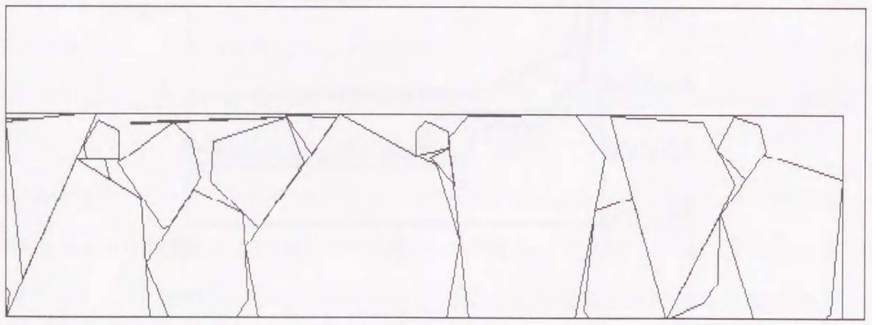
Fig.4.2 Comparison of meta-heuristics search techniques



a) Data1



b) Data2



c) Data5

Fig.4.3 Simulation results by ASA

例えばData3の場合のように，局所探索がこれらの手法の中で最も良い解を見つけるようなことが起こる．そのため，これだけで簡単に比較は行えないが，突然良い解を見つける可能性があるという評価は行える．これらのことを考慮しても，適応型シミュレーテッド・アニーリングは良い解を見つける可能性という点でも優れていることが読みとれる．このように適応型シミュレーテッド・アニーリングでは，配置する部品のデータがどのようなデータであっても常に一定レベル以上の結果を得ることができている．このことから，問題によって扱う部品の形状がまったく異なっていても，その問題に適応した探索が実現できていることが言える．

また，Fig.4.2を見ると，適応型シミュレーテッド・アニーリングでは，定常状態に陥りそうになった場合に，適応的状態での探索が有効に機能していることも分かる．このことによって，着実に暫定解の更新が行われている．ここでもシミュレーテッド・アニーリングは，温度パラメータが大きい探索初期では着実に暫定解を更新しているが，温度パラメータが小さくなると，定常状態に陥り，そこから抜け出すことが困難であることが分かる．

適応型シミュレーテッド・アニーリング以外のメタヒューリスティック手法について，ここで得られた結果は，第３章で矩形部品を配置対象として得られた結果とほぼ同様となった．ここでは，任意多角形状部品の配置を行っているが，ここでもシミュレーテッド・アニーリングが最も優れているという結果が出ている．結果の良い順に並べるとSA，GA2，GA1，TS，LS，RSであった．ここでGA1とGA2及びTSとLSについては似たような結果が得られている．使用したオプションやパラメータ値がほぼ同じであったことから，配置対象つまり部分アルゴリズムである配置位置選択のアルゴリズムが変わっても，メタヒューリスティック手法の性能にはほとんど影響がなく，それぞれの手法の中でのパラメータ設定によって性能を上げることが期待できる．よって，3.3節で示した独立性がここでも実証されたことになる．

また意外であったのがランダム探索の標準偏差がどのデータの結果でも小さかったことである．すなわち，突然良い解を見つける可能性は非常に小さいということである．この結果は，3.2節で矩形部品を配置したシミュレーションの場合でも同様で

ある．このことは，ランダム性だけでは探索が有効に機能しないことを物語っている．よって，シミュレーテッド・アニーリングのようにランダム性と近傍探索をうまく組み合わせることで，それらの性質が生きるのだと思われる．

第５章

適応的タブー探索による最適化

5.1 適応的タブー探索 (adaptive tabu search)

第３章のシミュレーションでは，タブー探索はあまり良い結果は得られず，そのベースである局所探索と同程度のパフォーマンスしか得られなかった．このことは，タブー探索のオプションである様々なメモリーを十分に活用できていないことで局所的な状態から抜け出せていないことに起因している．タブー探索の戦略として探索を高速化させるために並列化する手法などが提案されている[52],[59]が，タブー探索をうまく機能させるためには，様々なメモリーをどのような構成にするのかが，最も重要である．このメモリーを多用することにより，探索領域にある種の制約を課して，有効な探索へと導こうとすることが不可欠である．これまでに様々な問題にタブー探索は適用されているが，このメモリーの要素として用いられているものは，近傍の定義である挿入操作や交換操作のムーブや，至近性や頻度などの解の属性及び良解を与える順列としての解そのものであったりした[32],[42],[60]．これらは，情報としては有意であるのだが，制約としては小さくなるため，探索回数を大きくしなければ効果が得られないという問題が起こる．また，このように様々なメモリーを用いることにより，構造が複雑になり，問題への適用が非常に困難となってしまう．よって，効果的な探索を実現するためには，多大な労力を必要とした．

タブー探索で用いられるメモリーとしては，タブーリスト，アスピレーション基準，中期記憶，長期記憶などが一般的である．タブー探索は近傍探索であるため，ある解から別の解へ到達するためのムーブを定義し，このムーブに対して各種戦略が取られる．このムーブには，非常に多くの組み合わせが考えられる．一般的に，最適化問題で扱われる個々の対象は，それ自体がある意味を持ったものであることは第４章でも

述べた．例えば，ある対象１つが別のものに変わってしまったとき，最適解はまったく違ったものになると思われる場合と，それほど大きな変化がないと思われる場合がある．このことから，ムーブで得られるような対象相互の関係の評価や至近性や頻度などの解全体の評価も重要であるが，個々の対象だけに着目しても，十分評価に値すると思われる．よって，従来のようにムーブや頻度などをメモリーの要素として用いるのではなく，問題で扱われる対象をメモリーの要素として用いても機能することが考えられる．実際に人がこのような問題に直面した場合でも，まずその対象に着目して解決法を見い出そうとするであろう．このことにより，問題をシンプルに捉えられるようになり，見通しも立ちやすくなる．

そこで本論文では，ムーブに対してではなく，問題で扱われる個々の対象に対して各種戦略が取られるようにする適応的タブー探索を提案する[61]．そのために，それぞ

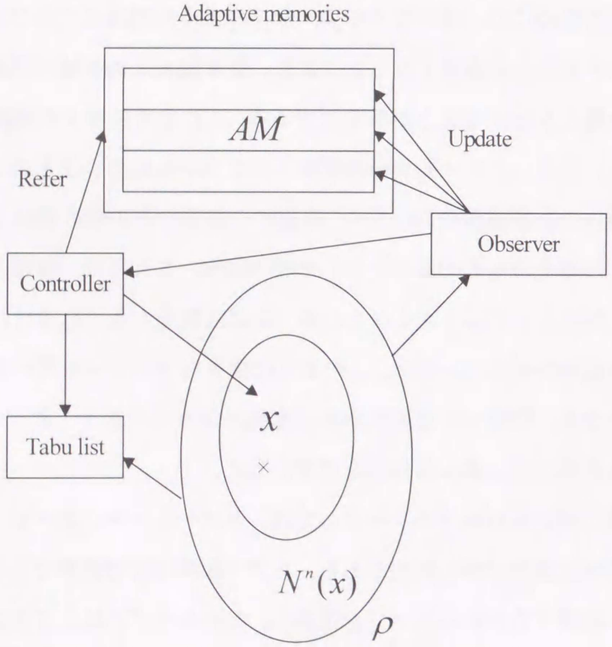


Fig.5.1 Concept of adaptive tabu search

れの対象に対して，探索履歴から様々な指標における評価付けを行い，その値を適応度評価メモリー AM (adaptive memories)に記憶しておく．ここでは，複数の指標による評価を保持することで，状況に応じて様々な戦略が取れるようにする．このことにより，簡便な探索が行えるようになる．その適応的タブー探索の構成図を**Fig.5.1**に示す．

この図を見ても分かるように，構成は非常に単純である．基本的にメモリーとして用いられるものは，タブー集合の役割を果たすタブーリストと複数のリストで構成される適応度評価メモリーだけである．ここでは，解 x は

$$x=\left\{b_1,b_2,.....,b_i,...,b_j,.....,b_n\right\}\left(\in\rho\right) \tag{5.1}$$

として，問題で扱われる対象による順列の集合 ρ の要素として表現できるとする．この要素に対して近傍 $N(x)$ が定義される．

ここで提案する手法では，常にオブザーバーが探索履歴を監視しており，状況に応じて適応度評価メモリーを更新する．また，コントローラによって，探索の方向が決定される．つまり，どのようなムーブを用いて近傍探索を行うかが決定される．ここでは，コントローラが探索の進捗状況をオブザーバーと情報交換し，適応度評価メモリーやタブーリストを参照して探索解を決定する．このことにより，状況に応じて適応的に探索戦略を取ることが可能となる．

5.1.1 ムーブの拡張

タブー探索のような近傍探索をベースとする手法では，近傍の定義が必要である．適応的タブー探索では，適応度評価メモリー AM を参照して探索解を決定するメモリーモード（memory mode）とタブーリストを参照してタブーリストに含まれないムーブによる近傍内を探索するタブーモード（tabu mode）とがある．このメモリーモードの場合にも，探索するムーブを選択できる近傍を決める必要がある．ここで，この近傍をタブーモードと同様の近傍にしてしまうと，タブーモードで探索される範囲と変わらなくなってしまい，メモリーモードの意義が失われてしまう．これまで，アスピレーション基準がうまく機能していなかったのも，ここに原因があると考えられ

る．そこで，メモリーモードでのムーブの選択範囲を拡張するために，一般的な近傍の定義の拡張を行う．その近傍が，メモリーモードでムーブを選択できる近傍であるとする．

ある解xが存在したときに，その近傍を$N(x)(\in\rho)$とする．また，この近傍内に含まれる要素数をnum_1とする．この近傍に含まれるすべての要素の近傍を解xの２階近傍として

$$N^2(x)=\bigcup_{y\in N(x)}N(y) \tag{5.2}$$

と定義する．同様にして，解xのｎ解近傍として

$$N^n(x)=\bigcup_{y\in N(x)}N^{n-1}(y),\quad n=2,3,... \tag{5.3}$$

が定義できる．ここで，この近傍内の要素数はnum_nとする．そうすると，ムーブmについても次のように定義できる．

$$m=(x\to y),\quad y\in N(x) \tag{5.4}$$

また，近傍の場合と同様にしてムーブの拡張として

$$m^n=(x\to z),\quad z\in N^n(x) \tag{5.5}$$

が定義できる．

実際に近傍としてよく用いられるものとして，挿入操作や交換操作がある．この場合にムーブは，挿入されたり交換されたりしてその操作に関与した要素の組$(b_i,...,b_j)$もしくはその位置の組$(i,...,j)$として記憶される．このムーブの評価は，目的関数などによって決定される．ここで拡張した近傍は，基本となる近傍操作を複数回行うことでその領域を広げている．よって拡張されたムーブも，基本的なムーブが組み合わせられたものとなる．

5.1.2　タブーリスト

適応的タブー探索の中で，タブー集合として機能するタブーリストTLについて説明する．これはサイクリックな探索，すなわち同じ解を重複して探索することを防ぐ

ために設けることが基本的な概念である．しかし，これを積極的に活用することで，より効率の良い探索が可能となる．あまりにタブー集合の領域が狭いと，局所探索と変わらなくなり，探索効率が悪くなる．よって，むしろタブー集合によって探索が禁止される領域を大きく取るほうが，タブー探索としてうまく機能する．これは，シミュレーテッド・アニーリングにおいて，１ステップの反復回数を少なく保つことが望ましいことと同義である．例えば，交換操作において，ムーブ(b_i,b_j)によって暫定解が見つかったとする．このとき，このムーブが意味するこれら２つの要素の交換を禁止するだけでなく，どちらか一方を含むような交換を禁止するようにする．そうすると，タブーリストTLは次のように定義される．

$$TL=\bigcup_{i=k-sl+1}^{k}L_i,\quad k=1,....,n \tag{5.6}$$

ここで，L_iはi番目の探索で暫定解となった解を見つけ出したムーブに関与した要素のリストである．またslはタブーリストサイズであり，どれくらい過去の履歴までを保持するかを示している．このように，ムーブ単位で禁止するのではなく，要素単位で禁止するようにする．そうすることによって，近傍内でのタブー集合で禁止される領域は広がる．このため，実際の探索領域は狭くなるが，同じ計算時間でも探索回数を増やすことができる．

また，タブーリストサイズslであるが，探索中は長さを変えない静的なものと，状況に応じて長さを変える動的なものとが考えられる．この長さを制御することによって，タブーとなる領域を増やすことも可能であるが，一般的に静的なリストが用いられる．

5.1.3　アスピレーション基準

タブー集合に記憶されているために，実行できないムーブがあるが，そのようなムーブであっても良い解が得られそうであれば実行できるようにするのがアスピレーション基準である．これまで，比較のためにタブー探索を用いたシミュレーションを行ってきたが，その中でアスピレーション基準とタブーリストを戦略として利用して

いた．しかし，結果としてその効果はほとんど得られなかった．探索回数が少なかったこともその原因の１つであると思われるが，アスピレーション基準という戦略にも問題がある．禁止領域の中から有望そうな解を見い出すというアイデアは良いが，現実的に考えると，禁止領域にそのような解がどれくらいの確率であるのか，またその解をうまく拾い出せるのかという問題がある．そういった意味で，この戦略が有効に働く可能性は低いと考えられる．実際にこれまでのシミュレーションでも局所探索の域を抜け出せていないという結果が出ている．この戦略により，タブー集合による制約は緩むが，緩んだとしてもそれは結局局所探索の延長でしかないということになる．よって，それほど重要な戦略であるとは考えられない．

このことから，ここで提案する適応型タブー探索において，アスピレーション基準という戦略は取らない．この戦略により，タブーリストによる探索領域の制約を緩めるようなことはせず，制約はそのままにして，同じ計算時間で探索回数を増やすほうがタブー探索の特徴が生きると考えられる．このことにより，ここで提案する適応度評価メモリーを頻繁に活用することが可能となる．

5.1.4　適応度評価メモリー（adaptive memories）

タブー探索において，様々な戦略を取るために必要なメモリーとして，適応的タブー探索では適応度評価メモリーAMを用いる．このメモリーは複数のリストltによって構成され，このリストごとに問題で扱われる対象に対して様々な指標による適応度を探索履歴から決定する．その構成は次のようなものである．

$$AM=\{lt_1,lt_2,.....,lt_i,.....,lt_{mi}\}$$
$$lt_i=\{w_{i,1},w_{i,2},.....,w_{i,k},.....,w_{i,n}\} \tag{5.7}$$

ここで，lt_iがある指標による対象に対しての適応度を表したリストであり，その要素である$w_{i,k}$が対象b_kの適応度を表している．またmiは，この探索で用いる指標の最大数である．このメモリーの更新はオブザーバーによって行われる．またコントローラは，メモリーモードにおいてこのメモリーを参照することで探索を行う解の選択を行う．ここで用いるリストは非常に簡単な構成をしており，問題で扱う対象個々に重み

付けをするだけのものである．

タブー探索では，頻度メモリーを用いた中期記憶とか長期記憶といった戦略を取ることが一般的である．適応度評価メモリーのリストは，それぞれが意味を持っており，指標をうまく選択することで，これらの戦略と同様の効果を得ることが可能である．基本的に中期記憶は局所的な探索を実現する戦略であり，長期記憶は大域的な探索を実現する戦略である．それぞれの特徴をうまく捉えたリストを構成することで，局所的探索や大域的探索は実現可能である．

5.1.5　オブザーバー

オブザーバーでは，探索過程を観測し，特徴的な解が得られるようなことがあると，適応度評価メモリーAMの更新を行う．ここでは，それぞれのリストを更新するための条件と，そのリストの更新の方法を決定する必要がある．リストが持つ指標がうまく表現されるような更新条件を見つけなければならない．

まず，更新の条件として次のような場合が考えられる．

(1)　暫定解よりも良い解が得られた場合

(2)　暫定解と同じ目的関数値となる解が得られた場合

(3)　ある閾値を越えるような解が得られた場合

この(3)で言う閾値は問題によって様々考えられるが，一般的に目的関数値で客観的に判定を行う．

また，リストの更新方法であるが，(1)〜(3)までの条件のどれかに該当するような解が得られた場合に，その時にムーブとなった要素に対して正の重み付けを行ったり，負の重み付けを行ったりすることが考えられる．さらに，これらの条件を組み合わせた更新方法を取ることも可能である．

ここで出した条件の中で，(1)及び(3)においては，問題に対しての影響度が大きいと思われる対象が捉えられるので，これに正の重み付けを行い，このリストを有効に活用することで，長期記憶的な探索が可能となる．また同様に，(2)においては，問題に対しての影響度が小さいと思われる対象が捉えられるので，これに正の重み付けを

行ったリストを活用することで，中期記憶的な探索も可能となる．これらのことは，第４章の中で適応型シミュレーテッド・アニーリングに用いた適応度評価テーブルの効果から想像できることである．

またこのリストを，適応的タブー探索では用いないアスピレーション基準に利用することも考えられる．つまり，リストを参照して，問題に対して影響度が大きいと思われる要素はタブーリストの中から外すといった利用方法である．

さらにこのオブザーバーのもう１つの働きとして，現在の探索の状況をコントローラに報告するという働きがある．この情報により，コントローラが探索モードを決定し，探索解を決定する．また，このオブザーバーからの情報によって，コントローラが探索の終了時期も決定する．

5.1.6 コントローラ

適応的タブー探索では，２つのモードを設定している．１つはタブーリストTLを参照して，解xの近傍$N(x)-TL$内を探索するタブーモードであり，もう１つは適応度評価メモリーAMを参照して，戦略的な探索を行うメモリーモードである．

タブーモードでは，一般的なタブー探索と同様にタブーリストに入っていない近傍$N(x)$内すべての解の探索を行い，その中での最良解の近傍を探索する次のステップに進むようになる．またメモリーモードでは，適応度評価メモリーを用いて探索戦略を立てるのであるが，ここでは探索がどのような状態にあるときにメモリーを参照するようにするのか，そしてその際にどのリストを参照するのかを決定しなければならない．ここでは，参照条件として次のような条件を考える．

(1) 探索が局所的な状態に陥り，うまく進まなくなった時に参照する．

(2) ある一定間隔で定期的に参照する．

(3) 暫定解が更新された時に参照する．

これらの条件は，適応度評価メモリー上のリストそれぞれに対して決定する必要がある．リストが持っている指標を有効に活用できるように決定する必要がある．

次に，メモリー上のリストを参照して戦略的に探索解を決定しなければならない．

この場合，対象に対応するリスト要素の適応度によって，確率的にムーブの要素として選択する方法がある．つまり，メモリー内のリストlt_kを参照した時に，対象b_iがムーブの要素として選択される確率$Ps_{k,j}$は，次のように定義できる．

$$Ps_{k,j} = w_{k,j} \Big/ \sum_{j=1}^{n} w_{k,j} \tag{5.8}$$

この方法では，問題に対する適応度の高い対象が高い確率で選択されるようになる．その他の方法として，適応度によって順序付けを行い，その上位の対象だけを選択するようなことも考えられる．このような選択方法はアスピレーション基準に近い操作として，タブーリスト内に含まれている対象であってもコントローラがムーブの要素として選択できるようにする場合に利用できる．

またメモリーモードでは，探索を行う範囲である近傍について考える必要がある．5.1.1でも述べたように，タブーモードとメモリーモードが近傍を用いると，タブーモードで探索を行う領域と変わらなくなるため，メモリーを用いる意味が失われる．そこで，このモードではさらに広い領域を近傍として定義することで，探索の効率化を図る．よって，タブーモードよりも広い近傍を定義すればよいが，ここでは，ｎ階近傍の概念を用いることで対応する．これにより，同様にムーブも決定できる．局所的に集中して調べるにはｎの値を小さくし，大域的に調べるにはｎの値を大きくするなどして近傍のサイズを変えることも考えられる．

ここでさらに検討が必要なことは，その近傍内でどの程度の探索を行うのかという意味での反復回数LtとタブーリストTLの取り扱いである．探索近傍がｎ階近傍であるとすると，近傍サイズである要素数num_nは大きくなる．ここではそれらすべてを探索するようなことはせず，１階近傍内の要素数numよりも少なくし，同じ計算時間で探索ステップ数を増やすように設定するほうが効果的であると考えられる．その探索回数を反復回数Ltとして定義する．これも，シミュレーテッド・アニーリングでの反復回数と同様に考えられる．また，タブーリストの取り扱いであるが，メモリーモードに入るとタブーリストを初期化してしまう場合と，初期化せずに引き続きタブーモードで更新しながら利用する場合がある．これは，メモリーモードで探索を行う

近傍のサイズも考慮して決定する必要がある．大きな近傍を用いた場合には，暫定解の構造が大きく変わってしまう可能性があるので，リストを初期化し，新たにタブーリストを構築したほうが有効であると考えられる．また適応的タブー探索では，基本的にタブーリストはタブーモードでのみ利用するものであり，メモリーモードでは利用しない．

5.2 シミュレーション結果

実際に，これまでに説明した適応的タブー探索を最適配置問題に適用し，その効果を見る．まず，適応的タブー探索において，適応度評価メモリーの構成やオブザーバーやコントローラの働きの違いによって探索にどのような影響があるのかについて調べる．次に，ここで利用している適応度評価メモリーAMを，第４章で説明した適応型シミュレーテッド・アニーリングの中で適応度評価テーブルETの代わりに利用した場合の効果を数値例により比較する．そして最後に，他のメタヒューリスティック手法との比較を行い，総合的な評価を行う．配置する部品として任意多角形状部品を用い，部品数$n=20$の第４章で用いたデータと同じデータ６種類を用いてシミュレーションを行った．また，近傍としては，第３章で最も効果的であった１ブロック交換操作を用いる．

5.2.1 適応的タブー探索の評価

ここでは，適応度評価メモリーを構成するリストとして，以下の６種類のリストにを用意した．これらは，探索を有効に進めるための指標を持たせたリストであり，それぞれの更新条件を示す．

List1: 暫定解よりも有効な解が見つかった場合に，そのムーブの要素に対応するリスト内の要素の適応度を１上げる．

List2: 暫定解と同じ評価の解が見つかった場合に，そのムーブの要素に対応するリスト内の要素の適応度を１上げる．

List3: ある閾値eを越える解が見つかった場合に，そのムーブの要素に対応するリスト内の要素の適応度を１上げる．

List4: 暫定解よりも有効な解が見つかった場合に，そのムーブの要素に対応するリスト内の要素の適応度を１上げ，暫定解と同じ評価の解が見つかった場合に，そのムーブの要素に対応するリスト内の要素の適応度を１下げる．

List5: 暫定解よりも有効な解が見つかった場合に，そのムーブの要素に対応するリスト内の要素の適応度を１上げ，ある閾値eを越える解が見つかった場合に，そのムーブの要素に対応するリスト内の要素の適応度を１下げる．

List6: 暫定解と同じ評価の解が見つかった場合に，そのムーブの要素に対応するリスト内の要素の適応度を１上げ，ある閾値eを越える解が見つかった場合に，そのムーブの要素に対応するリスト内の要素の適応度を１下げる．

この各リストで想定している指標は，List1，List3，List4，List5は大域的な探索が可能となるリストであり，List2，List6は局所的な探索が可能となるリストである．また，ここでは適応度評価メモリーを参照する条件としては，あるステップの間，解の更新が見られない場合に参照するようにする．このとき，このステップ数をsn，探索を行う近傍の階数をrとすると，あるリストlt_iを参照した探索は(lt_i, sn, r)と記述できる．この組み合わせを条件として，各リストを用いてコントローラが確率的に探索解を決定する．

またここでも分かるように，基本となるリストはList1，List2，List3の３種類であり，それらを組み合わせることによって他のリストを構成している．このように，少ない基本リストから多くの変化を持たせることが可能である．

a. １つのリストで構成されるメモリーの場合

まず，用意した６種類のリストをそれぞれ単独で用いた場合の比較を行う．これにより，それぞれのリストの特性が判断できる．ここで行った適応的タブー探索のシミュレーションでは，タブーリストサイズ$sl=5$とし，終了条件としての最大探索回数$tsn=30$とした．さらに，それぞれのリストでテーブルを参照する条件として，$sn=4$，$r=2$ (List_x,4,2)とし，メモリーモードでの解の探索数$Lt=num$とした．

Table 5.1 Comparison of each list

a) Data 1

(List_x,4,2)	List1	List2	List3	List4	List5	List6
Average	18.869	18.632	18.345	18.311	18.323	17.760
Best value	16.456	16.365	15.777	17.182	15.959	13.027
Deviation	1.279	1.256	1.448	0.733	1.507	2.255

b) Data 2

(List_x,4,2)	List1	List2	List3	List4	List5	List6
Average	18.663	18.929	18.841	18.525	18.347	17.789
Best value	16.674	17.250	16.884	16.730	17.490	15.625
Deviation	1.493	1.108	1.576	0.971	0.533	2.014

c) Data 3

(List_x,4,2)	List1	List2	List3	List4	List5	List6
Average	17.721	18.532	17.734	17.543	17.467	17.730
Best value	13.789	17.210	15.177	16.103	15.481	15.900
Deviation	1.915	1.156	1.622	0.898	1.301	1.143

d) Data 4

(List_x,4,2)	List1	List2	List3	List4	List5	List6
Average	15.386	16.113	15.374	15.373	15.167	15.529
Best value	13.156	15.028	13.495	14.678	12.756	14.092
Deviation	1.153	0.993	0.996	0914	1.565	0.688

e) Data 5

(List_x,4,2)	List1	List2	List3	List4	List5	List6
Average	17.931	18.567	18.563	17.560	18.101	17.930
Best value	15.621	16.973	16.753	16.143	16.374	15.988
Deviation	1.100	1.825	1.081	1.152	1.079	1.661

f) Data 6

(List_x,4,2)	List1	List2	List3	List4	List5	List6
Average	17.383	17.285	17.396	16.905	16.448	16.723
Best value	14.741	14.381	15.260	14.936	13.936	13.611
Deviation	1.773	1.643	1.348	1.236	1.481	1.981

つまり，１階近傍内の要素数と同じ数の解を探索するということである．そして，メモリーモードからタブーモードへ戻った場合にはタブーリストTLは初期化するようにしている．また，ここで用いるList3，List5，List6の閾値eは，配置する部品の面積和の約３倍に値を設定し，それを越えるような目的関数値が得られた場合に適用するようにした．それぞれのリストだけをメモリーとして用いてシミュレーションを行った結果をTable 5.1に示す．ここではそれぞれ10回のシミュレーションを行い，結果である余材率の平均値，それらの中での最良値及び標準偏差を示している．

この結果を見ると，List1，List2，List3よりもList4，List5，List6のほうが良い結果が得られる傾向にある．この６種類のリストの中で，List1，List2，List3は基本となるリストであり，その条件を組み合わせることでList4，List5，List6を作っている．このように基本リストを組み合わせることによって，リストの特徴を強調することが可能となり，強調したほうが探索には効果的であることが分かる．

b.　２つのリストで構成されるメモリーの場合

次に，これまで定義したリスト２つを適応度評価メモリーとして用いた場合についての比較を行う．ここでは，List_xとList_yの２つのリストを用意する．List_xは aの場合と同じ条件で参照し，探索近傍の階数$r=2$とする(List_x,4,2)．List_yは，参照する条件として$sn=5$とし，探索近傍の階数$r=3$とする(List_y,5,3)．よってこの場合，List_xを参照して探索を行っても新たな解が見つからなかった場合に，List_yを参照し，さらに広い探索空間での探索を試みるようになる．その他の条件は aの場合

Table 5.2 Search effects of double list memory

a) Data 1

(List_x,4,2) (List_y,5,3)	List4 —	List2 List4	List4 List2	List4 List3	List4 List4	List6 List4
Average	18.311	18.640	17.849	18.146	17.433	16.586
Best value	17.182	17.617	14.726	15.900	13.060	13.909
Deviation	0.733	0.843	1.691	1.513	2.293	1.826

b) Data 2

(List_x,4,2) (List_y,5,3)	List4 —	List2 List4	List4 List2	List4 List3	List4 List4	List6 List4
Average	18.525	18.210	18.395	18.012	17.831	17.423
Best value	16.730	16.681	17.230	14.945	15.429	15.025
Deviation	0.971	0.885	0.940	1.393	1.576	1.135

c) Data 3

(List_x,4,2) (List_y,5,3)	List4 —	List2 List4	List4 List2	List4 List3	List4 List4	List6 List4
Average	17.543	17.541	17.116	17.344	17.470	17.378
Best value	16.103	15.862	15.399	15.269	15.628	14.987
Deviation	0.898	0.999	1.181	1.024	1.123	1.529

d) Data 4

(List_x,4,2) (List_y,5,3)	List4 —	List2 List4	List4 List2	List4 List3	List4 List4	List6 List4
Average	15.373	15.420	14.967	14.748	14.878	15.051
Best value	14.678	13.125	12.502	12.425	13.314	12.510
Deviation	0.914	1.470	1.457	1.145	0.903	1.103

e) Data 5

(List_x,4,2) (List_y,5,3)	List4 —	List2 List4	List4 List2	List4 List3	List4 List4	List6 List4
Average	17.560	18.011	17.011	17.346	17.251	17.699
Best value	16.143	16.957	14.704	15.463	15.559	15.941
Deviation	1.152	0.795	1.270	1.257	1.36	1.186

f) Data 6

(List_x,4,2) (List_y,5,3)	List4 —	List2 List4	List4 List2	List4 List3	List4 List4	List6 List4
Average	16.905	15.695	16.448	16.519	16.151	16.356
Best value	14.936	13.720	15.076	13.893	13.703	13.626
Deviation	1.236	1.501	0.942	1.398	1.959	1.277

と同じである．いくつかのパターンでList1からList6を組み合わせたメモリーを用いたシミュレーションの結果をTable 5.2に示す．ここでは，aで比較的評価の高かった基本リストを組み合わせたリストであるList4を中心にメモリーを構成して比較を行った．シミュレーションの方法や結果の出力については aで行ったものと同様である．

この結果を見ると，２つのリストを組み合わせてメモリーを構成したほうが，１つのリストを用いるだけの場合よりも良い結果が得られる傾向にあることが分かる．特に，List4をList_xとして用い，List2，List3，List4をList_yとして用いた場合を見ると，どの場合もList4だけでメモリーを構成した場合よりも良い結果が得られている．このことからも，List_yがList_xを補う機能を果たしていることが分かる．また，List2及びList6をList_xとして用いた場合も同様のことがTable 5.1と比較することで言える．

c. ３つのリストで構成されるメモリーの場合

最後に，これまで定義したリスト３つを適応度評価メモリーとして用いた場合に

Table 5.3 Search effects of triple list memory

a) Data 1

(List_x,4,2)	List4	List4	List4	List4	List5
(List_y,5,3)	—	List4	List2	List4	List4
(List_z,6,4)	—	—	List3	List4	List3
Average	18.311	17.433	17.410	17.293	17.219
Best value	17.182	13.060	17.830	15.552	14.124
Deviation	0.733	2.293	1.163	0.969	1.781

b) Data 2

(List_x,4,2)	List4	List4	List4	List4	List5
(List_y,5,3)	—	List4	List2	List4	List4
(List_z,6,4)	—	—	List3	List4	List3
Average	18.525	17.831	17.494	17.428	17.674
Best value	16.730	15.429	15.762	14.872	15.450
Deviation	0.971	1.576	1.144	1.498	1.049

c) Data 3

(List_x,4,2)	List4	List4	List4	List4	List5
(List_y,5,3)	—	List4	List2	List4	List4
(List_z,6,4)	—	—	List3	List4	List3
Average	17.543	17.470	17.050	17.012	17.132
Best value	16.103	15.628	15.460	15.447	14.001
Deviation	0.898	1.123	1.058	1.188	1.432

d) Data 4

(List_x,4,2)	List4	List4	List4	List4	List5
(List_y,5,3)	—	List4	List2	List4	List4
(List_z,6,4)	—	—	List3	List4	List3
Average	15.373	14.878	14.973	14.649	14.742
Best value	14.678	13.314	13.158	12.795	13.432
Deviation	0.914	0.903	1.046	0.901	1.166

e) Data 5

(List_x,4,2)	List4	List4	List4	List4	List5
(List_y,5,3)	—	List4	List2	List4	List4
(List_z,6,4)	—	—	List3	List4	List3
Average	17.560	17.251	17.012	16.615	17.239
Best value	16.143	15.559	15.055	15.373	15.980
Deviation	1.152	1.359	1.200	0.921	1.170

f) Data 6

(List_x,4,2)	List4	List4	List4	List4	List5
(List_y,5,3)	—	List4	List2	List4	List4
(List_z,6,4)	—	—	List3	List4	List3
Average	16.905	16.151	15.721	15.464	16.204
Best value	14.936	13.703	13.088	14.197	13.486
Deviation	1.236	1.959	1.480	1.119	1.746

ついての比較を行う．ここでは，List_xとList_yとList_zの３つのリストを用意する．List_x及びList_yについては bの場合と同じ条件で参照し，探索近傍の階数も同じとする(List_x,4,2)(List_y,5,3)．List_zについては，参照する条件として，$sn=6$とし，その探索近傍の階数$r=4$とした(List_z,6,4)．いくつかのパターンでList1からList6を組み合わせたメモリーを用いたシミュレーションの結果を**Table 5.3**に示す．シミュレーションの方法，結果の出力については a，bの場合と同じである．

この結果を見ると，参照するリストの数を増やせば，それにつれて良い結果が得られる傾向にあることが分かる．特にList4に見られるように，同じリストを用いてメモリーを構成しても，探索を行う近傍の階数を変えることで近傍を広くすることによって，探索の効率を上げることが期待できる．

5.2.2 適応度評価メモリーを用いたシミュレーテッド・アニーリング

４章では，シミュレーテッド・アニーリングの枠組みの中に適応度評価テーブルを

用いた適応型シミュレーテッド・アニーリングを提案したが，その適応度評価テーブルに変えて，本章で用いている適応度評価メモリーを用いた場合にどのような効果があるのかについて調べる．

これら２つのアイテムの違いは，適応度評価テーブルでは問題で扱う対象に対して，その問題の中での意味としての指標を１つに限定して捉えようとするのに対して，適応度評価メモリーではその問題の中での複数の指標を同時に捉えようとすることである．適応的タブー探索では，ある指標の下に構成されたリストおのおのが補い合ってうまく探索を進めることが可能であることが分かった．これと同様のことがシミュレーテッド・アニーリングでも可能かどうかを検証する．

適応度評価メモリーでは，メモリー内のリストそれぞれが適応度評価テーブルとして機能する．また，考慮しなければならないこととして，適応的状態へ遷移した時に，適応的タブー探索のように探索近傍としてｎ解近傍を導入するかどうかの問題がある．適応的タブー探索の場合には，メモリーモードに移った場合にその効果を出すためには，近傍を拡張する必要性があった．シミュレーテッド・アニーリングでは，通常状態でも探索解の選択においてランダム性が導入されている．よって，近傍を拡張する必然性はないが，その効果についても比較を行う．

シミュレーションでは，5.2.1で用いたリストと同じList1からList6までのリストをメモリー要素として用いる．それぞれのリストを利用した適応的状態への遷移条件についても5.1.1と同様に定義する．シミュレーテッド・アニーリングの基本パラメータは，$t_0 = 8$，$T(0) = 1000$，$\gamma = 0.85$，$L = 100$ とした．ここでも，10回のシミュレーションを行い，得られた結果である余材率の平均，その中での最良値，また標準偏差を示す（Table 5.4）．

この結果によると，結論として適応度評価メモリーはシミュレーテッド・アニーリングの枠組みの中ではうまく機能していないということが分かる．メモリー内のリストを増やすことの効果が表れていない．この理由は，リストを増やすことによって適応的状態に遷移する回数が多くなり，通常状態での探索が少なくなる．そのことによって，本来シミュレーテッド・アニーリングが持っているアニーリングの特徴が

Table 5.4 Search effects using adaptive memories in ASA

a) Data 1

	(List2,4,1)	(List3,4,1)	(List4,4,1)	(List6,4,1)	(List6,4,1) (List4,5,1)
Average	18.604	18.222	17.843	18.368	18.712
Best value	16.877	15.982	15.177	15.986	15.617
Deviation	0.872	1.382	1.922	0.974	1.856
	(List2,4,1) (List4,5,1)	(List2,4,1) (List4,5,2)	(List4,4,1) (List4,5,2)	(List4,4,1) (List4,5,2) (List4,6,3)	(List2,4,1) (List4,5,1) (List6,6,1)
Average	19.195	17.947	17.823	18.019	18.861
Best value	17.679	15.664	14.258	14.269	15.054
Deviation	0.994	1.381	1.676	1.879	1.754

b) Data 2

	(List2,4,1)	(List3,4,1)	(List4,4,1)	(List6,4,1)	(List6,4,1) (List4,5,1)
Average	18.287	19.628	17.647	19.209	19.339
Best value	15.235	18.283	15.016	14.444	18.153
Deviation	2.075	0.994	1.734	2.218	1.049
	(List2,4,1) (List4,5,1)	(List2,4,1) (List4,5,2)	(List4,4,1) (List4,5,2)	(List4,4,1) (List4,5,2) (List4,6,3)	(List2,4,1) (List4,5,1) (List6,6,1)
Average	18.378	19.220	18.763	17.750	18.581
Best value	15.729	15.925	15.907	16.009	16.455
Deviation	1.339	1.797	1.722	1.363	1.344

c) Data 3

	(List2,4,1)	(List3,4,1)	(List4,4,1)	(List6,4,1)	(List6,4,1) (List4,5,1)
Average	17.599	17.594	18.286	18.406	17.936
Best value	15.600	15.805	16.366	16.262	15.923
Deviation	2.239	1.196	1.657	1.729	1.320
	(List2,4,1) (List4,5,1)	(List2,4,1) (List4,5,2)	(List4,4,1) (List4,5,2)	(List4,4,1) (List4,5,2) (List4,6,3)	(List2,4,1) (List4,5,1) (List6,6,1)
Average	17.245	17.425	18.322	18.008	18.180
Best value	14.987	15.795	15.937	15.652	14.020
Deviation	1.290	1.148	1.827	1.074	2.258

d) Data 4

	(List2,4,1)	(List3,4,1)	(List4,4,1)	(List6,4,1)	(List6,4,1) (List4,5,1)
Average	15.023	15.292	15.100	15.210	15.227
Best value	13.045	13.924	11.176	11.783	12.924
Deviation	1.288	1.085	2.130	1.377	1.194
	(List2,4,1) (List4,5,1)	(List2,4,1) (List4,5,2)	(List4,4,1) (List4,5,2)	(List4,4,1) (List4,5,2) (List4,6,3)	(List2,4,1) (List4,5,1) (List6,6,1)
Average	14.916	15.129	14.954	15.174	14.833
Best value	13.481	14.089	13.914	13.550	10.401
Deviation	1.368	0.732	0.944	1.175	1.818

e) Data 5

	(List2,4,1)	(List3,4,1)	(List4,4,1)	(List6,4,1)	(List6,4,1) (List4,5,1)
Average	18.108	16.615	17.680	17.692	18.109
Best value	15.602	12.530	15.900	14.603	16.234
Deviation	1.436	1.675	1.298	2.107	1.179
	(List2,4,1) (List4,5,1)	(List2,4,1) (List4,5,2)	(List4,4,1) (List4,5,2)	(List4,4,1) (List4,5,2) (List4,6,3)	(List2,4,1) (List4,5,1) (List6,6,1)
Average	17.691	17.790	17.676	17.711	17.112
Best value	16.367	14.165	15.038	14.865	15.949
Deviation	1.520	1.584	1.466	1.503	1.083

f) Data 6

	(List2,4,1)	(List3,4,1)	(List4,4,1)	(List6,4,1)	(List6,4,1) (List4,5,1)
Average	16.097	16.949	16.535	15.938	17.035
Best value	13.198	14.238	14.259	12.498	15.055
Deviation	1.628	1.379	1.667	2.228	1.140
	(List2,4,1) (List4,5,1)	(List2,4,1) (List4,5,2)	(List4,4,1) (List4,5,2)	(List4,4,1) (List4,5,2) (List4,6,3)	(List2,4,1) (List4,5,1) (List6,6,1)
Average	16.987	17.859	16.322	17.597	18.155
Best value	13.833	14.766	13.814	15.770	14.328
Deviation	1.888	2.028	1.646	1.217	1.735

消されていることにあると思われる．

また，適応的状態での探索解近傍について，適応的タブー探索のようにリストが多くなるにつれその探索近傍を広く取る場合と，どのリストの場合も通常状態と同じ探索近傍を用いた場合とで比較を行った．ここで得られた結果では，必ずしも近傍を広く取るほうが良いという結果は得られなかった．通常状態と同じ近傍定義で十分であり，むしろそのほうが良い結果が得られている．

ここでの結果から，タブー探索とシミュレーテッド・アニーリングの枠組みの違いが見える．タブー探索では一般的に様々なメモリーを用いて探索の効果を上げようとする手法であるが，5.2.1の比較でも分かるように適応度評価メモリー内のリストを多くすればそれだけ何らかの効果は得られている．よって，この手法のコンセプトとして言われるように，確かに様々なメモリーを用いることで効率が上がるような枠組みにはなっていると考えられる．よって，ある意味捉えどころのない手法であると言える．また一方で，シミュレーテッド・アニーリングの場合には，無闇にメモリーを用いても期待するような効果は得られず，返って逆効果となりシミュレーテッド・アニーリングの枠組みが持っている良さを消してしまう．このことから，タブー探索のようには様々なオプションを導入することは難しいが，明瞭な手法であると言える．

5.2.3 メタヒューリスティック手法としての評価

適応的タブー探索がメタヒューリスティック手法として有効な手法であるかどうか調べるために，他のメタヒューリスティック手法との比較を行う．これまでと同様にメタヒューリスティック手法として，局所探索(LS)，シミュレーテッド・アニーリング(SA)，遺伝アルゴリズム(GA)及び第３章で示した基本的なタブー探索(TS)を取り上げる．シミュレーションの条件としては，局所探索では最大探索回数$lsn=15$とした．シミュレーテッド・アニーリングは，$t_0=40$，$T(0)=1000$，$\gamma=0.85$，$L=100$とした．タブー探索では，タブーリストの長さ$sl=3$，最大探索回数$tsn=15$とした．その構成は，第４章で用いた構成と同じである．また遺伝アルゴリズムについては，世代数$gn=50$，個体数$gm=40$とし，交叉確率$P_{Crossover}=0.9$，突然変異確率$P_{Mutation}=0.1$

Table 5.5 Comparison of other meta-heuristics

a) Data 1

	LS	SA	GA	TS	ATS1	ATS2	ATS3
Average	18.887	17.670	18.849	19.489	18.311	17.433	17.293
Best value	16.664	16.417	16.688	16.036	17.182	13.060	15.552
Deviation	1.139	0.954	1.383	1.808	0.733	2.293	0.969

b) Data 2

	LS	SA	GA	TS	ATS1	ATS2	ATS3
Average	20.417	19.203	19.047	20.324	18.525	17.831	17.428
Best value	17.469	17.035	17.215	17.163	16.730	15.429	14.872
Deviation	1.720	1.502	1.458	2.039	0.971	1.576	1.498

c) Data 3

	LS	SA	GA	TS	ATS1	ATS2	ATS3
Average	17.825	17.491	18.464	18.845	17.543	17.470	17.012
Best value	14.683	15.526	16.854	16.920	16.103	15.628	15.447
Deviation	1.705	1.544	0.857	1.455	0.898	1.123	1.188

d) Data 4

	LS	SA	GA	TS	ATS1	ATS2	ATS3
Average	16.773	15.443	15.696	15.956	15.373	14.878	14.649
Best value	15.124	14.223	13.486	12.920	14.678	13.314	12.795
Deviation	1.376	0.731	1.391	2.047	0.914	0.903	0.901

e) Data 5

	LS	SA	GA	TS	ATS1	ATS2	ATS3
Average	19.479	17.806	17.328	19.572	17.560	17.251	16.615
Best value	16.221	16.059	14.690	17.449	16.143	15.559	15.373
Deviation	1.838	1.014	1.455	1.650	1.152	1.359	0.921

f) Data 6

	LS	SA	GA	TS	ATS1	ATS2	ATS3
Average	19.096	17.127	16.823	18.430	16.905	16.151	15.464
Best value	17.426	13.717	14.104	15.847	14.936	13.703	14.197
Deviation	1.498	1.553	1.745	2.005	1.236	1.959	1.119

とした．この交叉等の操作についても，第４章で用いた操作と同じである．適応的タブー探索として，5.2.1で行ったシミュレーションの中で，１つのリストで適応度評価メモリーを構成した(list4,4,2)，２つのリストでメモリーを構成した(List4,4,2) (List4,5,3)，及び３つのリストを用いてメモリーを構成した(List4,4,2) (List4,5,3) (List4,6,4)をそれぞれATS1，ATS2，ATS3として比較の対象とした．これまでと同様にそれぞれ10回のシミュレーションを行い，その平均，最良値，標準偏差の結果をTable 5.5に示す．

この結果から，適応的タブー探索ATS2及びATS3が他の手法と比較して最適解探索に有効であることが分かる．これまでにタブー探索が有効であるという報告がなされているが，これは中期記憶や長期記憶などのメモリーを用いた柔軟な探索が可能であることがそのような評価につながっていることはこの結果からも明らかである．これまでに用いられていた様々な複雑なメモリーを用いることで得られていた効果が，ここで提案したような簡単な構造のリストの集まりであるメモリーを用いることで十分に得られることが分かる．

ここで得られた結果では，１つのリストによってメモリーを構成した場合には，適応的タブー探索以外の手法で最も優れているシミュレーテッド・アニーリングと同等かむしろ劣る性能しかでていない．よって，これまでのタブー探索の中で戦略として複数のメモリーを用意しているように，適応的タブー探索においても複数のリストを用意する必要がある．簡単な構造のリストの集まりでメモリーを構成することで，メモリーの拡張も容易になり，探索の性能向上が図り易くなる．

Table 5.6 Search effects using tabu list

	Data1			Data2		
	LS	ALS	ATS2	LS	ALS	ATS2
Average	18.887	18.427	17.433	20.417	19.662	17.831
Best value	16.664	15.799	13.060	17.469	16.675	15.429
Deviation	1.139	1.620	2.293	1.720	2.044	1.576
	Data3			Data4		
	LS	ALS	ATS2	LS	ALS	ATS2
Average	17.825	17.881	17.470	16.773	15.968	14.878
Best value	14.683	15.567	15.628	15.124	13.851	13.314
Deviation	1.705	1.479	1.123	1.376	1.492	0.903
	Data5			Data6		
	LS	ALS	ATS2	LS	ALS	ATS2
Average	19.476	17.586	17.251	19.096	18.445	16.151
Best value	16.221	15.496	15.559	17.426	16.681	13.703
Deviation	1.838	1.682	1.359	1.498	1.404	1.959

5.2.1でも述べたが，タブー探索はこのようなメモリーを用いれば用いるほど何らかの効果が得られるような枠組みを持っている．このような単純な性質は，タブー探索のベースとなっている局所探索の持つ性質である．そこで，タブー探索おいてタブー領域を設定することの効果を検証するために，局所探索に適応度評価メモリーを利用して探索を行った．その結果をTable 5.6に示す．ここでは，２つのリストでメモリーを構成した(List4,3,2) (List4,4,3)をALSとしており，最大探索回数$lsn = 10$とした．これまでと同様にそれぞれ10回のシミュレーションを行い，その平均，最良値，標準偏差の結果を示している．

この結果を見ると，局所探索に適応度評価メモリーを用いることによって効果があることは明らかであるが，適応的タブー探索に比べると劣る．ATS2のシミュレーションでは，タブーリストのサイズ$sl=5$としており，このシミュレーションで用いたすべての要素20個の中で最大で10個の要素の移動が禁じられることになる．よって探索近傍のサイズは，LSやALSで用いられる近傍サイズと比較して，最も小さいときで約４分の１になる．このことから，タブー領域を設定することで近傍サイズを小さくし，同じ計算時間でも最大探索回数を大きく取ったほうが探索の効率は上がるということが言える．

これらのことから，適応的タブー探索が従来のタブー探索や他のメタヒューリスティック手法よりも容易にコーディングが可能であり，また有効な手法であることが分かる．また，タブーリストなどでタブー領域を適度に取ることでも探索の効率は上がることも示した．

第６章

結論

本論文では，最適化問題の中でも解くことが非常に困難である最適配置問題に対して，メタヒューリスティック手法を用いた解法に関する研究を行った．その過程の中で，矩形形状及び任意多角形状部品配置のための新たなアルゴリズムや扱う問題に対して適応性を持たせた柔軟なメタヒューリスティック手法を提案した．

第２章では，部品配置を行う位置選択に用いる２つのアルゴリズムについて説明した．１つは矩形部品を配置するための稜線法であり，もう１つは任意多角形状部品の配置も可能となるように稜線法を拡張したアルゴリズムである．そして，これらのアルゴリズムを用いて，同じ形状ばかりの部品を配置したシミュレーションや違う形状の部品を配置したシミュレーションの結果や，配置する部品数による配置効率への影響について示した．特に，従来の方法との比較や実際に板金加工工程へ適用した例などを示して，このアルゴリズムが有効であることを確認した．

第３章以降では，第２章で説明した配置アルゴリズムを利用して，メタヒューリスティック手法を用いた配置（配置順序）の最適化について検討を行った．

まず第３章では，数多く提案されているメタヒューリスティック手法の中でも，最も基本的な枠組みであると思われる局所探索，シミュレーテッド・アニーリング，タブー探索及び遺伝アルゴリズムの間で，最適配置問題に対しての性能比較を行った．最初にこれらを２つのグループに分けて，それぞれの特徴について検討した．１つは近傍探索をベースとした局所探索，シミュレーテッド・アニーリング，タブー探索のグループ，そしてもう１つは集団探索である遺伝アルゴリズムである．近傍探索のグループでは，その近傍の定義について検証を行った．その中で，できるだけ解の構造を変えない近傍で，かつそのサイズが小さい近傍が好ましいことが分かった．また遺

伝アルゴリズムでは，集団を進化させるための様々なオプションについて検証を行った．ここでは，配置順序の最適化を行うために解のコーディングを順序表現で行っている．遺伝アルゴリズムの特徴は親の性質を受け継ぐという交叉にあると考えられるが，順序表現ではこの交叉のオペレータによって解構造が大きく変わってしまい，大域的な探索しか実現できず，期待した効果を得ることが困難であった．これは，最適配置問題の性質にも関係していることが考えられる．また，突然変異のオペレータに近傍操作に近いオペレータを用い，かつ突然変異確率を高くすることで集団の多様度を維持することは可能であり，探索効率も上げることができることも分かった．これらのことから，この最適配置問題に対して遺伝アルゴリズムを用いた最適化はかなり困難であることが分かった．次に，ここで取り上げたメタヒューリスティック手法とそれ以外の手法としてランダム探及び簡単なヒューリスティック手法との間での比較を行った．比較したメタヒューリスティック手法は基本的なオペレーションを用いるだけとし，そのシミュレーションでは，稜線法を用いて矩形部品の配置を行った．その結果として，これらの中で最も優れていたのは，シミュレーテッド・アニーリングであった．タブー探索については，これまでに報告されているような良い結果は得られなかったが，その点を除けばスケジューリング問題などこれまでに他の問題で示されている結果と一致していた．

また，第２章で提案した稜線法などの配置位置選択のアルゴリズムが変わった場合に，メタヒューリスティック手法を用いた配置順序による最適化にどのような影響があるのかについても調べた．その結果，稜線法などの部分アルゴリズムが変わると，そのアルゴリズムの性能が線形にメタヒューリスティック手法を用いた最適化に反映され，またメタヒューリスティック手法が変わっても，その性能がそのまま最適化の性能に反映された．よって，これらはまったく独立に考えることができることを示した．

第４章では，第３章で最も結果の良かったシミュレーテッド・アニーリングの基本的な枠組みを用いて，問題に対する適応性を持たせた適応型シミュレーテッド・アニーリングを提案した．この手法は，知識ベースなどを構築する必要はなく，探索による履歴を適応度評価テーブルに保持し，それを活用して探索の効率化を図ろうとするものである．第３章で用いた各種メタヒューリスティック手法と比較した結果，適応型シミュレーテッド・アニーリングが最も優れており，配置するデータがどのようなデータであっても常に一定レベル以上の結果を得ることが可能であることが分かった．また従来のシミュレーテッド・アニーリングで問題であった，温度パラメータが小さくなると定常状態に陥りやすいという問題についても，適応的状態に遷移することで抜け出すことが可能である．

第５章では，第３章で探索効率の悪かったタブー探索を取り上げ，その特徴であるメモリーを用いた改善を行った．ここでは，適応度評価メモリーを用いた適応的タブー探索について提案した．ここで提案したメモリーは構造が非常に単純であり，従来のタブー探索で用いられるような複雑なメモリー構造は取らない．このメモリーは複数のリストから構成され，それぞれのリストがある指標を持って情報を保持するようになる．またこの手法では，第３章であまり効果のなかったアスピレーション基準は戦略としては用いない．むしろ，タブー領域を故意に大きく取ることで最大探索回数を大きくする戦略を取る．最適配置問題に対するシミュレーションによると，この適応的タブー探索はメモリーを構成するリストを多くすればそれだけ探索効率の向上が期待できるという結果が得られた．また，局所探索やシミュレーテッド・アニーリングにこの適応度評価メモリーを用いた場合についても検証した．シミュレーテッド・アニーリングの場合には，メモリーに用いるリストの数を多くしてもタブー探索の場合のような効果は得られなかった．このことは，メモリーを多用することによってシミュレーテッド・アニーリングが持っている良さであるアニーリングの効果が消されてしまうことに原因があると思われる．また，タブー探索のベースである局所探索にメモリーを用いれば，タブー探索と同様に効果はあるが，適応的タブー探索のような効果は得られず，タブー領域を大きく取り，最大探索回数が多く取れる適応的タブー探索のほうが良い結果が得られた．その意味で，ある程度タブー領域を設定して近傍サイズを小さくすることで，最大探索回数を多くすることによる効果があるということが分かった．

Table 6.1 Comparison of ASA and ATS

	Data1			Data2		
	SABM	ATS2	ATS3	SABM	ATS2	ATS3
Average	17.150	17.433	17.293	18.198	17.831	17.428
Best results	14.533	13.060	15.552	15.733	15.429	14.872
Deviation	1.142	2.293	0.969	0.934	1.576	1.498
	Data3			Data4		
	SABM	ATS2	ATS3	SABM	ATS2	ATS3
Average	16.743	17.410	17.012	14.956	14.878	14.649
Best results	15.413	15.628	15.447	14.144	13.314	12.795
Deviation	0.830	1.123	1.188	0.677	0.903	0.901
	Data5			Data6		
	SABM	ATS2	ATS3	SABM	ATS2	ATS3
Average	17.357	17.251	16.615	15.446	16.151	15.464
Best results	15.183	15.559	15.373	13.434	13.703	14.197
Deviation	1.193	1.359	0.921	1.635	1.959	1.119

最後に，第４章で提案した適応型シミュレーテッド・アニーリング(ASA)と第５章で提案した適応的タブー探索(ATS)の比較をTable 6.1に示す．示しているデータは，Table 4.2及びTable 5.5で他のメタヒューリスティック手法と比較したときのデータである．これを見た限りでは，優劣は付けがたい．シミュレーテッド・アニーリングはその枠組みそのものが完成されたものであり，そこにあまり大きく手を加えると，5.2.2で示したように本来の良さを失わせる結果となる．そのため，適応型シミュレーテッド・アニーリングを用いて，Table 6.1で示した探索よりも効率を上げることは困

難であることが予想される．それと比べて，タブー探索の場合，基本的には局所探索であり，そこにいかようにも手を加えることが可能である．その意味では，適応的タブー探索を用いて，さらにパフォーマンスを上げることがある程度期待できる．しかし，これら２つの手法はどちらも問題で扱う対象の適応度を評価し，その評価を探索の中でうまく利用する手法である．よって，コーディングが非常に簡単で高い探索性能を持つ適応型シミュレーテッド・アニーリング，メモリー構造を単純にすることで拡張性の高い適応的タブー探索と言うことができる．これらように，近傍探索をベースとし適応度を評価したメモリーを用いた最適化手法を総称して，ここでは適応型メタヒューリスティックス手法と呼ぶ．

これらのことから，本論文で扱った最適配置問題のような複雑な組合せ最適化問題では，メタヒューリスティック手法が効果的であり，その中でも本論文で提案した適応型メタヒューリスティック手法は様々な問題に対して柔軟に適応した最適化を実現可能な有効な手法である．

今後の課題としては，本論文で取ったアプローチ方法と同様に大きく分けて２つある．１つは最適配置問題の配置位置選択の問題であるが，部品の配置位置に制約があるような場合にどのような解決方法があるのか，また配置対象が２次元部品でなく箱のような３次元部品となった場合にどうするかといったことが上げられる．またもう１つは，メタヒューリスティック手法の改良であるが，これまでにも報告されている処理の並列化を適応型メタヒューリスティック手法の中で実現し，処理速度の高速化を図るとともに，それぞれの処理が自律的な動作が可能かどうかを検証することが上げられる．また，本論文では十分な検討を行わなかった集団探索である遺伝アルゴリズムについても，詳細な検討を行う必要がある．さらに，本論文では最適配置問題について考察を行ったが，ここで提案した適応型メタヒューリスティック手法が他の最適化問題に対しても同様に有効であるのかについても検証する必要がある．

参考文献

(1) K.A.Dowsland and W.B.Dowsland: Packing Problems, *European Journal of Operational Research*, 56 (1992), 2-14.

(2) C.H.Papadimitriou and K.Steiglitz: *Combinatorial Optimization: Algorithms and Complexity*, Prentice Hall, (1982) (also Dover (1998)).

(3) M.R.Garey and D.S.Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, W.H.Freeman, San Francisco, (1979).

(4) R.J.Fowler, M.S.Paterson and S.L.Tanimoto: Optimal Packing and Covering in the Plane are NP-Complete, *Information Processing Letters*, 12, 3 (1981), 133-137.

(5) P.C.Gilmore and R.E.Gomory: Multistage Cutting Stock Problems of Two or More Dimensions, *Operations Research*, 13,1 (1965), 94-120.

(6) N.Christofides and C.Whitlock: An Algorithm for Two-Dimensional Cutting Problems, *Operations Research*, 25, 1 (1977), 30-44.

(7) P.Y.Wang: Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems, *Operations Research*, 31,3 (1983), 573-586.

(8) A.A.Farley: Selection of Stockplate Characteristics and Cutting Style for Two Dimensional Cutting Stock Situations, *European Journal Operational Research*, 44 (1990), 239-246.

(9) J.F.Oliveira and J.S.Ferreira: An Improved Version of Wang's Algorithm for Two-Dimensional Cutting Problem, *European Journal Operational Research*, 44 (1994), 256-266.

(10) M.J.Haims and H.Freeman: A Multistage Solution of the Template-Layout Problem, *IEEE Transactions on Systems Science and Cybernetics*, 6, 2 (1970), 145-151.

(11) H.Freeman: On the Packing of Arbitrary-Shaped Templates, *Second USA-*

JAPAN Computer Conference, Tokyo, Japan, August (1975), 102-107.

(12) D.Dori and M.B.Bassaat: Efficient Nesting of Congruent Convex Figures, *Communications of the ACM*, 27,3 (1984), 228-235.

(13) B.T.Cheok and A.Y.C.Nee: Algorithms for Nesting of Ship/Offshore Structural Plates, *Advances in Design Automation*, 32,2 (1991), 221-226.

(14) A.Albano and G.Sapuppo: Optimal Allocation of Two-Dimensional Irregular Shapes Using Heuristic Search Methods, *IEEE Transactions on Systems, man, and Cybernetics*, 10, 5 (1980), 242-248.

(15) S.S.Israni and J.L.Sanders: Performance Testing of Rectangular Parts-Nesting Heuristics, *International Journal of Production Research*, 23,3 (1985) 437-456.

(16) Y.Cai, L.Liu, W.Wang and J.Sun: An Expert System for Automatic Allocation of 2D Irregular Shapes, Expert System in Computer-Aided Design, *Proceedings of the IFIP WG 5.2 Working Congress*, Sydney, Australia, February (1987), 407-422.

(17) C.R.Reeves (Eds.): *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific (1993).（横山隆一他訳: モダンヒューリスティックス－組合せ最適化の先端手法, 日刊工業新聞社 (1997)）

(18) 茨木俊秀: 組合せ最適化とスケジューリング問題 : 新解法とその動向, 計測と制御, 34, 5 (1995), 340-346.

(19) E.Aarts and J.K.Lenstra (Eds.): *Local Search in Combinatorial Optimization*, John Wiley & Sons (1997).

(20) D.Smith: Bin Packing with Adaptive Search, *Proceedings of the First International Conference on Genetic Algorithms and Their Applications*, Pittsburgh, July (1985), 202-207.

(21) B.Krö ger, P.Schwenderling and O.Vornberger: Parallel Genetic Packing of Rectangles, *Proceedings of Parallel Solving from Nature 1st Workshop* (1991), 160-164.

(22) 藤田喜久雄, 赤木新介, 廣川敬康: 遺伝的アルゴリズムと極小値探索アルゴリズムとのハイブリッド化による板取り問題の一解法, 日本機械学会論文集 (C編), 59,564 (1993), 312-319.

(23) E.Horowitz and S.Sahni: *Fundamentals of Computer Algorithms*, Computer Science Press (1978).

(24) S.Takahara, S.Miyamoto: A study of nesting problem by genetic algorithm. *Proceedings of the 3rd International Conference on Fuzzy Logic, Neural Nets, and Soft Computing*, Iizuka, Japan, August (1994), 459-460.

(25) 高原茂幸, 宮本定明: 適応型シミュレーテッド・アニーリングによる任意形状部品の最適配置, 計測自動制御学会論文集, 34, 11 (1998), 1724-1730.

(26) S.Kirkpatrick, C.D.Gelatt and M.P.Vecchi: Optimization by Simulated Annealing, *Science*, 220 (1983), 671-680.

(27) E.Aarts and J.Korst: *Simulated Annealing and Boltzmann Machines*, John Wiley & Sons (1989).

(28) F.Glover: Tabu Search Part I, *OPSA Journal on Computing*, 1 (1989), 190-206.

(29) F.Glover: Tabu Search Part II, *ORSA Journal on Computing*, 2 (1990), 4-32.

(30) J.H.Holland: *Adaptation in Natural and Artificial Systems*, University of Michigan Press (1975) (also MIT Press (1992)).

(31) L.Davis (Eds.): *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York (1991)

(32) B.L.Fox: Integrating and Accelerating Tabu Search, Simulated Annealing, and Genetic Algorithms, *Annals of Operations Research*, 41 (1993), 47-67.

(33) 北野宏明 (編): 遺伝アルゴリズム, 産業図書 (1993), 61-88.

(34) 田村坦之, 平原明, 鳩野逸生, 馬野元秀: 組合せ最適化問題の近似解法－遺

伝アルゴリズムとラグランジュ緩和法を併用したハイブリッド法ー，計測自動制御学会論文集, 30,3 (1994), 329-336.

(35) P.Jain, P.Fenyes and R.Richter: Optimal Blank Nesting Using simulated Annealing, *Advances in Design Automation*, ASME 1990, 109-116.

(36) K.A.Dowsland: Some Experiments with Simulated Annealing Techniques for Packing Problems, *European Journal of Operational Research*, 68 (1993), 389-399.

(37) M.J.Brusco, G.M.Thompson and L.W.Jacobs: A Morph-Based Simulated Annealing Heuristics for a Modified Bin-Packing Problem, *Journal of the Operational Research Society*, 48 (1997), 433-439.

(38) 松田浩一，高井幸秀，吉田耕作，西村勝，松山峯雄，澤田泰明: シミュレーテッド・アニーリング法を用いた板取計画の最適化，計測自動制御学会論文集, 31, 5 (1995), 544-552.

(39) K.Shahookar and P.Mazumder: A Genetic Approach to Standard Cell Placement Using Meta-Genetic Parameter Optimization, *IEEE Transactions on Computer-Aided Design*, 9,5 (1990), 500-511.

(40) 小圷成一，須貝康雄，平田廣則: 遺伝的要素を取り入れた改良型アニーリング法によるブロック配置手法，電子情報通信学会論文誌，J73-A, 1 (1990), 87-94.

(41) 白井裕，松本直文: レイアウト設計のためのメタ戦略によるハイブリッド技法，日本機械学会論文集(C編), 63,611 (1997), 2337-2344.

(42) I.H.Osman: Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem, *Annals of Operations Research*, 41 (1993), 421-451.

(43) N.Metropolis, A.W.Rosenbluth, M.N.Rosenbluth, A.H.Teller and E.Teller: Equation of State Calculations by Fast Computing Machines, *Journal of Chemical Physics*, 21, 6 (1953), 1087-1091.

(44) B.Hajek: Cooling Schedules for Optimal Annealing, *Mathematics of Operations Research*, 13,2 (1988), 311-329.

(45) F.Glover and H.J.Greenberg: New Approaches for Heuristic Search: a Bilateral Link with Artificial Intelligence, *European Journal of Operational Research*, 39 (1989), 119-130.

(46) D.de Werra and A.Hertz: Tabu Search Techniques: A Tutorial and an Application to Neural Networks, *OR Spectrum*, 11 (1989), 131-141.

(47) F.Glover, E.Taillard and D.de Werra: A User's Guide to Tabu Search, *Annals of Operations Research*, 41 (1993), 3-28.

(48) D.E.Goldberg: *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Massachusetts (1989).

(49) 三宮信夫: 遺伝アルゴリズムによる最適化問題の解法，第36回システム制御情報学会研究発表講演会論文集 (1992), 9-18.

(50) 高原茂幸，宮本定明: メタ戦略を用いたネスティング問題の解法，計測自動制御学会論文集, 33,11 (1997), 1081-1086.

(51) S.Takahara, S.Miyamoto: Solution for a Class of Nesting Problems Using Genetic Algorithms, Simulated Annealing and Tabu Search, *Proc. of the 7th IFSA World Congress*, Prague, June (1997), 463-468.

(52) M.Malek, M.Guruswamy and M.Pandya: Serial and Parallel Simulated Annealing and Tabu Search Algorithms for the Traveling Salesman Problem, *Annals of Operations Research*, 21 (1989) 59-84.

(53) J.Skorin-Kapov: Tabu Search Applied to the Quadratic Assignment Problem, *ORSA Journal on Computing*, 2,1 (1990), 33-45.

(54) E.H.Aarts, P.J.M.van Laarhoven, J.K.Lenstra and N.L.J.Ulder: A Computational Study of Local Search Algorithms for Job Shop Scheduling, *ORSA Journal of Computing*, 6, 2 (1994), 118-125.

(55) D.S.Johnson, C.R.Aragon, L.A.McGeoch and C.Schevon: Optimization by

Simulated Annealing: An Experimental Evaluation; Part I, Graph Partitioning, *Operations Research*, 37 (1989), 865-892.

(56) D.S.Johnson, C.R.Aragon, L.A.McGeoch and C.Schevon: Optimization by Simulated Annealing: An Experimental Evaluation; Part II, Graph Coloring and Number Partitioning, *Operations Research*, 39 (1991), 378-406.

(57) S.Takahara, S.Miyamoto: A Method of Adaptive Simulated Annealing and Application to a Class of Optimal Allocation Problems, *Proc. of the 5th International Conference on Soft Computing and Information/Intelligent Systems*, Iizuka, Japan, October (1998), 227-230.

(58) C.A.Tovey: Simulated Simulated Annealing, *American Journal of Mathematical and Management Sciences*, 8 (1988), 389-407.

(59) E.Taillard: Parallel Taboo Search Techniques for the Job Shop Scheduling Problem, *ORSA Journal on Computing*, 6,2 (1994), 108-117.

(60) E.Taillard: Robust Taboo Search for the Quadratic Assignment Problem, *Parallel Computing*, 17 (1991), 443-455.

(61) S.Takahara, S.Miyamoto: An Adaptive Tabu Search (ATS) and Other Metaheuristics for a Class of Optimal Allocation Problems, *Journal of Advanced Computational Intelligence*, 3, 1 (1999), 21-27.

本研究における公表論文

(1) S.Takahara, S.Miyamoto: A study of nesting problem by genetic algorithm. Proceedings of the 3rd International Conference on Fuzzy Logic, Neural Nets, and Soft Computing, Iizuka, Japan, August (1994), 459-460.

(2) S.Takahara, S.Miyamoto: Solution for a Class of Nesting Problems Using Genetic Algorithms, Simulated Annealing and Tabu Search, Proc. of the 7th IFSA World Congress, Prague, June (1997), 463-468.

(3) 高原茂幸，宮本定明: メタ戦略を用いたネスティング問題の解法，計測自動制御学会論文集，33, 11 (1997), 1081-1086

(4) S.Takahara, S.Miyamoto: A Method of Adaptive Simulated Annealing and Application to a Class of Optimal Allocation Problems, Proc. of the 5th International Conference on Soft Computing and Information/Intelligent Systems, Iizuka, Japan, October (1998), 227-230.

(5) 高原茂幸，宮本定明: 適応型シミュレーテッド・アニーリングによる任意形状部品の最適配置，計測自動制御学会論文集，34, 11 (1998), 1724-1730.

(6) S.Takahara, S.Miyamoto: An Adaptive Tabu Search (ATS) and Other Metaheuristics for a Class of Optimal Allocation Problems, Journal of Advanced Computational Intelligence, 3, 1 (1999), 21-27.

謝辞

本論文にまとめた「メタヒューリスティック手法を用いた最適配置問題」に関する一連の研究は，筑波大学　機能工学系　宮本定明教授の多大なる御指導により成し得たものであり，同教授に深甚なる感謝の意を表します．

また，研究を進めるにあたり，御指導と御教示をいただきました京都大学　工学部片山徹教授に心より感謝致します．また，本論文の作成にあたり，ご多忙のところ有益な御指導と御助言を賜りました筑波大学　電子・情報工学系　西原清一教授，同機能工学系　大田友一教授，同　電子・情報工学系　稲垣敏之教授，同　機能工学系鬼沢武久教授に深く感謝致します．そして，計算シミュレーションにおけるシステム利用にご助力いただいた筑波大学　機能工学系　馬屋原一孝助手及びシステムモデリンググループの学生の皆様に深く感謝致します．

本研究のきっかけは，一光電機株式会社との共同研究として「板金用ＣＡＤ／ＣＡＭシステムの開発」を行った中で，板金部品の展開図の自動配置（ネスティング）を行うという課題に出会ったことからでした．その中で一光電機株式会社では，研究を進める上で様々なご支援，ご協力をいただきました．ここに一光電機株式会社　池田晃代表取締役社長，木澤正広設計主任並びに社員の方々に深く感謝致します．

最後に，本研究を進めるにあたり，様々な面でご協力，ご支援くださいました香川県工業技術センター　宮本康弘所長，神高幸則研究主幹はじめ，職員の皆様に深く感謝致します．