平成 26 年度

修士学位論文

仮想サーバ配置問題のための最適解近似アルゴリズム

Optimal solution approxmation algorithm for virtual server placement problem

1175085　播田 直紀

指導教員　福本 昌弘

2015 年 2 月 27 日

高知工科大学大学院 工学研究科 基盤工学専攻

情報システム工学コース

要 旨

仮想サーバ配置問題のための
最適解近似アルゴリズム

播田 直紀

仮想サーバ配置問題とは，統合先サーバに収容される仮想サーバの各計算資源使用量の合計値が閾値を超えないという制約の下で，必要とする統合先サーバ台数が最小となるサーバの組合せを求める問題である．配置された仮想サーバの資源の必要量が統合先サーバの備える資源容量を超えると，重大な性能劣化を引き起こすことになる．

現実の問題では，計算資源の数は 4〜6，統合対象の仮想サーバ数は数百から千台前後となり，エンジニアによる現場での提案書の作成や複数回実行することを考えると計算時間は数分以内である必要がある．

仮想サーバ配置問題を定式化した N 次元パッキング問題は NP 困難な問題に分類されており，アイテム数の多い問題となると現実的な時間内に厳密解を得られない．そのため，数分以内という短時間で解を必要とする場合，短時間で解を算出できるヒューリスティックなアルゴリズムにより近似解を求める必要がある．

本研究では，より適したアルゴリズムを検討するため，仮想サーバ配置問題にヒューリスティックなアルゴリズムを適用し，改良したランダムサンプリング，遺伝的アルゴリズム，ランダム探索，改良 FFD の 4 つのアルゴリズムで比較・検証を行った．

その結果，改良型ランダムサンプリングを適用することで出力される解に大きく差はないが他のアルゴリズムと比べて良い平均値を示す可能性があることが分かった．

キーワード　仮想サーバ配置問題，N 次元パッキング問題，ヒューリスティックなアルゴリズム，ランダムサンプリング手法

Abstract

Optimal solution approxmation algorithm for virtual server placement problem

Naoki HARITA

An objective of virtual server placement problem is to minimize the number of integrated destination server. The number of integrated destination server is determined by combinations of virtual server placement. Virtual servers must be located so as not to exceed threshold value of integrating destination server. If total value of computational resources exceeds resource capacity provided in integrated destination server, integrated destination server cause significant performance degradation.

As a practical matter, computation time must be within a few minutes, because engineers may create a proposal in outside company and run program more than once. Virtual server placement problem is formulated as N-dimensional packing problem. When there are many items of N-dimensional packing problem, no one can find an exact solution within practical time, because N-dimensional packing problem is NP-hard problem. Therefore, when NP-hard problem's solution is needed in a short time, it is necessary to find an approximate solution by heuristic algorithms.

In this research, I have been applied heuristic algorithms to virtual server placement problem in order to consider appropriate algorithm. Applied four algorithms have been improved random sampling, genetic algorithm, random search, improved FFD. I have been comparing and verifying them.

As a result, there are almost no difference in solutions which four algorithms output. But, I found that improved random sampling is a possibility of indicating average value better than other algorithms.

key words Virtual server placement problem, N-dimensional packing problem, Heuristic algorithm, Random sampling

目次

図目次

図目次

表目次

第 1 章

序論

サーバ統合は企業などで増えたコンピュータを仮想化し，仮想化されたサーバ群を高性能な大容量サーバに集約することで一元管理を行えるようにするシステムシステムソリューションである．クラウドコンピューティングにより世界各地のサーバ群からネットワークを介してリソースの利用が可能になった現在でも，自社の業務に合った高い機能や安全性が求められている場合に，企業が用意したサーバの中だけでクラウドサービスを行うシステムが必要とされている．そのようなクラウドサービスの形態はプライベートクラウドと呼ばれ，その実現のために企業によるサーバ統合が行われている．

サーバ統合を行う際には，仮想化されたサーバをどのように配置するかが最も重要であり，仮想サーバ配置問題と呼ばれる問題が存在する．仮想サーバ配置問題は，統合先サーバに収容されるサーバの各計算資源使用量の合計値が閾値を超えないという制約の下で，目的関数である統合先サーバ台数が最小になるようなサーバの組合せを求める問題である．統合対象のサーバによる資源の必要量が統合先サーバの備える資源容量を超えると，重大な性能劣化を引き起こしてしまう．

仮想サーバ配置問題は N 次元パッキング問題として定式化することができる [2]．N 次元パッキング問題は NP 困難な問題に分類されており，問題として近い 2 次元ビンパッキング問題ではアイテム数が 100 以下の場合は数十分で厳密解法を見つける手法が提案されている [1]．しかし，それ以上のアイテム数の問題となると現実的な時間内では厳密解を得られない．

現実の問題ではサーバ統合を行う場合，計算資源の数は 4〜6，統合対象となる仮想サーバ数は数百から千台前後となる．エンジニアによる現場での提案書の作成や複数回実行する

ことを考えると計算時間は数分以内である必要がある．そのため，数分以内という短い時間内に解を必要とする場合，短時間で解を算出できるヒューリスティックなアルゴリズムにより近似解を求める必要がある．

本研究では，より適したアルゴリズムを検討するため，仮想サーバ配置問題にヒューリスティックなアルゴリズムを適用し，改良したランダムサンプリング手法，及び他の手法との比較・検証結果を報告する．

第 2 章

仮想サーバ配置問題

サーバ統合を行う際には，仮想化された統合対象サーバをどのように配置するかが最も重要であり，仮想サーバ配置問題と呼ばれる問題が存在する．仮想サーバ配置問題では，まず，統合対象となるサーバそれぞれに必要な各計算資源使用量が見積もられる．仮想サーバ毎に見積もられる計算資源は CPU やメモリ，ディスク，ネットワークなどである．そのとき，統合対象サーバ統合先サーバに収容されるサーバの各計算資源使用量の合計値が閾値を超えないという制約の下で，目的関数である統合先サーバ台数が最小になるようなサーバの組合せを求める問題である．

2.1 仮想サーバ配置問題の例

計算資源の数を 3，統合対象サーバ数を 5 とし各統合対象サーバには計算資源の使用量が見積もられているとする．そのとき，統合先サーバの各計算資源使用量の閾値を 80%とした場合の仮想サーバ配置問題の例を図 2.1 に示す．

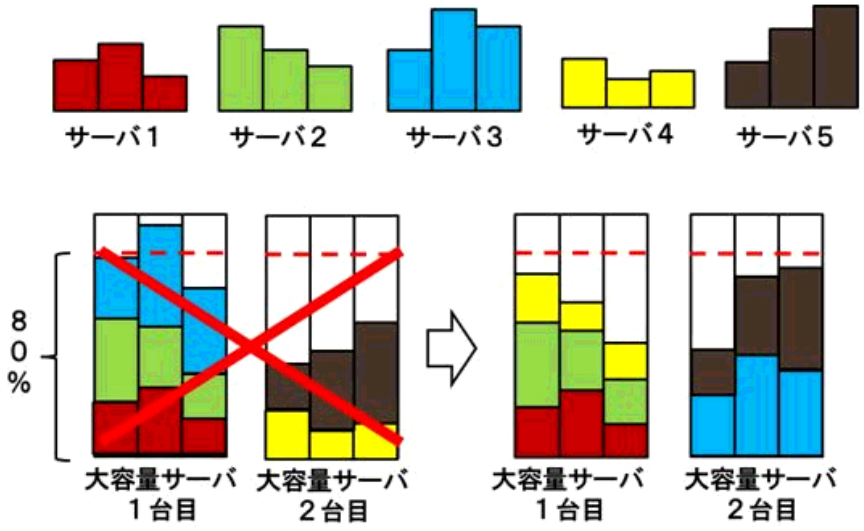

図 2.1　仮想サーバ配置問題の例

このとき，閾値を越えないように配置を行う場合，1 台目に 1，2，4 の仮想サーバを配置，2 台目に 3，5 の仮想サーバを配置することで必要な統合先サーバ数は 2 となる．このように統合対象のサーバ数が少ない問題では必要な統合先サーバ数が最小となる厳密解は見るだけですぐ分かるが統合対象のサーバ数が数百から数千となった場合，必要な統合先サーバ数が最小となる厳密解を人が短時間で求めることは難しい．そのため，定式化を行うことでコンピュータによる計算を可能にし，それにより問題を解く必要がある．

2.2　サーバ統合における 5 次元パッキング問題の定式化

サーバ統合における統合対象サーバをアイテム，統合先サーバをビンと考えた場合，与えられたアイテムをビンに詰める際に使用するビンの数や詰めたアイテムの高さを最小化する組合せ最適化問題はビンパッキング問題と呼ばれ，計算資源が複数ある場合，計算資源の種類数が N となり，N 次元パッキング問題として定式化されている [2].

本研究では，仮想サーバの計算資源を CPU 性能，メモリ使用量，ディスク性能，ディスク容量，ネットワーク性能の 5 つとし，5 次元パッキング問題として定式化される．これらの計算資源の値はすべて統合先サーバの最大使用量を 100%とした場合の割合で与えられる．

統合対象のサーバ数を n，統合対象のサーバを
$s_i(1 \leqq i \leqq n)$ とする．s_i には計算資源としてそれぞれ CPU 性能使用量 pc_i，メモリ使用量 pm_i，ディスク性能使用量 $pd1_i$，ディスク容量使用量 $pd2_i$，ネットワーク性能使用量 pn_i が与えられる．また，m 台の統合先サーバ $s'_j(1 \leqq j \leqq m)$ に統合される統合対象サーバの集合を X_j とする．統合先サーバはすべて同じ容量を持つものであることを前提とし，与えられる閾値はそれぞれ CPU 性能使用量 Rc，メモリ使用量 Rm，ディスク性能使用量 $Rd1$，ディスク容量使用量 $Rd2$，ネットワーク性能使用量 Rn とする．

このとき，サーバ統合の最適化問題は以下のように定式化される．

目的関数:m →最小化

制約条件:$\sum_{s_i \in X_j} pc_i \leqq Rc$,

$\sum_{s_i \in X_j} pm_i \leqq Rm$,

$\sum_{s_i \in X_j} pd1_i \leqq Rd1$,

$\sum_{s_i \in X_j} pd2_i \leqq Rd2$,

$\sum_{s_i \in X_j} pn_i \leqq Rn (i = 1, ..., n, j = 1, ..., m)$

2.3　N 次元パッキング問題の既存研究

N 次元パッキング問題は NP 困難な問題であるため，厳密解法は次元数やアイテム数の少ない問題を対象にしたものしか研究が行われていない．そのため．実用上の次元数やアイテム数の問題を解く手法としてヒューリスティックなアルゴリズムにより近似解を求める手法の研究が多く行われている．

2.4　次元数の少ないパッキング問題でよく用いられるアルゴリズム

次元数の少ないパッキング問題でよく用いられるアルゴリズムとして FFD，BFD，WFD，NFD がある．これらの 4 つのアルゴリズムはどれも順列で表現された解候補を評価するア

ルゴリズムであり，アイテムの添字番号順に詰め込むアイテムが決まるため，アイテムのソートによって得られる値が大きく変わる．そのため，これらのアルゴリズムを基にした研究では，主にアイテムを詰める順番をソートする手法について研究されている．1 次元のパッキング問題において FFD が最も良い結果を出すとされており，ソート方法として最も基本的な方法は優先順位を設定し，降順にソートするものである．多次元のパッキング問題においては各計算資源の使用量の積で降順にソートする手法 [3] が提案されている．解候補の順列表現方法については 4 章で説明する．

2.4.1　FFD(First-Fit Decreasing)

まず箱が 1 つしかない状態からはじめてアイテムが箱に詰められるかどうかを箱の添字番号順に調べ，詰めるのが可能な箱に詰める動作を繰り返す．詰められなかった場合は箱を 1 つ追加してそれにアイテムを詰める．この過程のフローチャートとその例を図 2.2，図 2.3 に示す．

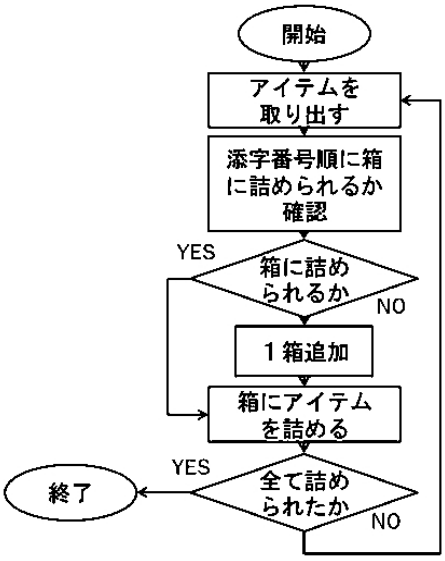

図 2.2　FFD のフローチャート

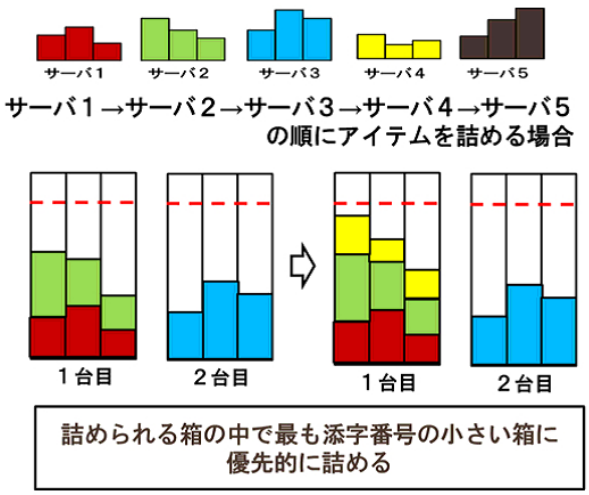

図 2.3　FFD の詰め方の例

2.4.2　BFD(Best-Fit Decreasing)

まず箱が 1 つしかない状態からはじめてアイテムが箱に詰められるかどうかを箱の添字番号順に調べ，最も空き容量の小さい箱に詰める動作を繰り返す．詰められなかった場合は箱を 1 つ追加してそれにアイテムを詰める．この過程のフローチャートとその例を図 2.4，図 2.5 に示す．

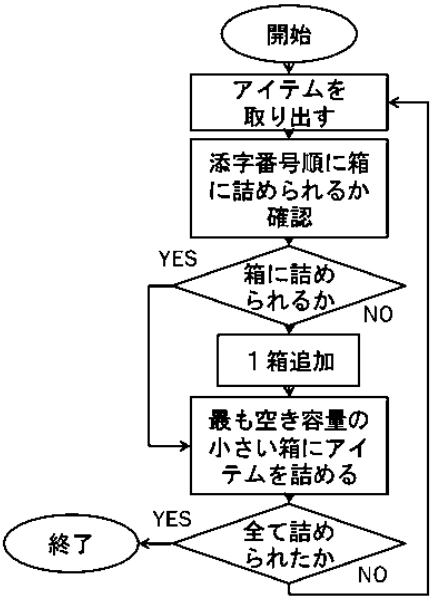

図 2.4　BFD のフローチャート

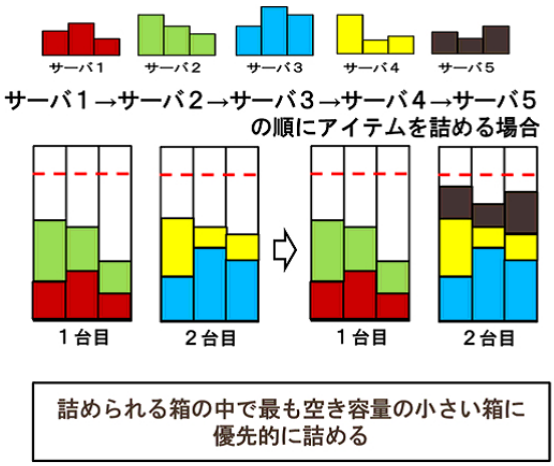

図 2.5　BFD の詰め方の例

2.4.3　WFD(Worst-Fit Decreasing)

まず箱が 1 つしかない状態からはじめてアイテムが箱に詰められるかどうかを箱の添字番号順に調べ，最も空き容量の大きい箱に詰める動作を繰り返す．詰められなかった場合は箱を 1 つ追加してそれにアイテムを詰める．この過程のフローチャートとその例を図 2.6，図 2.7 に示す．

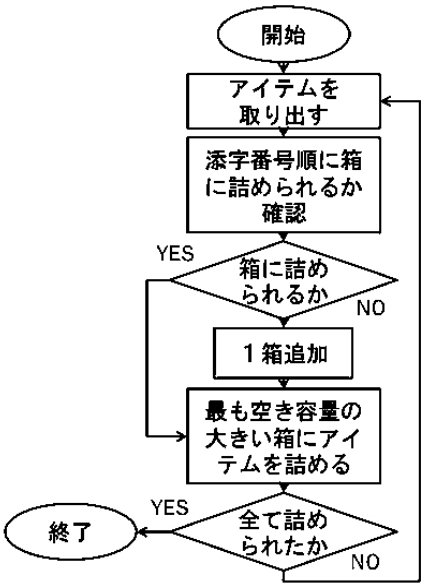

図 2.6　WFD のフローチャート

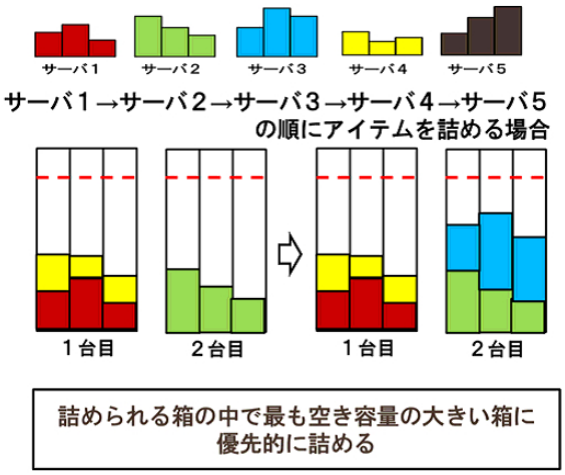

図 2.7　WFD の詰め方の例

2.4.4　NFD(Next-Fit Decreasing)

まず箱が 1 つしかない状態からはじめて一番新しい箱にアイテムが箱に詰められるかどうかを調べ，詰められるなら詰める動作を繰り返す．詰められなかった場合は箱を 1 つ追加してそれにアイテムを詰める．この過程のフローチャートとその例を図 2.8，図 2.9 に示す．

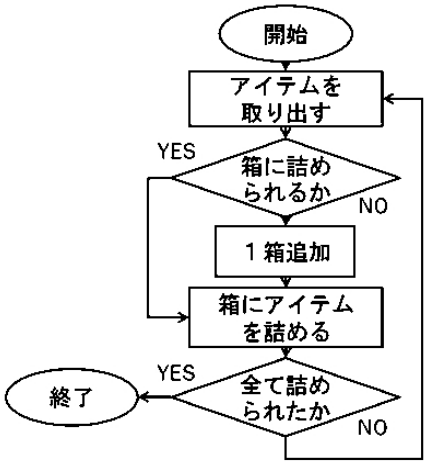

図 2.8　NFD のフローチャート

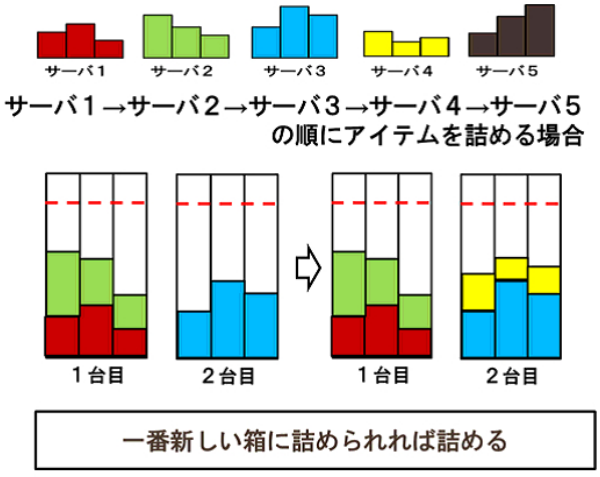

図 2.9　NFD の詰め方の例

2.4.5　4 つのアルゴリズムの特徴

4 つのアルゴリズムの特徴についてまとめたものを表 2.1 に示す.

表 2.1　4 つのアルゴリズムの特徴

アルゴリズム	特徴
FFD	詰められる箱の中で箱の最も添字番号の小さい箱に詰める
BFD	詰められる箱の中で最も空き容量の小さい箱に詰める
FFD	詰められる箱の中で最も空き容量の大きい箱に詰める
NFD	一番新しい箱に詰め，無理なら新しい箱に詰める

2.5　N 次元パッキング問題のためのアルゴリズム

N 次元パッキング問題における詰め込み手法も提案されており，Permutation Pack 法 [4] や Choose Pack 法 [4] などがある．それらのアルゴリズムを以下に示す.

2.5.1　Permutation Pack 法

まず箱が 1 つしかない状態からはじめて 1 つ目のアイテムを詰める．一番新しい箱の各計算資源毎に合計値を算出し，$b_3 \leqq b_1 \leqq b_4 \leqq b_2$ なら $a_2 \leqq a_4 \leqq a_1 \leqq a_3$ を満たすアイテムを選んで詰める動作を繰り返す．どれも詰められなかった場合は箱を 1 つ追加してそれにアイテムを詰める．この過程のフローチャートを図 2.10 に示す.

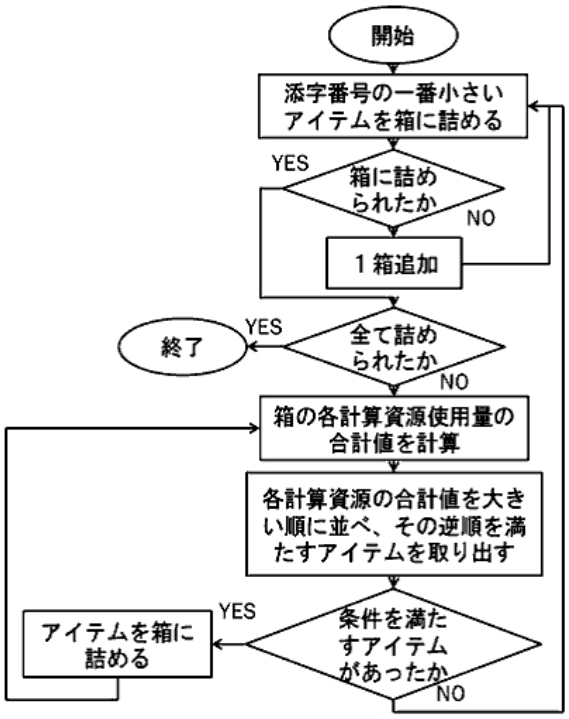

図 2.10　Permutation Pack 法のフローチャート

2.5.2　Choose Pack 法

まず箱が 1 つしかない状態からはじめて 1 つ目のアイテムを詰める．一番新しい箱の各計算資源毎に合計値を算出し，$b_3 \leqq b_1 \leqq b_4 \leqq b_2$ なら $a_2, a_4 \leqq a_1, a_3$ を満たすアイテムを選んで詰める動作を繰り返す．どれも詰められなかった場合は箱を 1 つ追加してそれにアイテムを詰める．この過程のフローチャートを図 2.11 に示す．

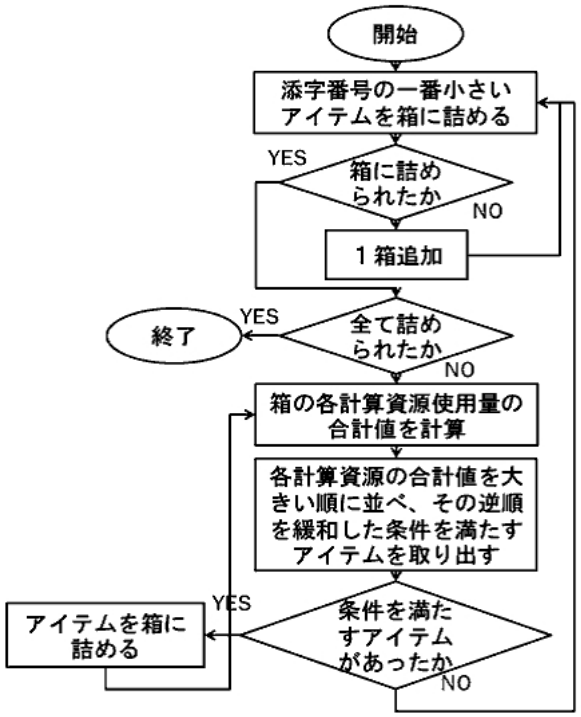

図 2.11　Choose Pack 法のフローチャート

2.5.3　Permutation Pack 法の例と Choose Pack 法との違い

Permutation Pack 法の例と Choose Pack 法との違いを図 2.12 に示す.

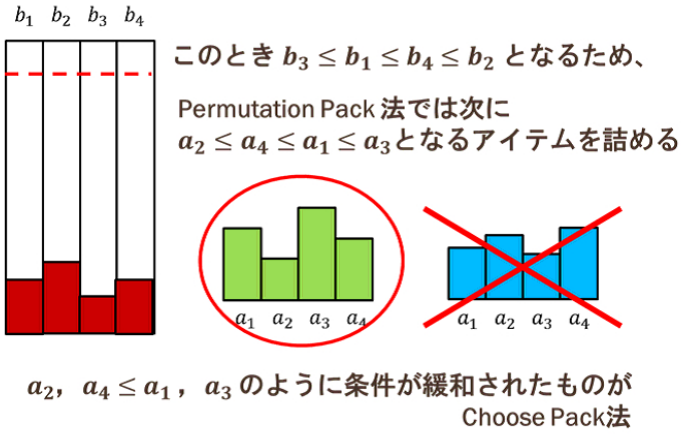

図 2.12　Permutation Pack 法の例と Choose Pack 法との違い

2.6　時系列を考慮したパッキング問題

実用上の問題では時系列を考慮したパッキング問題も解ける必要があり，各計算資源が時系列中のすべての時間帯で統合先サーバの容量を超過してはならない．そのとき，時系列を考慮したパッキング問題の次元数は時系列長と計算資源の種類数との積となる．計算資源の種類は CPU，メモリ，ディスク，ネットワークの 4 種類とし，30 分ごとのデータがあると仮定する．これを 1 日の時系列で考えた場合，$4 * 48 = 192$ もの次元数になり，このような次元数の問題には Permutation Pack 法や Chose Pack 法のような手法では詰め込むアイテムを見つけるための計算時間がかかりすぎるため適用できない．また，単純に各計算資源の使用量の積で降順にソートする手法を用いて FFD や BFD などの時間のかからない詰め方を適用しても [3] で Roy らの結果が示しているように統合対象サーバの各計算資源使用量にばらつきがあると各統合先サーバの空き容量の断片化によって必要な統合先サーバ数が増加してしまう傾向があるため，次元数の多い問題への適用が難しいとされている．

また，従来の FFD アルゴリズムを用いて改良型 FFD のように次元数の多い問題にも適

用できるアルゴリズムも提案されている．改良型 FFD では，箱が増える要因となるアイテムを保持し，優先的に詰める仕組みを用いることで解の評価を行う．各計算資源の使用量に関係なく詰められるかどうかだけで増える要因となるアイテムを判別しているため，数百次元の問題に対しても計算時間を増やすことなく適用が可能となっている．2 次元パッキング問題においては従来の FFD アルゴリズムと比較して 20%もの削減を可能としている [2]．改良型 FFD の詳しい流れは次章で説明する．

既存研究から詰め込み手法として時間のかからない従来の FFD などのアルゴリズムで評価を行い，ソートではなく局所探索による改善法を用いて解の改善を行うメタヒューリスティックを適用することで短時間での近似解を得られる可能性があると考えられる．そこで，本研究ではメタヒューリスティックを適用し，その有効性の比較・検証を行う．

第 3 章

ヒューリスティックなアルゴリズム

基本的にある問題に対して解法が存在する場合，その解法が適用できる範囲はその問題に対してのみである．しかし，厳密解ではなくそれに近い近似解を求める場合には，ヒューリスティックなアルゴリズムを用いることが可能となる．ヒューリスティックなアルゴリズムの中でも問題に依存せず，様々な問題で適用することができるものはメタヒューリスティックと呼ばれ，それらは局所探索を基に解の改善を行うものである．メタヒューリスティックでは，以下の動作の反復により解を導き出す.

1. 過去の探索の履歴を利用して新たな解を生成する.
2. 生成した解を評価し次の解に必要な情報を取り出す.

すなわち，生成された解のどのような情報を探索履歴として保存し，探索履歴をどのように利用して新たな解を生成するかに対する様々なアイデアの集合がメタヒューリスティックである [5].

本研究では，メタヒューリスティックとして独自の改善法を持つランダムサンプリングを改良したものと遺伝的アルゴリズム，それらのアルゴリズムの改善法による効果が得られているか確認するためのランダム探索，またはヒューリスティックなアルゴリズムとしてすでに計算資源数の少ない問題で良い結果を示している改良型 FFD の 4 つのアルゴリズムを用いて比較・検証を行う.

- 改良型ランダムサンプリング
- 遺伝的アルゴリズム
- ランダム探索
- 改良型 FFD

3.1　改良型ランダムサンプリング

3.1.1　ランダムサンプリング

ランダムサンプリングはメタヒューリスティックの 1 つである．順列組み合わせによる総当たり手法を基にした単純なアルゴリズムであるため，文献等ではランダム探索のように書かれていないことも多い．ランダムサンプリングでは局所探索による改善法として，ランダムに選択した要素から得た部分要素の順列組み合わせパターンを用いて改善を行う．そのため，仮想サーバ配置問題のように解候補を順列で表現できる問題では基となる解候補から順列を崩すことなく時間をかけずに一度に複数の別の解候補を作ることが可能となる．また，ランダムに選択された要素を用いるため，評価される解候補はそれに依存してしまう．それにより世代数を増やしても解候補が変わらないといったことも起きやすい．その場合，選択する要素数を増やすことでできる限り解候補の変動を促すことが可能であり，解候補の要素数と同じ数を選択すると総当たり手法となる．

基本的な流れは以下のようになる．選択された要素から作られる順列組み合わせパターンの例を図 3.1 に示す.

1. ランダムな解候補を 1 つ生成する.
2. 解候補からランダムにユーザが指定した数だけ要素を選択する.
3. 選択された部分要素からなるすべての順列組み合わせパターンを入れ替えた解候補の評価を行う.
4. 最も評価の高い解個体を残す.
5. 終了条件を満たすまで 2〜4 を繰り返す.

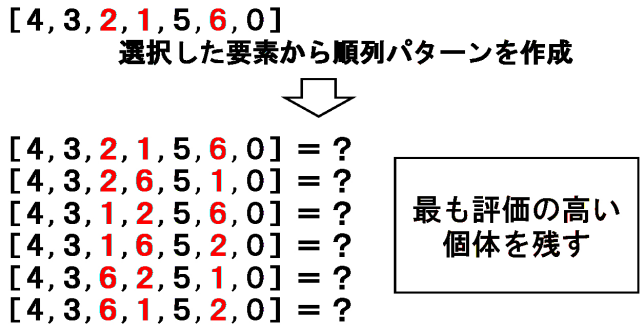

図 3.1　選択された要素から作られる順列組み合わせパターンの例

3.1.2　ランダムサンプリングの問題点

ランダムサンプリングは 1 つの解候補からスタートするタイプのメタヒューリスティックであるため，近似解に到達するためには問題サイズの大きな問題には一度に選択する要素の数を増やすか実行時間を増やす必要が生まれる．しかし，選択する要素数を増やすことで解を評価する回数が大きく増えてしまうため，計算時間が制限されている問題では何度も繰り返し行うことで解を改善する仕組みを生かすことが出来なくなる．また，計算時間が短く制限されるような問題では解の評価の中で良い評価にするための仕組みを追加されることがあるため，一度の評価に単純な評価手法に比べて計算時間が多くかかってしまうことも多い．この理由から選択される要素数は簡単に多くすることが出来ないと考えられる．

選択する要素数による評価回数の増加を表 3.1 に示す．

表 3.1　選択する要素数による評価回数の増加

選択する要素数	1 世代における評価回数
3	6
4	24
5	120
6	720
7	5040
8	40320

3.1.3　改良型ランダムサンプリング

ランダムサンプリングは 1 つの解候補から局所探索のみを行うため，問題サイズの大きい問題では選択する要素を増やすか時間をかけないとあまり良い結果を示さない．また，初期に生成された解候補の善し悪しに左右されやすいといった欠点が存在する．そこで，初期解候補をできる限り選別し，大域探索後に局所探索を行えるように改良を行った．しかし，選別した 1 つ，または複数の解候補に少ない選択要素数でランダムサンプリングを適用することを前提としているため，基のランダムサンプリングと同じく解の改善が行われるかどうかは乱数に依存している．

基本的な流れは以下のようになる．この過程のフローチャートを図 3.2 に示す．

1. ユーザが指定した任意の回数だけランダム探索を行う．
2. ランダム探索終了後，その時点で得られた同じ最良解を持つ解個体集団を用意する．
3. 解候補集団の解候補それぞれにランダムサンプリングを適用する．
4. 最良解が更新されれば，その時点で得られた同じ最良解を持つ解個体集団を更新する．
5. 終了条件を満たすまで 3 と 4 を繰り返す．

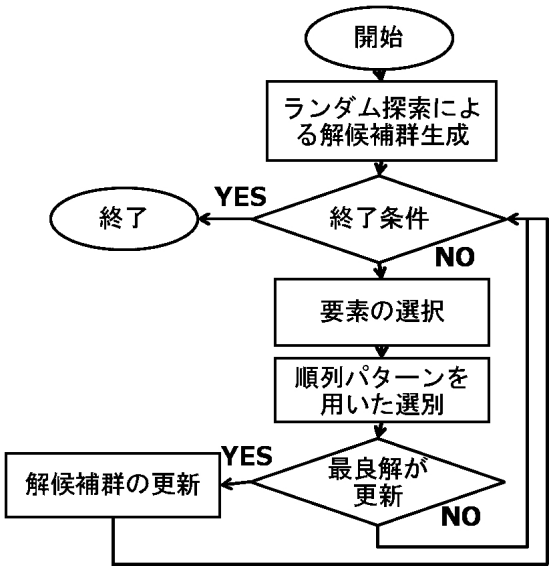

図 3.2　改良型ランダムサンプリングのフローチャート

3.1.4　改良型ランダムサンプリングの収束時間

統合対象サーバ 100 台の問題に対して乱数の種を 100 種類用い，改良型ランダムサンプリング (RAM2) とランダム探索 (RS) のそれぞれの解の収束時間を比較した．RAM2 のランダム探索回数は 100，選択する要素数は 3 として解として出力される必要な統合対象サーバ数の理論下限値誤差 1 以下の解を探索するまでの計算時間で比較を行う．このとき，統合先サーバの閾値はすべて 80%とする．

理論下限値は，統合対象サーバの各計算資源使用量の合計を閾値で割った値を下回らない整数であり，各計算資源で最も大きい値をそのテストデータにおける必要な統合先サーバ数の理論下限値とする．テストデータの生成方法は 5 章，解候補の順列表現方法については 4 章で説明する．

表 3.2　理論下限値誤差 1 以下の解を探索するまでの計算時間の比較

アルゴリズム	平均 (秒)
改良型ランダムサンプリング	6.62125
ランダム探索	10.25196

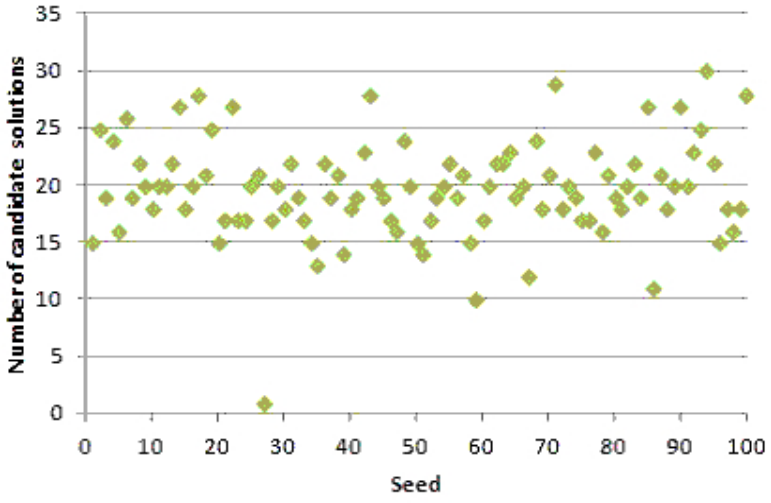

図 3.3　問題サイズ 100-初期解候補群生成時点での解候補数-

表 3.1 から統合対象サーバ数 100 の問題では RAM2 は RS に比べて 3.6 秒前後早い計算時間で理論下限値誤差 1 以下の解を探索できていることが分かった．まったく局所探索による改善法を用いないランダム探索より早い計算時間で理論下限値誤差 1 以内の解が探索できていることから局所探索による改善法は仮想サーバ配置問題に対して有効である可能性があると考えられる．また，図 3.3 から最初に行われる 100 回のランダム探索によりほぼすべての乱数で 10〜30 の解候補を生成できていることが確認され，大域探索により初期解候補を複数個絞ることに成功していると考えられる．

3.2　遺伝的アルゴリズム

遺伝的アルゴリズム (Genetic algorithm：GA) とは，問題の解を遺伝子の配列で表現し，自然界の進化と淘汰を模倣した近似解法である．親となる解候補に対して，交叉, 突然変異を行い，子となる新しい解候補を作成する．生まれた子に優劣を設け，優秀な解候補を次世代に優先的に残し，さらなる交叉，突然変異を行うことで優秀な子で形成された最終世代ができ，その世代の最良解をその問題に対する近似解とするものである [6][7]．GA による改善法の大きな特徴は解候補集団を用いた交叉による局所探索である．交叉では 2 つの解候補

を掛け合わせ新たな解候補を生成し，この操作により 1 世代で一度に複数箇所の局所探索を可能にしている．

GA はまず，初期集団の生成から始まり，世代毎に選択，交叉，突然変異，エリート戦略の遺伝的操作を行っていく．これらの操作を，終了条件を満たすまで繰り返すことで解の改善を行う．

この過程のフローチャートを図 3.4 に示す．

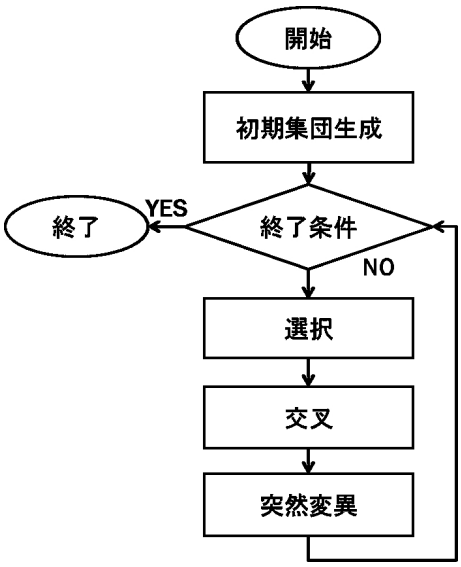

図 3.4　遺伝的アルゴリズムのフローチャート

3.3　ランダム探索

ランダム探索 (Random Search： RS) は最もシンプルな大域探索アルゴリズムである．ユーザが指定した数だけランダムな解候補を生成し，最も評価が良い解を残す．局所探索により解を改善していく仕組みがないため，ランダムに生成された解候補を終了条件を満たすまで延々と比較・評価し続けるアルゴリズムである．

この過程のフローチャートを図 3.5 に示す．

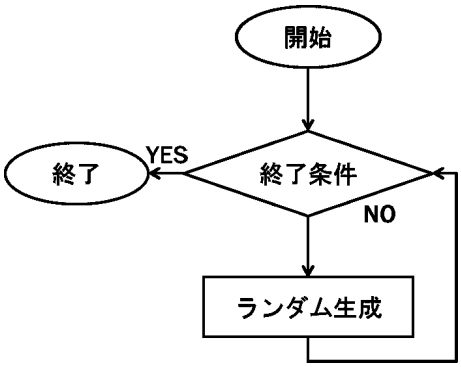

図 3.5　ランダム探索のフローチャート

3.4　改良型 FFD

改良型 FFD では，箱が増える要因となるアイテムを保持し，優先的に詰める仕組みを用いることで解の評価を行う．各計算資源の使用量に関係なく詰められるかどうかだけで増える要因となるアイテムを判別しているため，数百次元の問題に対しても計算時間を増やすことなく適用が可能となっている．2 次元パッキング問題においては従来の FFD アルゴリズムと比較して 20%もの削減を可能としている [2].

基本的に箱に詰める方法は FFD と同じである．箱の数は理論下限値から始める．アイテムを格納する T’ と T の順序付き集合を 2 つ用意され．T’ のアイテムが優先的に詰められる．初期状態では T にすべてのアイテムがあり，下限値の箱数からはじめて箱に入れられなかったアイテムを T から T’ に移動してパッキングをやり直す．T’ の要素数がユーザーが与える定数 MAXr に到達するまで繰り返して詰められなかった場合は 1 個箱を追加して同様の処理を繰り返す．T’ に箱から溢れるアイテムを格納することで必要とする統合先サーバ数の増加要因となるアイテムを MAXr 個まで検出できる [2].

この過程のフローチャートを図 3.6 に示す.

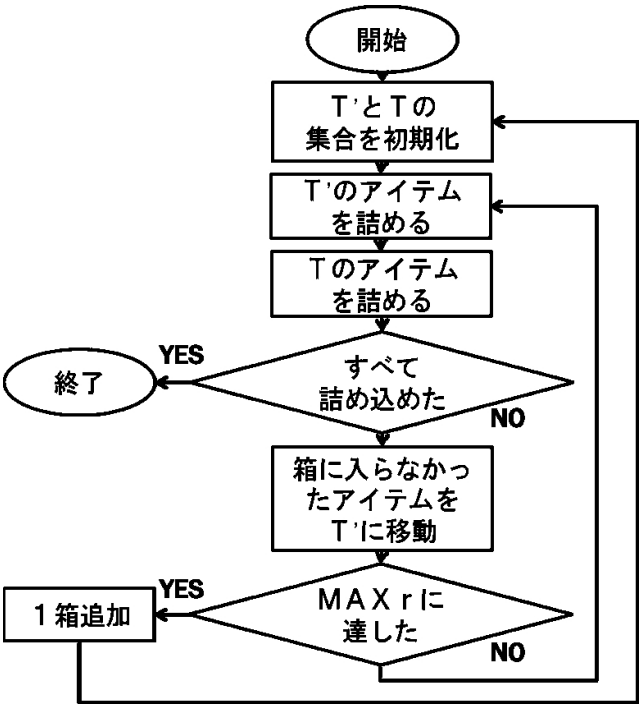

図 3.6　改良型 FFD のフローチャート

第 4 章

ヒューリスティックなアルゴリズムによる 5 次元パッキング問題の解法

コンピュータを用いて問題を解くためには，その解候補となるものを配列の並びによって表現する必要がある．解候補の表現方法は問題によって様々であり，解候補の表現方法とその評価手法が揃ってはじめてアルゴリズムを適用して問題を解くことが可能となる．この章では解候補の表現方法とその評価手法について説明を行う．また，遺伝的アルゴリズムにおける遺伝的操作について説明を行う．

4.1 解候補の表現方法とその評価手法

仮想サーバ配置問題における解候補からは以下の情報が分かる必要がある．

- どの統合先サーバにどの統合対象サーバが配置されるか．
- 必要な統合先サーバ台数は何台か．

そのため，すべてのアルゴリズムは統合対象サーバの台数 n の長さを持つリスト $[s_1, s_2, \cdots, s_i](1 \leqq s_i \leqq n)$ を順列で表現し，s_1 から順に統合先サーバに FFD アルゴリズムを用いて詰めることで評価を行うこととした．

本研究で用いる解候補の表現方法について図 4.1 の例を用いて説明する．

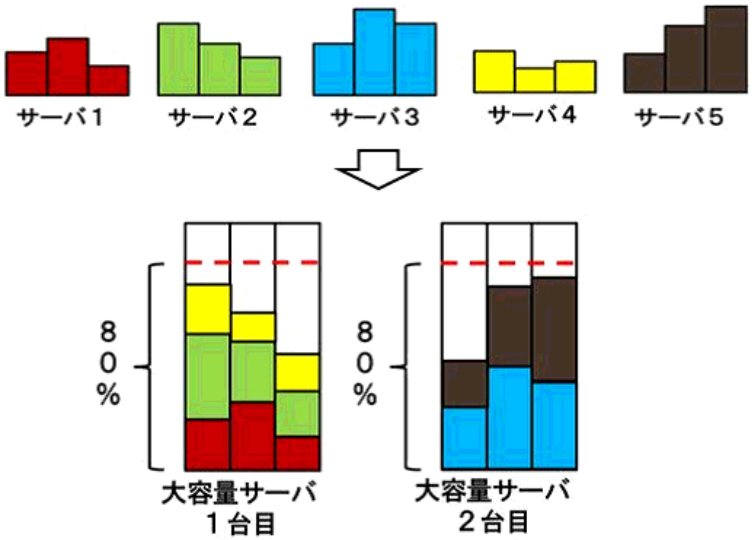

図 4.1　仮想サーバ配置問題の例

図 4.1 を例とすると $[1, 2, 3, 4, 5]$ の順に統合対象サーバを詰めることになる．FFD アルゴリズムを用いて順に詰めていくと 1，2 番目の統合先サーバは問題なく 1 台目の統合先サーバに詰められる．しかし，3 番目の統合対象サーバが 1 台目の統合先サーバの閾値を越えてしまうため，統合先サーバを 1 台追加してそれに詰める．そして，4 番目の統合対象サーバは 1 台目の統合対象サーバに詰められ，5 番目の統合対象サーバが 1 台目の統合先サーバの閾値を越えてしまうため，2 台目の統合先サーバに詰められる．評価の結果，必要な統合先サーバ数は 2 であることが分かる．

このように FFD アルゴリズムを用いて評価を行うことで，解として成り立たない解が生まれることなく，どの統合先サーバにどの統合対象サーバが配置されるか，必要な統合先サーバ台数は何台かという情報を得ることが可能となる．

4.2　遺伝的アルゴリズムの遺伝的操作

4.2.1　初期集団の生成

初期集団の生成には FFD アルゴリズムを用いる．ランダムに生成された解候補に対して FFD アルゴリズムを適用した後，決められた割当を解除し，そのまま詰める統合対象サーバの順番として戻すことで，詰める際により時間のかからない解候補に編集しなおす．

4.2.2　適応度

適応度は，FFD アルゴリズムで出力された統合先サーバ台数が少ないほど良いとした．ランダムに生成した初期解候補集団の中で最も FFD アルゴリズムで出力された統合先サーバ台数の多かった値を BAD とする．適応度を t とし，その個体の FFD アルゴリズムで出力された統合サーバ台数を Servers とした場合，$t = \frac{1}{(BAD-Servers)^2}$ となっている．

4.2.3　ルーレット選択

本研究では，選択法としてパラメータの設定を必要としないルーレット選択を用いた．ルーレット選択とは，適応度に比例した確率で個体を選択する手法である．ルーレット選択では解候補集団の適応度の合計値とそれぞれの適応度を基にどの解候補が選ばれるかの確率を設定する．それぞれの適応度とその合計値を除算した値がそれぞれの解候補が選ばれる確率となる．

例えば，解候補となるそれぞれの適応度を 1，2，3，4 とし，合計値が 10 の場合，それぞれが選ばれる確率は 10%, 20%, 30%, 40%となる．また，評価方法を工夫することにより，選択される解候補の傾向を変化させることが可能である．この手法では，確率的に選択を行っているため，適応度の低い解候補も選ばれる可能性が残り，他の選択法よりも収束が起こりにくいと言える．

4.2.4　部分一致交叉

交叉は GA で最も重要な遺伝的操作である．2 つの親となる解候補を掛け合わせることで子となる新たな解候補を生成する．交叉方法には様々なものがあり，方法によって受け継ぐ要素の並びが変化する．部分一致交叉 (Partially Mapped Crossover: PMX) は，ランダムなカットポイントを 1 つ，もしくは 2 つ選び，一方の親のその部分集合の要素を受け継ぐ．PMX の一例を図 4.2，図 4.3，図 4.4 で説明する．PMX ではまず，図 4.2 のように親となる解候補 2 つから交叉範囲を決め, その交叉範囲内の要素をそれぞれ関連付ける．次に，図 4.3 のように関連付けた要素をその解候補の中で入れ替える．この操作はそれぞれの親で行う．結果として図 4.4 のようにそれぞれの親の特徴を引き継いだ子ができる．

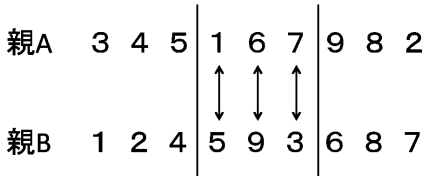

図 4.2　PMX による遺伝子の関連付け

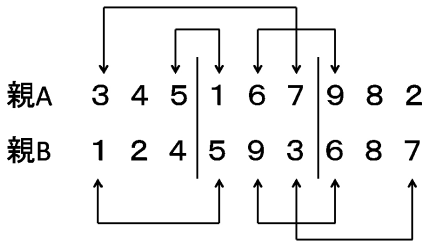

図 4.3　PMX により関連付けられた遺伝子の移動

子A　7 4 1 5 9 3 6 8 2
子B　5 2 4 1 6 7 9 8 3

図 4.4　PMX により生まれる子の例

4.2.5　突然変異

GA では世代を重ねることで，選択と交叉によりほとんど同じ解候補ばかりの集団になってしまう可能性がある．ほとんど同じ解候補を持つ集団になってしまった場合，一度に複数の解候補の局所探索を行う交叉にほとんど意味がなくなってしまう．そのため，個体集団の多様度を保つ目的で突然変異が行われる．しかし，交叉により生成された染色体の特徴を壊す恐れがあることや突然変異確率が高ければ高いほどランダム探索になり遺伝的アルゴリズムを用いる意味が薄れてしまうことから，基本的に突然変異確率は染色体の特徴を壊さない程度の低確率に設定されることが多い．

本研究では突然変異としてランダムなカットポイントを 2 つ選び，その 2 点間の要素をランダムにシャッフルする手法を用いた．

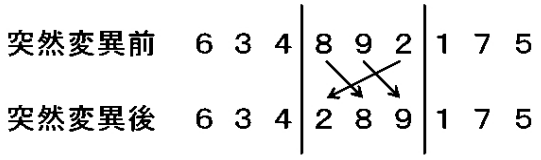

図 4.5　突然変異の例

第 5 章

予備実験

予備実験としてパラメータ設定のため予測される最大評価回数の確認と改良型ランダムサンプリングの性能比較を行った.

5.1 テストデータの生成

テストデータは文献 [2] のランダム生成方法を基に生成を行った．各統合対象サーバの各計算資源の値はすべて統合先サーバの最大使用量を 100%とした場合の割合で与えられる．CPU 性能使用量，メモリ使用量，ディスク性能使用量，ディスク容量使用量，ネットワーク性能使用量がすべて平均値以上か平均値以下になる確率 P を導入し，以下に示すアルゴリズムを用いて使用量のデータを生成した.

$for\ i \leftarrow 1\ to\ m\ do$

$pc_i \leftarrow rand(2pc)$; $pm_i \leftarrow rand(pm)$; $pd1_i \leftarrow rand(pd1)$;

$pd2_i \leftarrow rand(pd2)$; $pn_i \leftarrow rand(pn)$; $r \leftarrow rand(1.0)$;

$if(r < P \wedge pc_i \geqq pc) \vee (r \geqq P \wedge pc_i < pc) then$

$pm_i \leftarrow pm_i + pm\ fi$

$pd1_i \leftarrow pd1_i + pd1\ fi$

$pd2_i \leftarrow pd2_i + pd2\ fi$

$pn_i \leftarrow pn_i + pn\ fi$

$end\ for$

$rand(x)$ は範囲 $[0, x)$ の double 型の一様乱数を返す関数，pc, pm, $pd1$, $pd2$, pn はそれぞれ CPU 性能使用量，メモリ使用量，ディスク性能使用量，ディスク容量使用量，ネットワーク性能使用量の平均使用量を表している．今回の実験及び比較・検証では確率 P を 0 として正の相関が低いか，負の相関が高いものをテストデータとした．また，各計算資源の平均使用量はすべて 15%とした．

5.2　予備実験環境

予備実験を行った環境は以下の通りである．

CPU : INTEL(R)CORE(TM)i5-3230M CPU @2.60GHZ

MEMORY : 1 GByte

OS : Ubuntu 14.04

コンパイラ: Python 2.7.6

5.3　300 秒間で実行される評価回数

仮想サーバ配置問題ではエンジニアによる現場での提案書の作成や複数回実行することを考えると計算時間は数分以内である必要があるため，300 秒間の実行により比較を行うこととする．

統合対象サーバ 500 台と 1000 台の問題に対して乱数の種を 100 種類用い，最も評価回数が多くなると考えられるランダム探索における解候補の評価回数を確認した．

表 5.1　300 秒間で実行された評価回数-ランダム探索-

問題サイズ	評価回数
統合対象サーバ 500 台	約 6000 回
統合対象サーバ 1000 台	約 1500 回

表 5.1 を見るとどちらの問題サイズでも最大で評価できる回数は 1 万を越えず，この結果から 300 秒間という短い時間ではあまり多く解候補の評価を実行することはできないことが分かる．そのため，改良型ランダムサンプリングにおけるランダム探索回数は 500〜1000 回が限界であり，選択する要素数が 4 以上の場合，統合対象サーバ 500 台では約 200 世代以下，統合対象サーバ 1000 では約 25 世代以下になると考えられ，少ない世代分しか実行できないことになる．よって，ランダム探索回数は 1000 回，選択される要素数は 3 とする．また，遺伝的アルゴリズムにおいても集団サイズを 100 とした場合，統合対象サーバ 500 台では約 60 世代，統合対象サーバ 1000 では約 10 世代になると考えられる．よって，少ない世代数で改善法となる交叉に重点をおいて実行するため，交叉確率 90%，突然変異確率は 10%とする．

5.4　ランダムサンプリングと改良型ランダムサンプリングの比較

統合対象サーバ 500 台と 1000 台の問題に対して乱数の種を 100 種類用い，ランダムサンプリング (RAM)，改良型ランダムサンプリング (RAM2) のそれぞれの試行平均の推移を比較した．RAM2 のランダム探索回数は 1000，選択する要素数はどちらも 3 として 300 秒間の試行を繰り返した．このとき，統合先サーバの閾値はすべて 80%とする．

図 5.1 と図 5.2 の結果を見るとどちらの問題サイズでも試行平均では RAM2 が良い平均値を示している．仮想サーバ配置問題において改良型ランダムサンプリングに行った改良により基のランダムサンプリングに比べて良い平均値を出力する可能性があると考えられる．

5.4　ランダムサンプリングと改良型ランダムサンプリングの比較

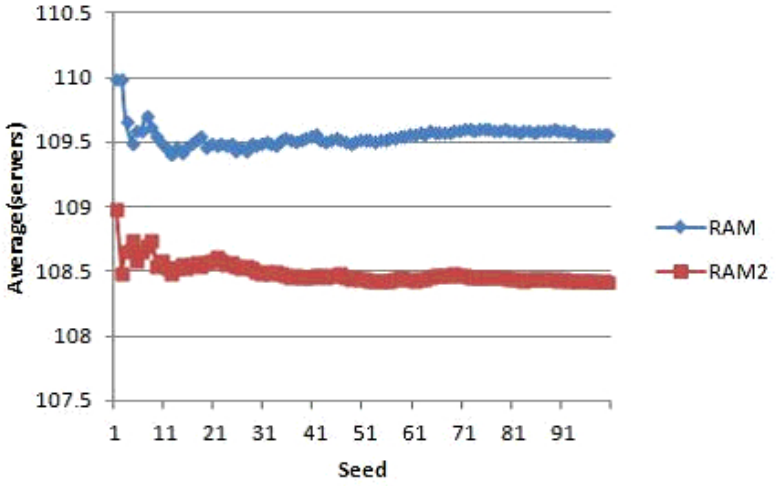

図 5.1　問題サイズ 500-試行平均の推移-

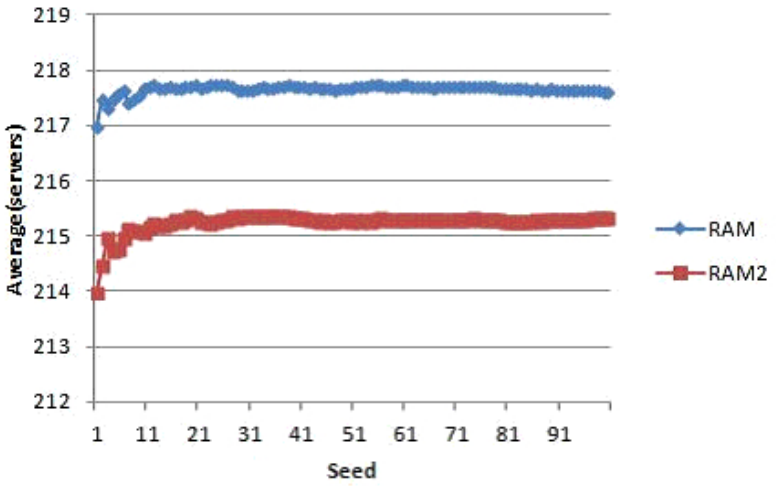

図 5.2　問題サイズ 1000-試行平均の推移-

第 6 章

比較・検証

仮想サーバ配置問題に改良型ランダムサンプリング (RAM2)，改良型 FFD(FFDR)，ランダム探索 (RS)，遺伝的アルゴリズム (GA) の 4 つのアルゴリズムを適用して出力される解を基に比較・検証を行った．

6.1 比較・検証環境

比較・検証を行った環境は以下の通りである．

CPU : INTEL(R)CORE(TM)i5-3230M CPU @2.60GHZ

MEMORY : 1 GByte

プロセッサー数 : 1

OS : Ubuntu 14.04

コンパイラ: Python 2.7.6

6.2 比較検証結果

6.2.1 統合対象サーバ台数 500 の結果

統合対象サーバ 500 台の問題に対して乱数の種を 100 種類用い，改良型ランダムサンプリング (RAM2)，改良型 FFD(FFDR)，ランダム探索 (RS)，遺伝的アルゴリズム (GA) の 4 つのアルゴリズムでそれぞれの試行平均と試行分散の推移を比較した．RAM2 のランダム探索回数は 1000，選択する要素数は 3，FFDR の MAXr は統合先サーバ台数と同数，GA

の集団サイズは 100，交叉方法は PMX，交叉確率 90%，突然変異確率 10% として 300 秒間の試行を繰り返した．このとき，統合先サーバの閾値はすべて 80%とする．試行平均と試行標準偏差の推移を図 6,1，図 6.2 に示す．また，標準偏差を σ とした場合の ±2σ 以内を反映したグラフを図 6.3 に示す．

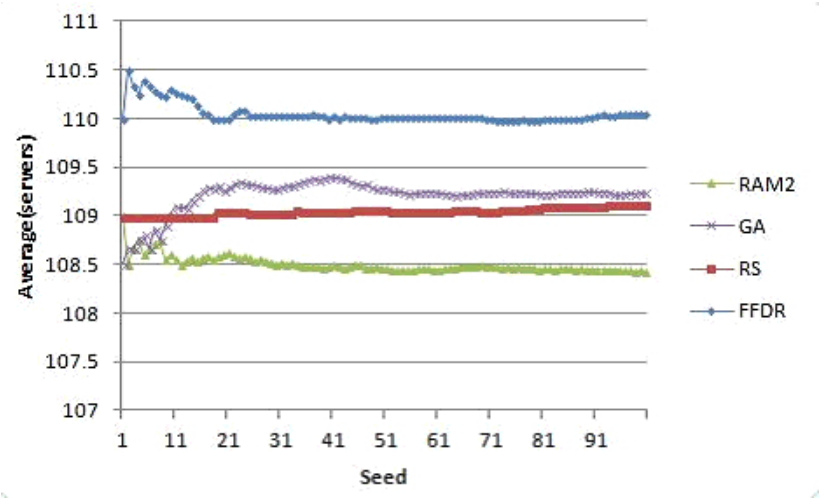

図 6.1　問題サイズ 500-試行平均の推移-

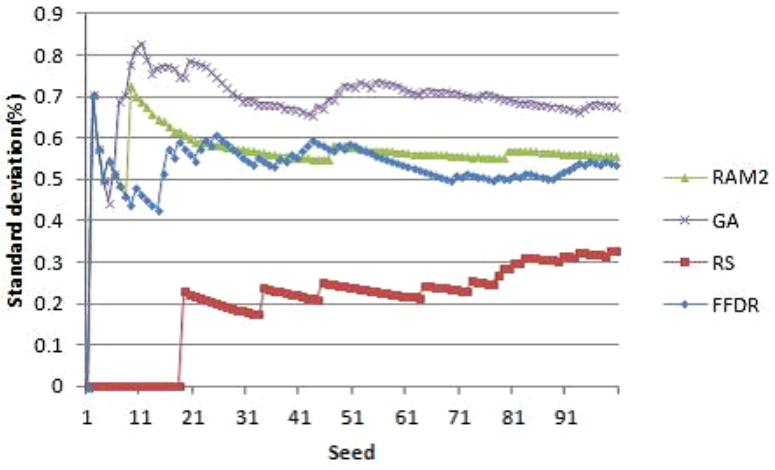

図 6.2　問題サイズ 500-試行標準偏差の推移-

標準偏差を σ とした場合，図 6.1 から試行平均の最も良い RAM2 でも図 6.2 と図 6.3 を見ると ±2σ 以内では他のアルゴリズムの ±2σ 以内と範囲が被ってしまうことが分かっ

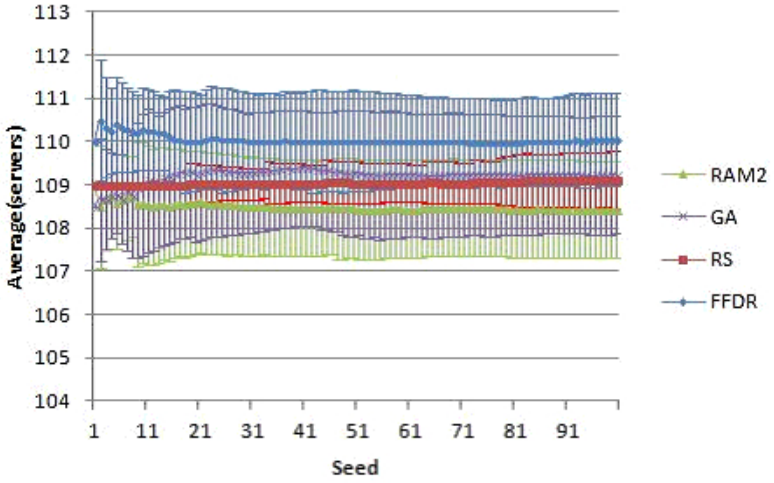

図 6.3　問題サイズ 500-±2σ 以内を反映-

た．データが正規分布に従う場合，平均値 ±2σ 以内に約 95.4%の出力される解が存在することになるため，この問題サイズでははっきりとした差は見られなかった．

6.2.2　統合対象サーバ台数 1000 の結果

問題サイズ 500 では差が確認できなかったため，さらに統合対象サーバ 1000 台の問題に乱数の種を 100 種類用いて 300 秒間の試行を繰り返した．その試行平均と試行標準偏差の推移を図 6,4，図 6.5 に示す．また，標準偏差を σ とした場合の ±2σ 以内を反映したグラフを図 6.6 に示す．

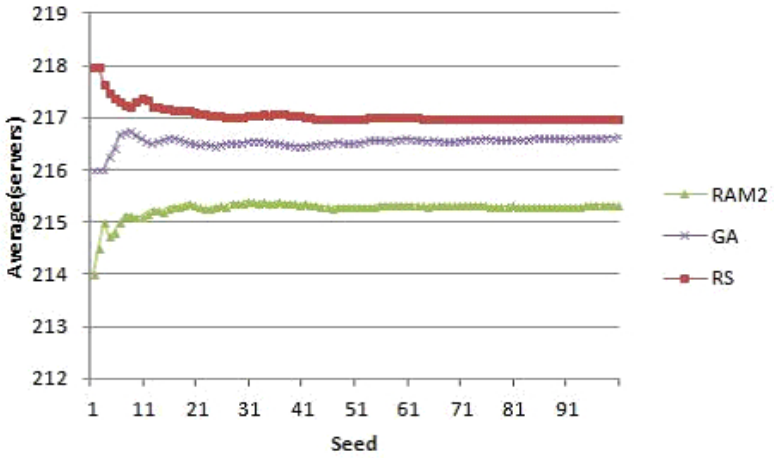

図 6.4　問題サイズ 1000-試行平均の推移-

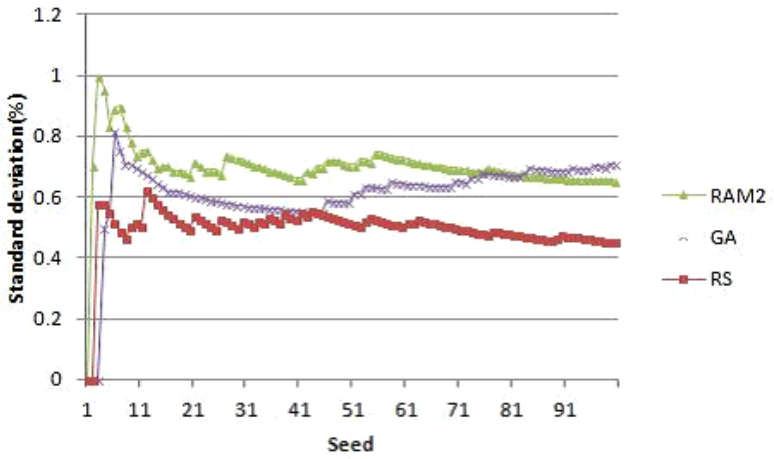

図 6.5　問題サイズ 1000-試行標準偏差の推移-

FFDR は問題サイズ 500 の問題ですでに 160〜280 秒前後の時間を必要としており，問題サイズ 1000 の問題では 1 回の試行に約 3500 秒必要とするため，300 秒以内に有効な解が得られなかった．図 6.4 を見ると試行平均では，問題サイズ 500 のときと同じく RSM2 が最も良い結果を示した．しかし，試行平均では問題サイズ 500 のときよりも差は生まれたが，この問題サイズでも図 6.5 と図 6.6 を見る限り $\pm 2\sigma$ 以内では他のアルゴリズムと範囲が被っており，出力される解の範囲にはっきりとした差はなかった．

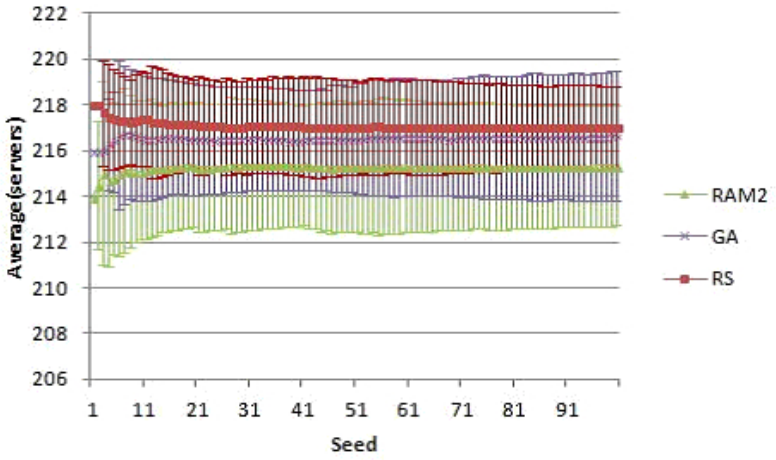

図 6.6　問題サイズ 1000-±2σ 以内を反映-

6.2.3　問題サイズ 200，500，1000 でそれぞれ 10 種類の問題毎の平均値

統合対象サーバ 1000 台の問題 10 種類に対して乱数の種を 10 種類用い，改良型ランダムサンプリング (RAM2)，改良型 FFD(FFDR)，ランダム探索 (RS)，遺伝的アルゴリズム (GA) の 4 つのアルゴリズムでそれぞれの問題の平均値を比較した.

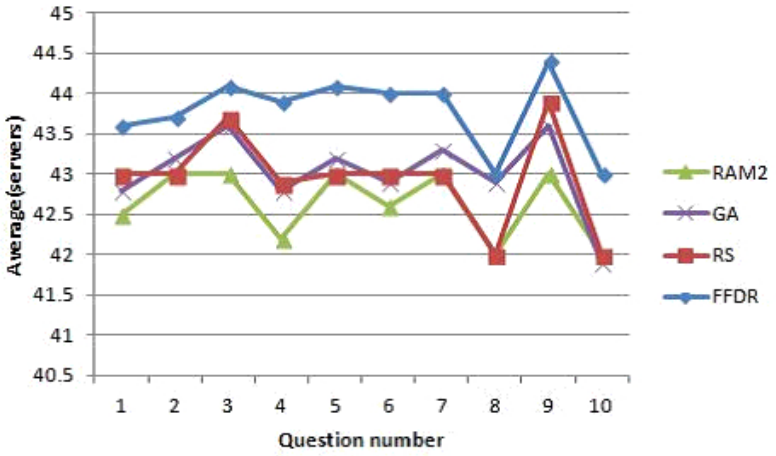

図 6.7　問題サイズ 200-問題毎の平均値-

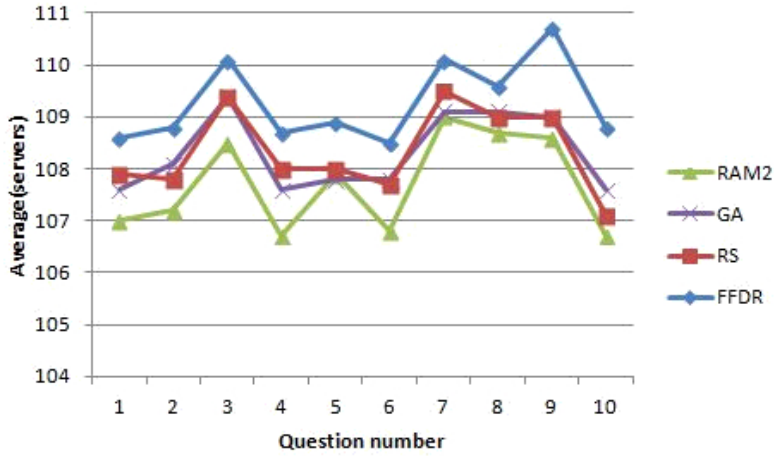

図 6.8　問題サイズ 500-問題毎の平均値-

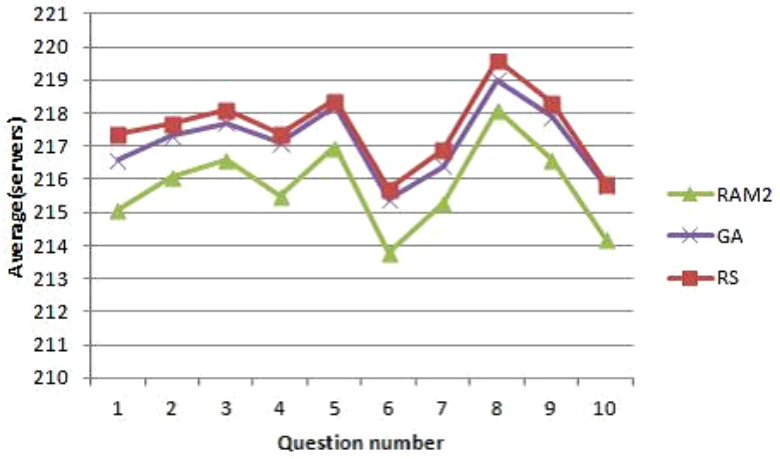

図 6.9　問題サイズ 1000-問題毎の平均値-

問題サイズ 200 の問題ではどのアルゴリズムにも大きな差は見られなかった．問題サイズ 200 の 1 問を除く問題サイズ 200，500，1000 のすべての問題で RAM2 が最も良い平均値を出力した．問題サイズが大きくなるほど RAM2 の平均値が他のアルゴリズムよりも良い結果を示す可能性があると考えられる．

第 7 章

考察

出力される解の平均値で見れば改良型ランダムサンプリングが良い結果を出力する可能性が高いことが分かったが，標準偏差を用いて統計的に見た結果では，解のばらつきから他のアルゴリズムと大きく差が出ない結果となった．そのため．出力される解のばらつきの原因とその解決案について考察を行った．

7.0.4　標準偏差で見た解のばらつきの考察

標準偏差を用いて統計的に確認した結果，平均値で見れば最も良い解を出力すると考えられる改良型ランダムサンプリングでも他のアルゴリズムの出力する解の範囲と大きく被ってしまうことが分かった．そのため，改良型ランダムサンプリングにおける標準偏差による解のばらつきの原因について考察する．

ランダムサンプリングのような局所探索手法は 1 点からスタートするため，計算時間が決まっている場合，基となる初期解候補による解のばらつきが考えられる．また，改良型ランダムサンプリングではランダム探索により，解候補群を生成することである程度選別された解を初期解候補としているため，より軽減されていると考えていた．しかし，実際に確認を行うと解候補群の最良値は大きく変わらないが，図 7.1，図 7.2 のようにランダム探索により得られた解候補群の解候補数に差が生まれていることが分かった．統合対象サーバ数が 1000 のように問題サイズが大きくなるほど顕著に現れ，解候補群の解候補数が 10 前後あるものや 1 か 2 しかないものなどバラバラでほとんど一定していないことが確認できた．おそらくこれが出力される解のばらつきの原因だと考えられる．

比較的良い解を示した Seed に多かった特徴は以下の 2 パターンである．今回の実験の中では最良値が良く解候補を多く持つパターンは存在しなかった．

- その時点での最良値は良くないが同じ値の解候補を多く持つ．
- その時点での最良値が良いが同じ値の解候補がほとんどない．

比較的悪い解を示した Seed に多かった特徴は以下のパターンである．

- 最良値に関係なく同じ値の解候補がほとんどない．

もしその時点での最良値が良くなくても同じ値の解候補を多く用意できれば良い解を示す可能性があると考えられる．

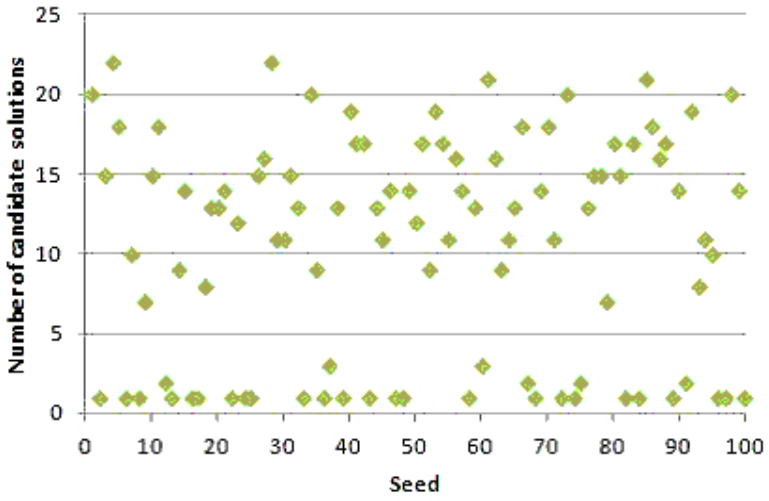

図 7.1　問題サイズ 500-初期解候補群生成時点での解候補数-

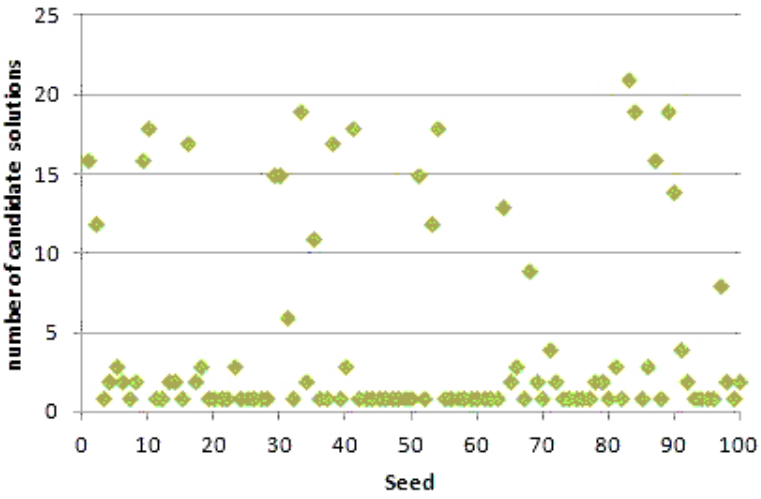

図 7.2　問題サイズ 1000-初期解候補群生成時点での解候補数-

そこで，ランダム探索で 1 つや 2 つしかその時点での最良値を持つ解候補が見つからなかったときに 1 つの解候補から複数の別の解候補を生成する手法を例を示す．1 つの解候補から複数の別の解候補が複数生成できれば一定数の解候補群を常に維持することが出来，それにより解のばらつきも抑えられる可能性がある．

7.1　1 つの解候補から同じ評価値を持つ別の解候補を生成する手法の例

解候補を評価した段階でどの統合先サーバにどの統合対象サーバが配置されるかが決まる．すべての統合先サーバの閾値は同じであるため，統合対象サーバの組み合わせによるセットを 1 つの集合とし，それをどの統合先サーバと配置を入れ替えてもても必要な統合先サーバ数は変わらない．これを利用して図 7.3 のように入れ替えた後，配置を解除して解候補の状態に戻せば同じ評価値を持つ別の解候補を生成できる．

7.1　1 つの解候補から同じ評価値を持つ別の解候補を生成する手法の例

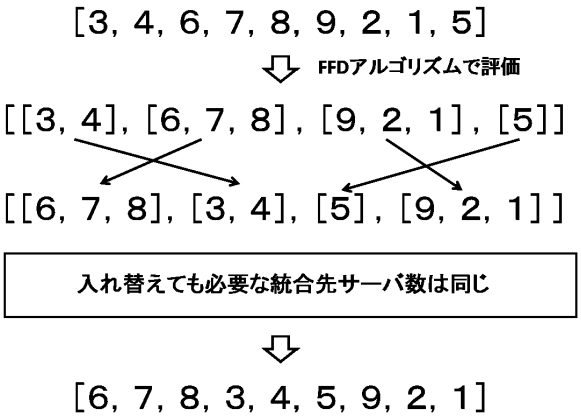

図 7.3　1 つの解候補から同じ評価値を持つ別の解候補を生成する手法の例

第 8 章

結論

計算資源の数を 5 とする仮想サーバ配置問題では問題サイズが 500 や 1000 の問題において 4 つのアルゴリズムを比較した結果，4 つのアルゴリズムにはっきりとした差は見られないが改良型ランダムサンプリングが他のアルゴリズムと比べて良い平均値を示す可能性があることが分かった．また，改良型ランダムサンプリングにおける標準偏差で見た解のばらつきの原因は，ランダム探索による解候補群の生成と最良解更新後の解候補群が定数維持できないことによるものであると考え，解候補群を常に複数維持する仕組みの例を示した．しかし，今回示した例では評価に時間がかかることから計算時間や評価回数に影響を与える可能性があるため，さらに計算時間のかからない手法を見つける必要があると考えられる．今後の課題として，時系列を考慮したさらに次元数の多い問題やさらに問題サイズの大きい問題に適用する必要があると考えられる．

謝辞

本研究は，著者が高知工科大学大学院工学研究科基盤工学専攻情報システム工学コース坂本研究室及び，福本研究室に所属中の 2013 年から 2015 年までに行った研究活動の記録である．

はじめに，学部生の時から引続きプログラミングの御指導やアドバイス，就職活動の時には暖かく励ましてくださいました坂本明雄教授に感謝を申し上げます．また，本研究の主査としてご迷惑ばかりおかけしましたが完成まで多大なアドバイスをしていただいた福本昌弘教授に感謝を申し上げます．

そして，本研究の副査をしていただいた吉田真一准教授，副査だけでなく就職活動でもとてもお世話になった妻鳥貴彦准教授に感謝を申し上げます．就職活動では，ご迷惑ばかりおかけした横山和俊教授に感謝を申し上げます．

最後に何不自由なく大学に通わせてくれて勉強や研究に打ち込める環境を与えてくれた両親に感謝します．

今までお会いしたすべての方々，本当にありがとうございました．

参考文献

[1] J.Carlier,F.Clautiaux,and A.Moukrim. New reduction procedures and lower bounds for the two dimensional bin packing problem with fixed orientation. Computers and operations reserch, Vol.34,No.8,pp.2223-2250, 2007.

[2] 網代育大，田中淳裕，NEC システムプラットホーム研究所，サーバ統合のための組み合わせ最適化アルゴリズムの提案と評価，2007.

[3] Gupta,R.,Bose,S.K.,Sundrrajan,S.,Chebiyam,M.and Chakrabarti,A.:A two stage heuristic algolithm for solving the server consolidation ploblem with item-item and bin-item incompatibility constraints,Proc.IEEE Int.Conf.on Services Computing(SCC'08),pp.39-46,2008.

[4] Leinberger,W.,Karypis,G.and Kumar,V.:Multi-capacity bin packing algolithms with applications to jobscheduling under multiple constraints,Proc.Int.Conf.on Parallel Processing(ICPP'99),pp.404-412,1999.

[5] 柳浦睦憲，茨木俊秀著，組み合わせ最適化～メタ戦略を中心として～，朝倉書店，2001.

[6] 棟朝雅晴著，遺伝的アルゴリズムーその理論と先端手法ー，森北出版株式会社，2008.

[7] 平野 廣美著，遺伝的アルゴリズムと遺伝的プログラミング，パーソナルメディア，2000.

[8] 久保幹雄・J.P. ペドロソ著，メタヒューリスティクスの数理，井立出版，2007.