データセンタ間のライブマイグレーションを考慮した仮想マシン最適配置技術

代表研究者　　橘　拓至　　福井大学　工学部　准教授

1 はじめに

本研究では，複数のデータセンタが接続された大規模データセンタネットワークに対して，ライブマイグレーション時に生じる遅延を抑制しながら，サーバ資源・リンク資源も有効利用可能な仮想マシンの最適配置技術を確立する．

データセンタは現在多様なデータの処理に利用されており，サーバ資源の有効利用が必要不可欠となっている[1,2,3]．ここで，データセンタ内のサーバは，仮想化技術によって 1 台のサーバに複数の仮想マシンを構築でき，各仮想マシンで異なるジョブを処理することができるようになっている．この仮想マシンを利用することで，限られたサーバ資源を有効利用でき，大量のジョブを処理することが可能になる．また，構築した仮想マシンは，ジョブの処理中に他のサーバへ移動させることができ，サーバのさらなる有効活用が実現できる．このような仮想マシンの移動はライブマイグレーションと呼ばれ，遠隔地の異なるデータセンタへもライブマイグレーションが可能である（図 1 参照）．

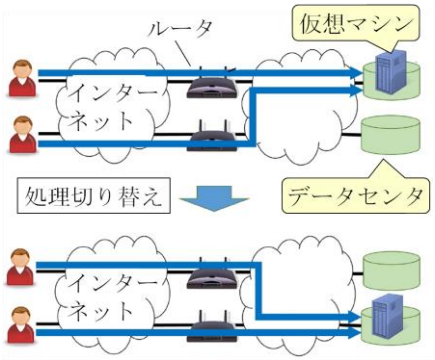

図 1 データセンタ間マイグレーションの利用例

このデータセンタ間ライブマイグレーションは，資源の有効利用だけでなく，データセンタの保守・管理や集中アクセス回避にも利用できる．一方，データセンタ間のマイグレーションを実行する場合，センタ間の距離やデータ量に応じて遅延が生じ，ジョブの処理性能が大きく劣化する．結果として，ジョブの処理性能要求を満足できず，データセンタ間のライブマイグレーションが実行できない可能性も高い．それゆえ，サーバ資源の有効活用はもちろんのこと，データセンタ間のライブマイグレーション時に大きな遅延が生じないような仮想マシンの配置が必要となる[4,5]．

そこで本研究では，データセンタ間のマイグレーションを考慮した仮想マシンの最適配置法を提案する．本提案方式では，データセンタ間の遅延時間とデータ量を考慮した最適化問題を定式化する．定式化する最適化問題では，データセンタ間のマイグレーション遅延を設定パラメータとして導入することで，マイグレーション遅延を考慮した仮想マシンの割り当てを実現する．この最適化問題に対して，遺伝的アルゴリズムによって最適解を導出することで仮想マシンの配置場所を決定する．また本方式では，データセンタ内の資源利用や輻輳も考慮する．さらに，最適化問題の解をマルコフ近似によって導出する方式についても検討する．

本方式によって，データセンタ間のライブマイグレーションを考慮した仮想マシン配置技術が確立できれば，大規模データセンタの発展・普及が期待でき，さらには大規模データセンタを利用する新たなアプリケーションの登場に対して多大な貢献が期待できる．また，提案技術では非線形計画問題の定式化，メタヒューリスティック法による近似解の導出，マルコフ連鎖による解導出を検討しており，理論的なアプローチを活用する点が学術的にも大きな価値がある．

2 関連研究

2-1 データセンタに対する仮想マシン割り当て

通信ネットワーク経由で様々なサービスを提供するクラウドサービスの普及により，データセンタの重要性は年々高まっており，データセンタの大規模化および複数拠点化が進んでいる．このようなデータセンタで大量のジョブを効率よく処理するためには，仮想マシンの配置が重要となり，それゆえ，仮想マシンの配置に対する様々な方式が提案されている．

サーバ資源およびリンク資源の有効利用を実現する仮想マシンの最適配置に関しては，通信トラヒック量

を考慮した仮想マシンの最適配置法や CPU・メモリ・伝送帯域の使用量を最小にする最適配置法などが提案されている．ネットワークの輻輳回避を実現する仮想マシンの最適配置に関しては，サーバ間の接続形態を考慮する方式が検討されている．さらに，資源の有効利用とネットワークの輻輳を同時に考慮する仮想マシン配置法も提案されている．また，データセンタの省電力化を前提とした仮想マシン配置方式も検討されている．

一方，データセンタ間のライブマイグレーションに関しては，実現に向けて数多くの実証実験が行われている．[1]では，広域に分散されたデータセンタ間でライブマイグレーションを実現しており，他の研究では，複数機関にまたがる長距離のライブマイグレーションの実証実験を行っている． また，Mobile IPv6 トンネリング機構を利用した実験行われている．

ライブマイグレーションを考慮した仮想マシン配置に関しては，仮想マシンの最適配置法が検討されている．この方式によって，ライブマイグレーションを実行した際の性能劣化を最小限に抑えることができる．その一方で，前述のデータセンタ間ライブマイグレーションに関しては考慮していない．データセンタ間のライブマイグレーションでは，データセンタ間の距離や伝送帯域，ジョブ量に応じて伝送遅延が大きくなってしまい，処理性能が著しく劣化してしまう．そのため，データセンタ間ライブマイグレーションを考慮した仮想マシンの最適配置が必要不可欠となる．しかしながら，データセンタ間ライブマイグレーションを考慮した非線形計画問題では，対象となるサーバ数やリンク数が膨大となり，最適解の導出が容易でない場合も多い．そこで，非線形計画問題の解の高速な導出が必要となる．

非線形計画問題の解を導出するには，一般に，焼きなまし法や遺伝的アルゴリズムなどの発見的手法が使用される．しかしながら，非線形計画問題の種類によってはマルコフ連鎖によって近似ができ，マルコフ連鎖の解を導出することで最適解を高速に導出できる．したがって，データセンタ間ライブマイグレーションを考慮した非線計画問題も，条件を満足することで高速な解導出が可能になる．

2-2 遺伝的アルゴリズムによる最適解導出

最適化問題の解を導出するために用いられるメタヒューリスティック法の一つとして，遺伝的アルゴリズムが提案されている[6,7]．この遺伝的アルゴリズムは生物界の進化の仕組みを模したアルゴリズムであり，あらゆる最適化問題に適用することが可能である．本アルゴリズムでは，最適化問題の解を遺伝子として表現し，環境に適応した遺伝子を持つ個体が生き残る確率が高くなるような世代交代を行って遺伝子を進化させる．最終的には，世代交代によって残った最適な遺伝子を導出することで最適解が導出される（図 2 参照）．

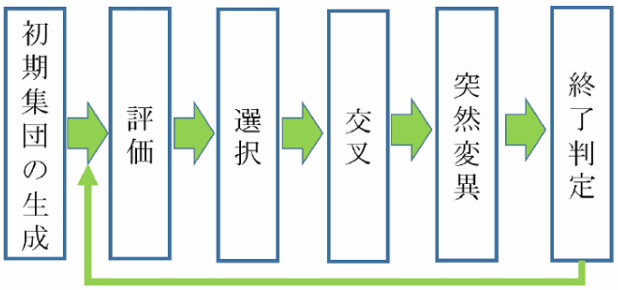

図 2 遺伝的アルゴリズム

基本的な手順は以下のとおりである．

1. 初期集団の生成
2. 各個体の適応度を計算
3. 個体の淘汰・選択
4. 遺伝子の交叉の実行
5. 突然変異の実行
6. 終了条件の判定

遺伝的アルゴリズムでは，まず初期集団として解の候補を遺伝子に持つ個体を複数生成する．手順 1 では，遺伝子（最適化問題の解）の初期集団をランダムに決定し，各遺伝子の適応度を最適化問題の目的関数によって導出する(手順 2)．それから，集団の中から次世代の親を選択する(手順 3)．このとき適応度が高い遺伝子ほど生き残る確率が高くなるように設定する． それから手順 4 で，交叉・突然変異の操作を行うことで新しい遺伝子を生成し，さらに手順 5 の突然変異によって，一定の確率(突然変異確率)で遺伝子を変化させる．それから，最大世代数などの終了条件が満たされた場合には処理を終了し，満たされていない場合には手順 2 に戻り続行する(手順 6)．

2-3 最適化問題のマルコフ近似による近似解導出

最適化問題の近似解を導出するために，マルコフ近似を用いた方式が提案されている[8,9]．本方式では，最適化問題の解を連続時間マルコフ連鎖の状態として表す．このマルコフ近似では，各状態に対する目的関数の値を基に状態遷移確率を設定することで，最適値（最小化問題の場合は最小値，最大化問題の場合は最大値）となる状態に遷移しやすくなる．これにより，システムが最適な状態に滞在する時間が最も長くなり，最適な動的システムの運用が可能となる．

最適化問題を Log-Sum-Exponential 関数を用いて近似してエントロピーの概念を導入することによって，最適化問題の最適解が連続時間マルコフ連鎖の定常解で近似することが可能となる．

VMPR-offline
Initialize a feasible system configuration f_0.
$f_{best} \leftarrow f_0, x_{best} \leftarrow x_{f_0}$
$t \leftarrow 0$
for $t = 0$ to T
  Randomly pick a VM i
  // Migrate VM i to a new configuration
  $g_i \leftarrow$ VM i's configuration
  Remove VM i from the system, $f_t \leftarrow f_t \backslash g_i$
  $\mathscr{G}_i \leftarrow \mathbf{GreedyAdd}(G, f_t, i)$
  $\mathscr{F}_t \leftarrow \{f_t\} \times \mathscr{G}_i$
  Probabilistically choose a configuration $f \in \mathscr{F}_t$ acc. to (8)
  Migrate VM i according to f
  if $x_f < x_{best}$
    $f_{best} \leftarrow f, x_{best} \leftarrow x_f$
  $f_{t+1} \leftarrow f$
end for

図 3 仮想マシン配置アルゴリズム –オフライン–

GreedyAdd(G, f_{now}, u)
// Find Δ feasible configuration by migrating VM i given f_{now}
  Initialize $\mathscr{S} = \{S_1, \ldots S_M\}$, where $V(S_i) = \{v_i\}$ and $E(S_i) = \emptyset$
  $\mathscr{G} \leftarrow$ set of feasible configurations for VM i
  $E_G \leftarrow E(G)$
  while $|\mathscr{G}| < \Delta$
    $(v_i, v_j) \leftarrow \arg\min_{l \in E_G} g'(r_l / C_l)$
    $E_G \leftarrow E_G \backslash \{(v_i, v_j)\}$
    Let $v_i \in V(S_{t_1}), v_j \in V(S_{t_2})$
    if $t_1 \neq t_2$ *// Merge two subgraphs S_{t_1}, S_{t_2}*
      $V(S_{t_1}) \leftarrow V(S_{t_1}) + V(S_{t_2}), E(S_{t_1}) \leftarrow E(S_{t_1}) + E(S_{t_2}) + (v_i, v_j)$
    else *// $t_1 = t_2$*
      $E(S_{t_1}) \leftarrow E(S_{t_1}) + (v_i, v_j)$
    $\mathscr{G} \leftarrow \mathscr{G} +$ set of new feasible configurations for VM u in $\mathscr{S}$
  end while
  return $\mathscr{G}$

図 4 仮想マシン配置アルゴリズム –オンライン–

前節で紹介したマルコフ近似を用いた最適化問題の解導出法を利用することで，データセンタの仮想マシンを管理する技術が提案されている[10]．本技術では，すでに設定されている仮想マシンの配置を，定期的にマイグレーションすることで最適な位置に移動させることを実現する．

2-4 マルコフ近似を用いたデータセンタの仮想マシン管理技術

図 4 は，本方式で用いるアルゴリズムを示している．このアルゴリズムでは，1 台の仮想マシンをマイグレーションする物理マシンを選択する．なお，複数の仮想マシンに対するマイグレーションは対象外となっている．本アルゴリズムでは特に，マルコフ連鎖の状態遷移確率を基にマイグレーションを行うことで，最適化問題の近似解を実現する仮想マシンの配置が実現される．

さらに，新しい仮想マシンの到着時と使用済みの仮想マシンの離脱時に対する仮想マシンの配置アルゴリズムが提案されている．このアルゴリズムでは，新しい仮想マシンの到着時にのみマイグレーションを実施する．

これらの方式を用いることで，最適化問題の解を膨大な時間をかけて導出する必要がなく，実環境での利用が期待できる．

3 提案方式

本章では，データセンタ間のマイグレーション遅延を考慮した最適仮想マシン配置法を提案する．以下では，仮想マシンの配置決定問題を最適化問題で定式化し，遺伝的アルゴリズムによって解を導出する．

3-1 概要

以下では，N 個のデータセンタが接続された通信ネットワークを考える．このネットワークにおいて，n（n=1，．．．，N）番目のデータセンタ内に M_n 台のサーバがあり，さらに M_n 台のサーバが L_n 本のリンクで接続されているとする．また，n 番目のデータセンタは R_n 個のデータセンタと隣接しているとする．

N 個のデータセンタでは，合計 K 個のジョブを処理する必要があり，k（k=1，．．．，K）番目のジョブをデータセンタ n から n' へマイグレーションする際のデータ量を $E^k_{nn'}$ とする．このとき，データセンタ n とデータセンタ n' 間の伝送遅延を Dnn' とする．

また，$g(r^l_n/C^l_n)$ をデータセンタ n 内のリンク l に対する利用率とし，さらにデータセンタ n 内のサーバ m が使用中であれば Y^m_n=1\$とし，未使用であれば Y^m_n=0 とする．

このとき，以下の最適化問題に従って，K 個のジョブを各データセンタに配置する．ここで，α と β は重み変数である．

$$\min \sum_{n=1}^{N} \frac{1}{L_n} \sum_{l} g\left(\frac{r_n^l}{C_n^l}\right) + \alpha \sum_{n=1}^{N} \frac{1}{M_n} \sum_{m} Y_n^m + \beta \frac{1}{K} \sum_{k} \sum_{n=1}^{N} \frac{1}{R_n} \sum_{n'=1}^{N} D_{nn'} E_{nn'}^k \quad (1)$$

また，サーバ m の物理資源量をベクトル H^m，ジョブ K が要求する仮想マシンの数を w_k，資源量のベクトルを $S_{k,i}$ としたときの制約条件は以下のとおりである．

$$\sum_{n=1}^{N} \sum_{k} \sum_{i=1}^{w_k} Z_{k,i}^{n,m} S_{k,i}^n \leq Y_n^m H^m, \forall m \quad (2)$$

このとき $Z^{n,m}_{k,i}$ はバイナリ変数であり，サーバ m にジョブ k の仮想マシン i が配置されるかどうかの二分決定で用いられる．具体的には，ジョブ k の仮想マシン i がサーバ m に配置されているときに 1 の値をとり，それ以外の場合に 0 となる．

ここで，式(1)の第 1 項は各データセンタのリンク利用率の総和を表しており，第 2 項は各データセンタ内のサーバ稼働率の総和を示している．また，第 3 項は，データセンタ間のマイグレーション時の平均遅延を示す．ここで，α と β は重み変数であり，値が大きい方の項がより重視されて仮想マシン音配置が決定される．この最適化問題の解は次章に示す遺伝的アルゴリズムによって導出し，仮想マシンの最適配置を決定する．

3-2. 提案方式における遺伝的アルゴリズム

本研究では，1 つの個体の遺伝子を用いて，各ジョブに対してどのデータセンタ内のどのサーバを割り当てるかを表す．

図 5 は，遺伝子を用いた仮想マシンの配置例を示している．このようにジョブが要求するサーバ数に応じて配置するサーバの番号を遺伝子として与える．

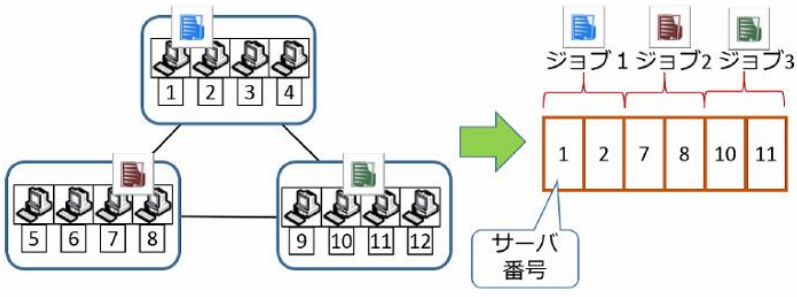

図 5 提案方式で用いる遺伝子の概要

このサーバ番号は全データセンタにあるサーバすべてを通し番号で表している．図 6 は，仮想マシンの配置に応じた遺伝子の表示例を示している．この例では，遺伝子 1 ではジョブ 1 に対してデータセンタ 1 のサーバを割り当て，ジョブ 2 に対してはデータセンタ 2 のサーバを割り当てている．

これらの例に従って,提案方式では，最適化問題の解を導出するために，全てのジョブに対する配置を並べたものを遺伝子型として定義する．

通常,初期集団の各遺伝子の値はランダムに決定するが，本研究では制約条件を満たすように遺伝子をランダムに生成し初期集団とする．

それから，式(1)の最適化問題の式を用いて，各遺伝子の適応度を計算する.式(1)は最小化問題であるため，その解の値が小さいほど適応度が高くなるように適応度は目的関数の値の逆数とする．

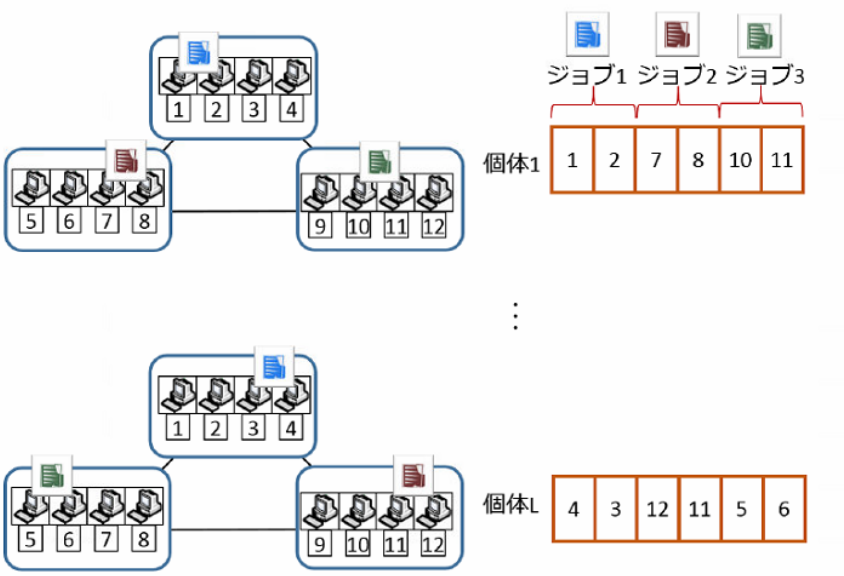

図 6 遺伝子の例

各遺伝子の適応度を計算した後，エリート選択を実施する．エリート選択では，各世代における適応度が最も高い遺伝子が選ばれる．この選択によって，目的関数の最小化に適した遺伝子を残すことが可能となる．その一方で，最小化問題に対して局所最適に陥りやすくなる可能性が高くなるという問題が生じる．さらに，適応度に従った確率によって遺伝子を決定するルーレット選択も実施する．ルーレット選択では，適応度 f_i の遺伝子 i を以下の確率 p_i で選択される．

$$p_i = \frac{f_i}{\sum_{i=1}^{N} f_i} \qquad (3)$$

本研究では，これら 2 つの選択方式を併用し，エリート選択により最適な個体を 1 つ選択した後にルーレット選択によって残りの個体を選択する．これにより，現時点で最良な個体が淘汰されてしまうことを防ぎつつ，局所最適解に陥らないように遺伝子を進化させるようにしている．

さらに，ルーレット選択で選んだ遺伝子に対して一点交叉を行う．一点交叉では，2 つの遺伝子で遺伝子情報を交換する基準点をランダムに 1 箇所選び，その基準点より後方部分を選択された個体間で遺伝子情報を入れ替える．

本研究では各ジョブに割り当てる仮想マシンはすべて同一データセンタ内でなければならないため，交叉する点は各ジョブの区切りでランダムに選ばれるものとする．

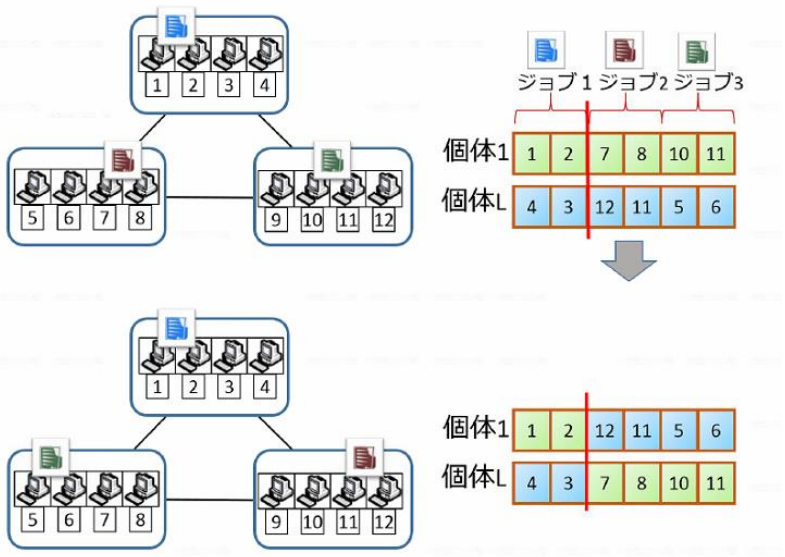

図 7 一点交差

さらに，一点交叉を処理した後に，生成された遺伝子に対して突然変異の処理を実行する．突然変異の処理は，生物における遺伝子の突然変異をモデル化したものである．具体的には，あらかじめ設定した突然変異率に従って，各遺伝子の一部を他の値に変化させる．

この突然変異の処理によって，最適化問題の解が局所最適に陥るのを防ぐことが可能になる．

ここで，突然変異率を小さく設定すると局所最適に陥りやすくなる．一方で，突然変異率を大きく設定す

ると，解の探索効率が落ちてしまい解の導出に長時間を要する．

最後に，遺伝的アルゴリズムでは，あらかじめ最大世代数を設定する．そして，遺伝子の世代数が最大世代数に達した時点で，本アルゴリズムを終了する．

4. 数値例

本章では，提案方式に従って仮想マシンの配置を行った場合の，性能を評価する．

4-1 シミュレーション条件

以下では，図 8 に示したシミュレーション環境で提案方式の性能を評価する．図 8 に示すように，3 つのデータセンタがネットワークで接続されており，各データセンタ間の伝送遅延は，5，10，20 と設定されている．また，各データセンタでは，16 台のサーバが 3 階層の Fat tree 構造で接続されている．

ここで，各ジョブは 4 台の仮想マシンを使用して実行されるとし，データセンタに計 25 個のジョブを配置する．すなわち，計 100 個の仮想マシンを配置する．

なお，各サーバの資源量は 20 とし，各仮想マシンで使用される平均資源使用量を 0.5 から 5.0 までの間でランダムに決定される．

このとき，データセンタ間のマイグレーションを考慮した提案方式の性能を評価する．ここで，提案方式で使用する遺伝的アルゴリズムにおいて，交叉率を 0.95，突然変異率を 0.10 とした．また，1 世代ごとに生成される遺伝子数を 20 とし，終了条件となる最大世代数を 300 とする．さらに本提案方式の性能を，マイグレーション遅延を考慮しない従来の配置法，およびランダム配置法の性能を比較する．なお，以下の数値例において，グラフ内の赤い実践は提案方式，青い破線は従来法，緑の点線はランダム法の結果を示す．

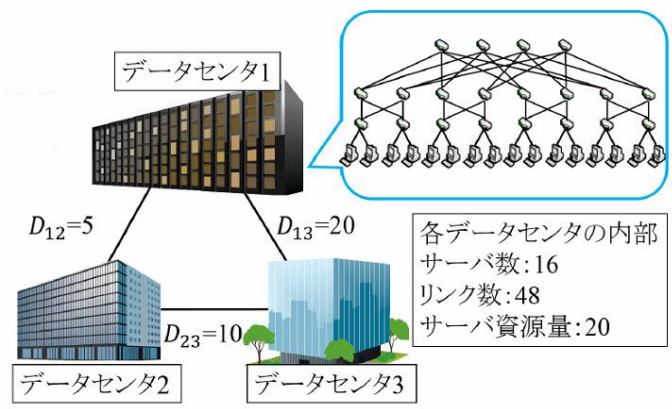

図 8 シミュレーション環境

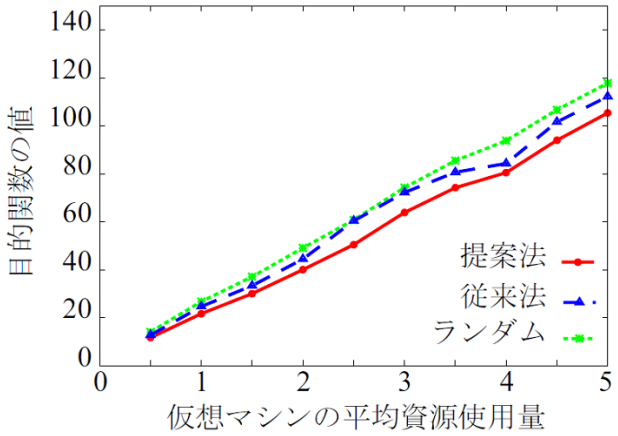

図 9 仮想マシンの平均資源使用量が目的関数の値に与える影響（β=0.5）

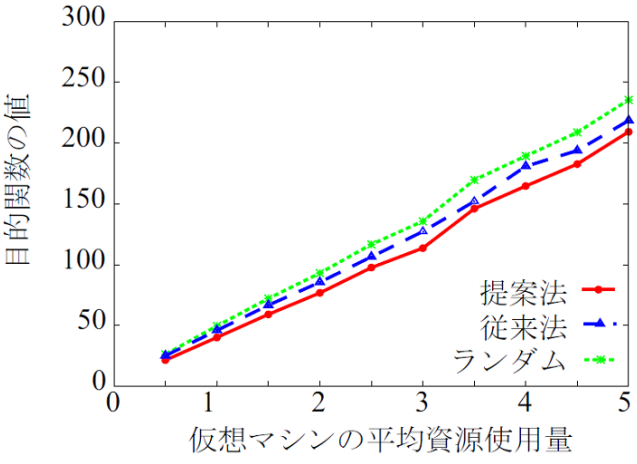

図 10 仮想マシンの平均資源使用量が目的関数の値に与える影響（β=1.0）

4-2 シミュレーション結果

最初に，仮想マシンの平均資源使用量が，式(1)の目的関数の値に与える影響を評価する．以下では，α=1.0 とし，β の値を 0.5，1.0，1.5 に変化させる．ここで，図 9 は β=0.5，図 10 は β=1.0，図 11 は β=1.5，の結果を示している．これらの図から，まず，仮想マシンの平均資源使用量が増加するにつれて，どの方式においても目的関数の値が増加することがわかる．これは，仮想マシンの設定に必要なサーバ数が増加し，さらにはマイグレーションされるデータ量が増加するためである．

仮想マシンの平均資源使用量が大きくなると，必要とするサーバの数自体が多くなることや，マイグレーションされるデータ量が大きくなることから目的関数の解の値は大きくなる．

ここで，各図において 3 つの方式を比較すると，仮想マシンの平均資源使用量によらず提案方式が最も小さな目的関数の値をとることが分かる．一方で，ランダム方式が最も大きな値となっている．このことから，提案方式を用いることで，仮想マシンの平均資源使用量によらず，常に提案方式が有効であることが分かる．

また，図 9，図 10，図 11 を比較すると，図 11 の場合に提案方式が他の方式よりも著しく有効になることが分かる． これは，β の値が大きい場合には，データセンタ間の伝送遅延を重要して，提案方式を用いる効果が大きくなるためである．それゆえ，提案方式は，データセンタ間の遅延が大きい場合に特に効果的であることが分かる．

さらに，従来法とランダム法の効果に関しては，β や平均資源使用量によらず，常に提案方式よりも性能が悪いことが分かる．

また，β の値を 0.1 と 100.0 の両極端に変化させた場合の結果を評価する．図 12 は，β=0.1 の場合に，仮想マシンの平均資源使用量が目的関数の値に与える影響を示している．また図 13 は，β=100.0 の場合に，仮想マシンの平均資源使用量が目的関数の値に与える影響を示している．

これらの結果から，β の値が小さくても大きくても，仮想マシンの資源使用量が増加するにつれて，目的関数の値が増加することが分かる．また，3 つの方式の性能に関しては，提案方式が最も目的関数の値が小さく，ランダム法が最も大きな値となる．このように，図 12 および図 13 の結果は，図 9～図 11 と同じ傾向を示していることが分かる．

さらに，各方式の性能差については，β=100 の場合が最も大きくなることが示されている．それゆえ，提案方式は，データセンタ間の伝送遅延が重要となる場合に特に有効であることが分かる．さらには，従来法とランダム法の性能に関しても，β の値によらずほぼ同じ傾向になることが分かる．

次に，3 つの方式の性能差を目的関数の値の減少率で比較する．ここで，仮想マシンの平均資源使用量が x の時に，提案法に対する目的関数の値を $F_1(x)$，従来法に対する目的関数の値を $F_2(x)$，ランダム法に対する目的関数の値を $F_3(x)$ とすると，「従来法のランダム法に対する減少率」は $(F_3(x)-F_2(x))/F_3(x)$ で計算される．また，「提案法のランダム法に対する減少率」は $(F_3(x)-F_1(x))/F_3(x)$ となり，「提案法の従来法に対する減少率」は $(F_2(x)-F_1(x))/F_2(x)$ となる．
それゆえ，この減少率が 0 より大きい場合は，前者の方式を用いることで目的関数の値が減少し，適切な仮想マシンの配置ができていることを示す．

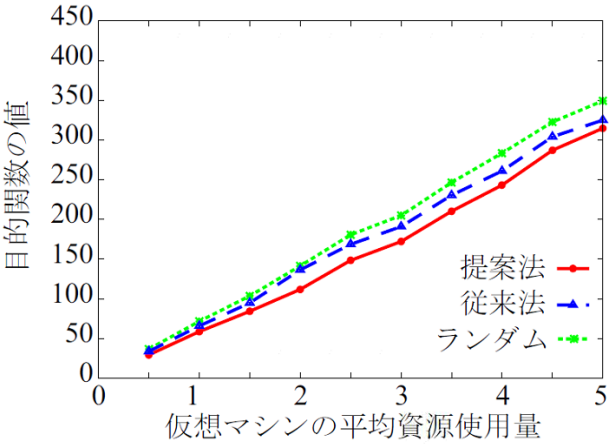

図 11 仮想マシンの平均資源使用量が目的関数の値に与える影響（β=1.5）

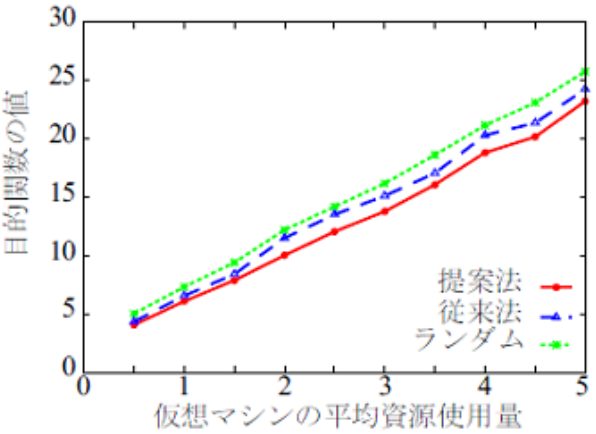

図 12 仮想マシンの平均資源使用量が目的関数の値に与える影響（β=0.1）

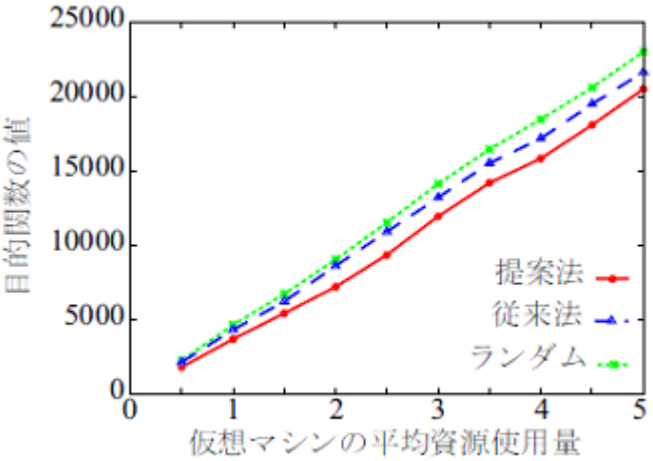

図 13 仮想マシンの平均資源使用量が目的関数の値に与える影響（β=100.0）

また，この値が大きいほど，後者の方式に比べて前者の方式が有効であることを示す．

図 14 は，β=0.1 の場合に，目的関数の減少率が仮想マシンの平均資源使用量に応じてどのように変化するかを示している．また図 15 および図 16 はそれぞれ，β=1.0 と 100.0 の場合に，目的関数の減少率が仮想マシンの平均資源使用量に応じてどのように変化するかを示している．

図 14 から，β=0.1 の場合は，仮想マシンの平均資源使用量が増加するにつれて各方式間の減少率が小さくなることが分かる．これは，平均資源使用量が増加するにつれて，各方式の性能差が小さくなることを示している．また，提案法は従来法とランダム法よりも常に有効であることが分かる．また，データセンタ間のマイグレーション遅延を考慮しない従来法も，ランダム法よりは有効であることがわかる．

次に図 15 から，β=1.0 の場合も，β=0.1 の場合と比較してほぼ同じ傾向であることが分かる．両者を比較すると，β=1.0 の場合のほうが，減少率がやや大きいことに注意する．さらに，図 16 は，β=100.0 の場合を示している．β=100.0 とした場合には，図 14 や図 15 よりも減少率が大きくなっている．これは，β の値を変更することでデータセンタ間のマイグレーション遅延に対する重要度が増し，提案方式の効果がより大きくなるためである．

これらの結果から，提案方式を用いることによって，仮想マシンの平均資源使用量によらず目的関数の値は減少することがわかる．特に，データセンタ間のマイグレーション遅延が大きい場合には，提案方式を使用することが望まれる．

最後に，提案方式を用いた場合に，遺伝的アルゴリズムによる遺伝子の最適な適応度が，どのように変化するかについて調査する．

図 17 は，仮想マシンの平均資源使用量が 2.0 の場合に，提案方式の遺伝的アルゴリズムに対する適応度が世代数に応じてどのように変化するかを示している．さらに，図 18 は，仮想マシンの平均資源使用量が 3.0 の場合に，提案方式の適応度がどのように変化するかを示している．

図 17 から，平均資源使用量が 2.0 の場合には，世代数が増えるにつれて目的関数の値が減少して，適応度が増加していることが分かる．特に，世代数が 40 以上になると目的関数の値および適応度が変化しないことが分かる．それゆえ，本提案方式を用いることで，最適化問題の最適解に収束し，最適な仮想マシンの配置が実現できていることが分かる．

また図 18 からは，平均資源使用量が 3.0 の場合

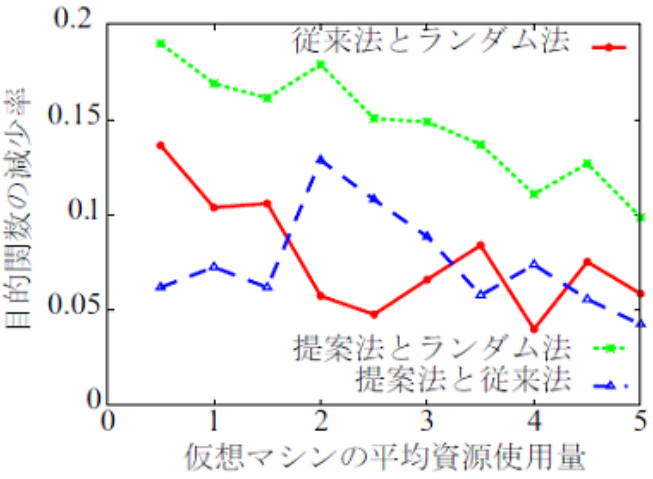

図 14 目的関数の減少率（β=0.1）

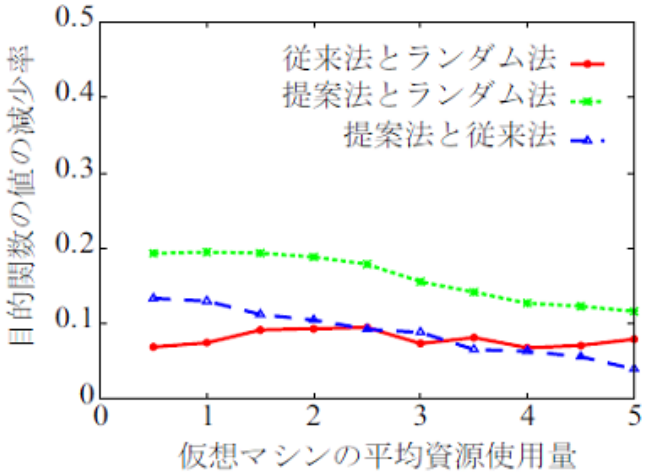

図 15 目的関数の減少率（β=1.0）

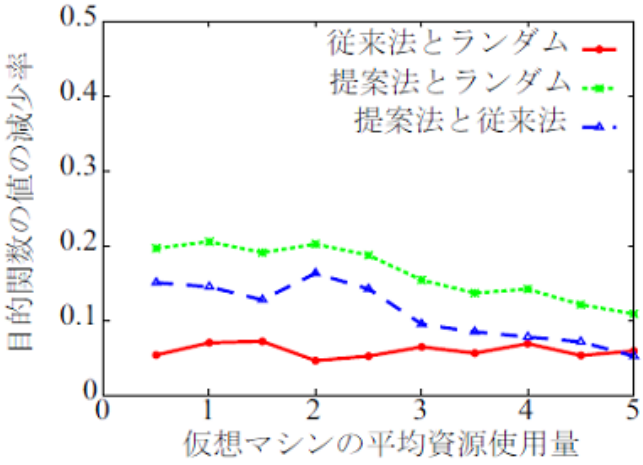

図 16 目的関数の減少率（β=100.0）

でも，世代数が増えるにつれて，目的関数の値が減少し，適応度が増加していることがわかる．図 17 と比較すると，目的関数の値が約 120 世代まで変化していることがわかる．図 18 では，仮想マシンの資源使用量が図 17 よりも大きく，仮想マシンの配置による影響が大きいためである．しかしながら，これらの結果から，提案方式を用いることで，数百世代で最適な仮想マシンの配置を決定できることが分かる．

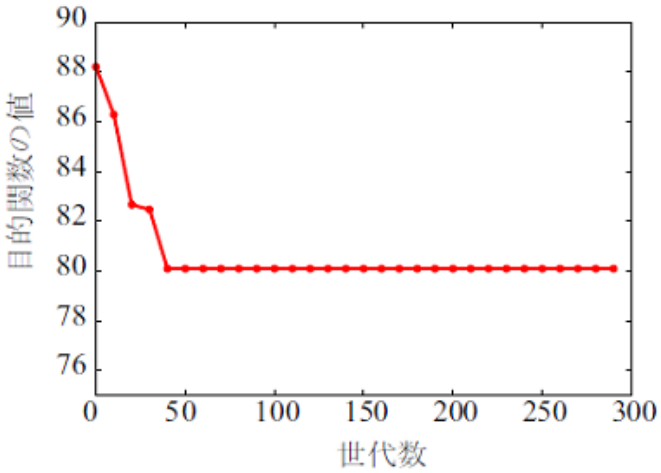

図 17 遺伝的アルゴリズムによる目的関数の値の変化(平均資源使用量 2.0)

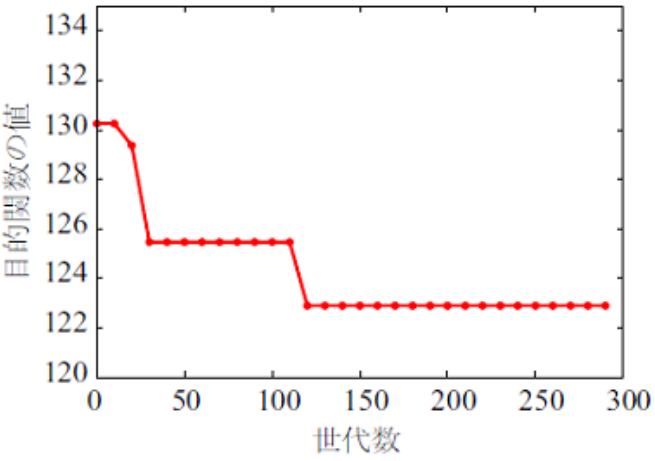

図 18 遺伝的アルゴリズムによる目的関数の値の変化(平均資源使用量 3.0)

5. マルコフ近似の適用

本章では，3 節で述べた最適化問題に対する解を，マルコフ近似によって導出する方法について検討する．

ここで，前述の最適化問題では，仮想マシンの新規割り当ておよび離脱を検討していない．そこで以下では，新しい仮想マシンの割り当てとし使用済の仮想マシンの離脱を考える．また，定期的に仮想マシンのマイグレーションを実施する．このようなシステムモデルの変更によって，最適化問題の解に従って仮想マシンの割り当てを，マルコフ近似によって迅速に実行することができる．

ここで，新しい仮想マシンの割り当て要求が到着すると，現在の状態 x に対する式(1)の目的関数の値 $F(x)$ を計算する．それから，新しい仮想マシンを割り当てた場合に遷移可能な状態を調査し，任意の状態 y に対する目的関数の値を $F(y)$ とする．このとき，$F(x)$ と $F(y)$ から導出された確率に従って，仮想マシンの割り当てを決定する．すなわち，次の状態 y を決定する．

次に，使用済の仮想マシンを削除する場合を考える． 仮想マシンが離脱する場合，仮想マシンのマイグレーションを実施する．ここで，仮想マシンのマイグレーションに関しては，新しい仮想マシンを割り当てる場合と同じく，現在の状態 x に対する式(1)の目的関数の値 $F(x)$ を計算する．それから，マイグレーションを行った場合に遷移可能な状態を調査し，任意の状態 y に対する目的関数の値を $F(y)$ とする．このとき，$F(x)$ と $F(y)$ から導出された確率に従って，仮想マシンのマイグレーションを決定する．

このように，目的関数から導出した遷移確率に応じて，新規の仮想マシンを配置し，さらにデータセンタ間のマイグレーションを実行することで，最適化問題(1)に応じた適切な仮想マシンの配置が実現できる．

さらに，仮想マシンのマイグレーションを定期的に実行するために，指数分布に従うマイグレーション実行タイマーを導入する．本タイマーが指数分布に従ってマイグレーションの実行タイミングを決定することによって，マルコフ近似によって引き続き解を導出することができる．このタイマーの実行タイミングを調整することで，適切な仮想マシンの配置を継続することが可能となる．

6. まとめと今後の予定

本研究では，データセンタ間のマイグレーション遅延の影響を最小にする仮想マシンの配置法を提案した．提案法では仮想マシンの配置を最適化問題として定式化し，遺伝的アルゴリズムによって解を導出した．

数値例において，提案方式を用いることで，データセンタ間のマイグレーション遅延を考慮した仮想マシンの配置が実現できることを示した．また，β の値に関わらず提案法が有効であることから，データセンタ内の負荷も考慮できていることを示した．

さらに，マルコフ近似を用いた最適解の導出法についても検討した．検討したアルゴリズムを利用することによって，最適化問題の最適解に近い状態に長時間滞在できるような仮想マシンの割り当ておよびマイグレーションが実現できる．このマルコフ近似による解導出は，データセンタを運用しながら実行することができるため，実環境での利用が容易となることが期待できる．このマルコフ近似を用いた仮想マシンの割り当てに関しては，まだ性能評価を行うことができていない．しかしながら，最適化問題の有効性は遺伝的アルゴリズムを用いた実験によって既に示されているため，このマルコフ近似を利用したアルゴリズムを用いることによってデータセンタ間のマイグレーション遅延を考慮した適切な仮想マシンの割り当てが期待できる．

今後は，マルコフ近似を用いた方式の性能をシミュレーションによって評価し，階の計算時間が大幅に短縮できることを示す．さらには，遺伝的アルゴリズムを用いた場合とマルコフ近似を用いた場合の 2 つの方式を状況に応じて使い分けることを考える．具体的には，仮想マシンの割り当て要求の到着が多い場合は，マルコフ近似によって仮想マシンの配置を決定し，到着が少ない場合は遺伝的アルゴリズムによって仮想マシンの配置を決定する．

【参考文献】

[1] 櫛山, 羽田, 太田, 波方, 浦賀, 日浦, ``広域分散化されたデータセンタ間でのライブマイグレーション実証実験," 信学技報, IN2011-39, JUN. 2011.

[2] 永渕, 岸, 井上, 小山, 北爪, ``国際通信網における仮想マシンのマイグレーション性能評価," 信学技報, IN2012-40, JUN. 2012.

[3] 小川, 得永, 響庭, 飯田, 北山, 松本, 堀坂, ``仮想マシン群の効率的な配置に関する一検討," 信学技報, IN2009-30, NOV.~2009.

[4] 永渕, 寺本, 小山, 北爪, ``データセンタ間ライブマイグレーション環境における冗長経路回避に向けた経路制御方式の提案," 信学技報, IN2013-48, JUN. 2013.

[5] 市原, 小泉, 大崎, 波戸, 村山, 今瀬, ``クラウド環境におけるライブマイグレーションとトラヒックエンジニアリングの統合制御の有効性評価," 信学技報, IN2011-172, DEC. 2012.

[6] 北野, 遺伝的アルゴリズム, 産業図書, 1993.

[7] D.E. GOLDBERG, ``GENETIC ALGORITHMS IN SEARCH, OPTIMIZATION AND MACHINE LEARNING," ADDISON-WASLEY PUBLISHING COMPANY, 1989.

[8] M. CHEN, S.C. LIEW, Z. SHAO, AND C. KAI, ``MARKOV APPROXIMATION FOR COMBINATORIAL NETWORK OPTIMIZATION," IN PROC. IEEE INFOCOM 2010, MAR. 2010.

[9] L. JIANG, AND J. WALRAND, ``A DISTRIBUTED CSMA ALGORITHM FOR THROUGHPUT AND UTILITY MAXIMIZATION IN WIRELESS NETWORKS," IEEE/ACM TRANS. ON NETWORKING, JUN. 2010.

[10] J.W. JIANG, T. LAN, S. HA, M. CHEN AND M. CHIANG, ``JOINT VM PLACEMENT AND ROUTING FOR DATA CENTER TRAFFIC ENGINEERING," IN PROC. IEEE INFOCOM 2012, MAR. 2012.