情報処理学会論文誌　コンピューティングシステム　Vol.5　No.5　33–42 (Oct. 2012)

省電力化のためのマッチングに基づく仮想計算機パッキングアルゴリズム

高橋 里司[1,3,a)]　竹房 あつ子[2]　繁野 麻衣子[1]　中田 秀基[2]　工藤 知宏[2]　吉瀬 章子[1]

受付日 2012年3月28日, 採録日 2012年6月6日

概要：データセンタでの消費電力低減の手法として，低負荷の仮想計算機（VM）を高速にマイグレーションして集約し，使用していない物理計算機を省消費電力モードで運用する方法がある．これを実現するには，消費電力を抑えつつユーザ体験の劣化を防ぐ，負荷が変動する VM 群を現実的な時間で再配置する，再配置時のマイグレーションによる通信衝突を考慮する，という課題がある．本研究では，ユーザ体験を劣化させることなく省電力化を図る仮想計算機パッキング問題に対するマッチングベースアルゴリズムを提案，評価する．提案手法では，再配置時に 1 つの物理計算機が送受信できる VM 数を制限し，VM を送受信する物理計算機の組合せをグラフのマッチングで表すことで，多項式時間で適切な再配置プランニングを可能にする．評価実験では，マッチングを用いた提案手法，仮想計算機パッキング問題を 0-1 整数計画問題として定式化して最適化ソルバを用いる手法とグリーディな手法に対して，消費電力削減効果，計算時間，ユーザ体験の劣化を比較する．この評価実験から，1) パッキングをしない場合より 18%から 50%の消費電力が削減できる，2) マッチングベースアルゴリズムにより，現実的な計算時間で適切な再配置を行うことができる，3) ユーザ体験は消費電力の削減効果に反比例する傾向があるが，再配置によりほぼ改善できる，ことが示される．

キーワード：仮想計算機パッキングシステム，省電力化，マッチングアルゴリズム

Matching-based Virtual Machine Packing Algorithm for Lower Power Consumption

Satoshi Takahashi[1,3,a)]　Atsuko Takefusa[2]　Maiko Shigeno[1]　Hidemoto Nakada[2]
Tomohiro Kudoh[2]　Akiko Yoshise[1]

Received: March 28, 2012, Accepted: June 6, 2012

Abstract: VM (Virtual Machine)-based flexible capacity management, which consolidates multiple VMs into few physical machines and other physical machines are allowed to be put in "stand-by" mode, is an effective scheme to reduce total power consumption in a datacenter. However, there have been the following issues, reconcilement of power-saving and user experience, decision of VM packing in feasible calculation time and collision avoidance of VM migration processes. In order to resolve these issues, we propose a matching-based VM packing algorithm (MBA), which enables to decide a suitable VM packing plan in polynomial time. In the experiments, we compare the proposed algorithm with 0-1 integer programming model, using optimization solvers, and a greedy algorithm. The results show that 1) the proposed algorithm MBA reduces the total power consumption by between 18% and 50% of when the packing plan does not work, 2) MBA enable to provide an optimal packing plan in feasible calculation time, and 3) there is a trade-off between the power-saving effect and the deterioration of the user experience.

Keywords: virtual machine packing system, low power consumption, matching algorithm

1. はじめに

クラウドコンピューティングの普及により，データセンタにおける IT 機器の消費電力が今後急激に増大すること

[1] 筑波大学
University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[2] 産業技術総合研究所（AIST）
National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Ibaraki 305–8568, Japan
[3] 日本学術振興会特別研究員（DC2）
JSPS Research Fellow

[a)] stakahashi@sk.tsukuba.ac.jp

が予想されている．消費電力増大の原因の 1 つとして，リクエストに対する資源割当て時に必要容量を設定することの困難さがあげられる．データセンタ内で運用されるサービスの負荷やそのリクエストの頻度は，事前に予測ができないうえ，時間によって変動する．サービス要求に対する資源割当てが負荷に対して過小であると，サービス処理性能が低下してユーザ体験が劣化してしまう．よって，データセンタの設計者は資源利用のピーク時に合わせた資源割当てを行わなければならず，低負荷時には資源浪費が発生する．さらに，物理計算機は負荷が割り当てられていない場合も稼働中の消費電力は飽和時の 5 割程度にもなることがあり，それによる電力の浪費も非常に大きい．

この問題に対して，仮想計算機（VM）の高速ライブマイグレーション技術を用いて，ユーザ体験を劣化させることなく VM を再配置して消費電力を削減する手法が提案されている [1]．低負荷時には，1 台の物理計算機に対して複数の VM を立ち上げ，たとえば ACPI S3 のような省電力モードなどで不要な物理計算機の稼働を抑えることでデータセンタ内で消費される電力量を抑制する．負荷が増大した際には，物理計算機を復帰させて VM を再配置することでサービスの劣化を防止する．

一方で，VM の再配置手法に関して十分に議論されているとはいえない．具体的には，(1) 消費電力を抑えつつサービス劣化を防ぐこと，(2) 変動する VM の負荷に対する現実的な時間での VM の再配置，(3) 複数の VM の同時ライブマイグレーションにおける衝突回避，が技術的課題としてあげられる．(1) は，少数の物理計算機に複数の VM を稼働させて消費電力量を削減したとしても，各 VM に必要とする資源が割り当てられないとユーザ体験が劣化してしまうため，VM の負荷に応じて再配置をしなければならない．(2) は，各 VM の負荷は刻々と変化するため，VM と物理計算機の監視，再配置のプランニング，再配置の実行を繰り返し行うことが求められる．この一連の処理を現実的な時間内に実行しなければ負荷状況が変わってしまい，再配置の効果が得られない．(3) は，再配置の実行では高速ポストマイグレーション技術 [1] により 1 秒程度でユーザ体験を劣化させることなく VM を移動させることができる．しかし，マイグレーション後にバックグラウンドでメモリイメージのコピーを続けるため，[メモリサイズ/帯域] 時間，送受信する物理計算機間で通信が行われることになる．この間に通信衝突が起こると，完全にマイグレーションが終了するまでの時間が長くなってしまう．

本研究では，ユーザ体験を劣化させることなく省電力化を行うことを目的として，VM を再配置する問題に対するマッチングベースアルゴリズム（MBA）を提案，評価する．提案手法では，物理計算機の CPU 使用率とメモリ量を VM への割当て対象の資源とし，1 つの物理計算機が送受信する VM の数を制限して再配置を試みる．この VM の再配置をグラフのマッチングで表すことで，計算機数と，VM 数の多項式時間で適切な再配置プランニングを可能にする．また，比較対象として 0-1 整数計画問題として定式化して最適化ソルバを用いる手法（IP），グリーディな手法（GREEDY）を提案する [4]．評価実験では，各 VM の CPU 使用量とメモリ使用量をタイムステップごとに変動させ，各タイムステップにおける総消費電力量，プランニングに要する計算時間，ユーザ体験の劣化率を調査する．評価実験により，提案手法を用いることで，1) パッキングをしない場合より 18%から 50%の消費電力が削減できる，2) MBA により，現実的な計算時間で適切な再配置を行うことができる，3) ユーザ体験は消費電力の削減効果に反比例する傾向があるが，再配置によりほぼ改善できることを示す．

2. 関連研究

消費電力削減を目的とした VM の再配置手法は複数提案されている．

文献 [2], [3] では，仮想計算機を割り当てる物理計算機の台数とマイグレーション数の低減を目的として，仮想計算機割当て問題を 0-1 整数計画問題として定式化し，そのヒューリスティック解法として，遺伝的アルゴリズムおよびグリーディアルゴリズムを提案している．しかしながら，0-1 整数計画問題を分枝限定法で求解する手法や遺伝的アルゴリズムでは，求解までの計算時間が長く，実用化という面では課題がある．また，グリーディアルゴリズムでは，計算時間は短いが，マイグレーションの総数を考慮するのが困難であるという課題がある．

Entropy [14] は，本研究同様に制約によって仮想計算機パッキング問題を解いている．Entropy では，VM 群を資源要求によっていくつかのクラスに分類することで，大きく計算時間を短縮している．しかし，本研究が想定する環境では，VM の CPU 使用率は相互に独立に変動するものであり，VM 群が少数のクラスに分類可能であるという前提は受け入れがたい．

Stillwell ら [15] は，VM マイグレーションコストを考慮しつつ性能劣化を軽減するクラスタスケジューリングアルゴリズムを提案している．彼らの提案手法では，スタベーションを考慮した優先度付けを行い，従来のヒューリスティック手法に対してサスペンドやマイグレーションを適用し，高優先度ジョブを実行する．優先度を考慮する手法として興味深いが，消費電力を削減するものではない．

3. 仮想計算機パッキング

データセンタにおいて，各物理計算機上に複数の VM を配置して消費電力量を低減することを考える．個々の VM は物理計算機だけでなくデータセンタ内の様々な資源を利用するが，ここでは，物理計算機のメモリ資源と計算資源

（CPU 量）を考える．VM が使用するメモリ量は比較的一定で安定しているが，CPU 使用量は時間に対して大きく変動する．特に，VM として，Web サーバを考えると，長時間の低負荷状態の合間に瞬間的に高負荷となることが一般的である．

3.1 VM パッキングシステム

このような VM を低消費電力で多数運用するための方法として，Hirofuchi らにより VM パッキングシステムが提案されている [5]．パッキングシステムでは，各 VM の負荷状況を監視し，その結果を用いて再配置のプランニング，VM のマイグレーションを行う．低負荷時には，メモリ容量の許す限り多数の VM を物理計算機上に収納する．このとき使用していない物理計算機は ACPI S3 などの省消費電力モードで待機させる．ACPI S3 では，無負荷運用時 70 W 程度の消費電力を 5–7 W 程度の消費電力で待機させることができる．特定の VM の負荷が上昇した場合，省消費電力モードで待機させておいた物理計算機を復帰させ VM をライブマイグレーション技術によって移動させ，対応する．

高速ポストライブマイグレーション技術 [5], [6] を用いると，VM のマイグレーション所要時間は 1 秒程度となる．ACPI S3 からの復帰は 1–2 秒程度であり，総計で 2–3 秒程度で資源の再割当てを行うことができる．ただし，高速ポストライブマイグレーションでは VM をマイグレーションした後，バックグラウンドでメモリのコピーを続けるため，マイグレーションにおける通信衝突を考慮する必要がある．

3.2 仮想計算機パッキング問題

複数の物理計算機上に多数の VM が配置されている状態で，VM の負荷状況が変化したとき，消費電力が最小となるように，稼働する物理計算機を決定して VM の再配置を行う問題を，ここでは仮想計算機パッキング問題と呼ぶ．まず，記号の定義を与える．P を物理計算機の集合，V を VM の集合とする．各物理計算機 $i \in P$ にはメモリと CPU のキャパシティ m_i，c_i が与えられている．また，各 VM $k \in V$ のメモリ使用量と CPU 使用量を vm_k，vc_k とする．さらに，CPU 使用量 z のときの物理計算機 i（$\in P$）の電力使用量を $cost_i(z)$，物理計算機 i の基本電力使用量を $base_i$ とする．x^0_{ik} は VM k（$\in V$）の初期配置を表し，k が物理計算機 i（$\in P$）に配置されているときに 1，そうでないときに 0 の値をとる．決定変数は x_{ik}（$i \in P, k \in V$）と y_i（$i \in P$）の 2 種類の 0-1 変数で，それぞれ，

$$x_{ik} = \begin{cases} 1 & \text{（物理計算機 } i \text{ に VM } k \text{ を配置）} \\ 0 & \text{（そうでないとき）} \end{cases} \quad (1)$$

$$y_i = \begin{cases} 1 & \text{（物理計算機 } i \text{ を稼働）} \\ 0 & \text{（そうでないとき）} \end{cases} \quad (2)$$

を表す．このとき，仮想計算機パッキング問題（VMPP）は以下のように定式化される．

$$\text{最小化} \sum_{i \in P} \left(cost_i \left(\sum_{k \in V} vc_k x_{ik} \right) + base_i y_i \right) \quad (3)$$

$$\text{条件} \sum_{i \in P} x_{ik} = 1, \forall k \in V \quad (4)$$

$$\sum_{k \in V} vm_k x_{ik} \le m_i, \forall i \in P \quad (5)$$

$$\sum_{k \in V} vc_k x_{ik} \le c_i, \forall i \in P \quad (6)$$

$$\sum_{k \in V} x_{ik} \le |V| \cdot y_i, \forall i \in P \quad (7)$$

$$\sum_{k \in V} |x_{ik} - x^0_{ik}| \le 1, \forall i \in P \quad (8)$$

$$x_{ik} \in \{0, 1\}, \forall i \in P, \forall k \in V \quad (9)$$

$$y_i \in \{0, 1\}, \forall i \in P \quad (10)$$

目的関数 (3) は，CPU の使用率に応じて発生する電力量と，物理計算機を稼働しているときに発生する電力量の総和で表す．なぜならば，文献 [7] では，メモリの使用率の変動による使用電力量の変動と，物理計算機間の通信によるネットワーク機器の使用電力量の変動はほぼ見られないためである．制約 (4) は，各 VM は必ずいずれかの物理計算機に割り当てられなければならないことを表す．制約 (5) と (6) は，それぞれ割り当てられた仮想計算機のメモリと CPU の上限を表す．制約 (7) は，1 台以上 VM を保持している物理計算機は必ず稼働していなければならないという条件である．制約 (8) は各物理計算機に対するマイグレーションの回数制約を表す．ここで，通信衝突による性能劣化を防ぐため，各物理計算機での VM の送受信は 1 回とする．

4. 仮想計算機パッキング問題の解法

本研究では，仮想計算機パッキング問題の最適解を現実的な計算時間で求める，マッチングに基づく仮想計算機パッキングアルゴリズムを提案する．また，比較として計算時間の短縮を目的としたヒューリスティック手法であるグリーディアルゴリズムを提案する．

4.1 マッチングベースアルゴリズム

本研究で扱う仮想計算機パッキング問題では VM の送受信数の上限が 1 回であるため，マッチングを用いたアルゴリズムが構成できる．物理計算機の集合を頂点集合とする有向完全グラフを (P, A) とする．つまり，$A = \{(i, j) \mid i, j \in P, i \neq j\}$ である．各物理計算機 i に対して，i に初期配置されている VM の集合を V^i，すなわち $V^i = \{k \in V \mid x^0_{ik} = 1\}$ とする．物理計算機 i に初期配置されている VM k が物理計算機 j に再配置されたときの消費電力量の削減量 $\tilde{c}(i, j, k)$ は

$$
\begin{aligned}
&cost_i\left(\sum_{k'\in V^i} vc_{k'}\right) - cost_i\left(\sum_{k'\in V^i\setminus\{k\}} vc_{k'}\right)\\
&\quad + base_i\cdot\max\{0, 2-|V^i|\}\\
&\quad + cost_j\left(\sum_{k'\in V^j} vc_{k'}\right) - cost_j\left(\sum_{k'\in V^j\cup\{k\}} vc_{k'}\right)\\
&\quad + base_j\cdot\min\{0, |V^j|-1\} \qquad (11)
\end{aligned}
$$

で与えられる．物理計算機 i と j に対して，資源制約 (5)，(6) を満たすように i から j へ移動できる VM の集合 $V^{i\to j}$ を

$$
\left\{ k\in V^i \;\middle|\; \begin{array}{l} \sum_{\ell\in V^i\setminus\{k\}} vm_\ell \le m_j,\\ \sum_{\ell\in V^i\setminus\{k\}} vc_\ell \le c_j,\\ \sum_{\ell\in V^j\cup\{k\}} vm_\ell \le m_j,\\ \sum_{\ell\in V^j\cup\{k\}} vc_\ell \le c_j \end{array} \right\} \qquad (12)
$$

としたとき，各枝 $(i,j)\in A$ の重み $c(i,j)$ を

$$
\max\{\tilde{c}(i,j,k) \mid k\in V^{i\to j}\} \qquad (13)
$$

で与え，$c(i,j)=\tilde{c}(i,j,k)$ を達成する k を保持する関数を σ とする．つまり，$c(i,j)=\tilde{c}(i,j,\sigma(i,j))$ が成り立つ．また，$\tilde{A}=\{(i,j)\in A \mid V^{i\to j}\neq\emptyset\}$ とする．物理計算機間の VM の移動は，マイグレーションの回数がたかだか 1 回に制限されているので，グラフ (P,A) 上の端点を共有しない枝集合，すなわち，マッチングで表される．**図 1** は 7 台の物理計算機と VM が与えられているときのマイグレーションの様子を表している．物理計算機 2 から 6，4 から 7，および 5 から 3 へ VM を移動したとする．この再配置を物理計算機を頂点とする完全有向グラフで表すと，VM の移動方向を表す有向枝のマッチングに対応していることが分かる．

また，初期配置において，資源制約 (5)，(6) を満たしていない物理計算機の集合 $\{i\in P \mid \sum_{k\in V^i} vc_k > c_i\}\cup\{i\in P \mid \sum_{k\in V^i} vm_k > m_i\}$ を U すると，U の物理計算機からは必ず VM を移動させるので，資源制約を満たすような再配置は $(P,\tilde{A})$ 上の U を端点として含むマッチング（以下，U-マッチング）で表せる．したがって，U が空集合の場合

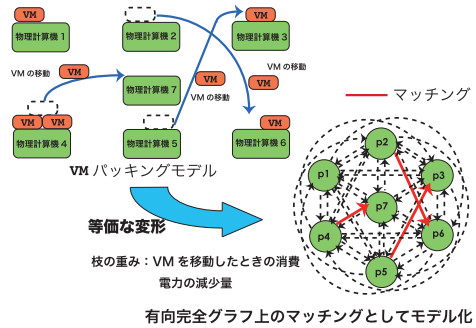


図 1　仮想計算機パッキング問題をマッチング問題に変換する例
Fig. 1　Example of transportation.

も含めて次の定理が成り立つ．

定理 1. グラフ $(P,\tilde{A})$ で重み c の和を最大にする U-マッチングを M^* とする．このとき，各 $(i,j)\in M^*$ に対して，i から j へ $\sigma(i,j)$ を移動させる再配置は仮想計算機パッキング問題の最適解となる．

Proof.　枝集合 $\tilde{A}$ と U の決め方より，得られた再配置が資源制約を満たすことは明らか．また，マッチングから得られているので，各物理計算機に対するマイグレーションの回数もたかだか 1 回である．

次に，仮想計算機パッキング問題の最適解から得られる各物理計算機 i への配置を $\hat{V}^i$ で表す．このとき，$\hat{M}=\{(i,j) \mid i,j\in P, V^i\cap\hat{V}^j\neq\emptyset\}$ とすると，各物理計算機間のマイグレーション回数がたかだか 1 回なので，$\hat{M}$ は $(P,\tilde{A})$ の U-マッチングになる．ここで，$P^+=\{i\in P \mid \hat{V}^i\setminus V^i\neq\emptyset\}$，$P^-=\{i\in P \mid V^i\setminus\hat{V}^i\neq\emptyset\}$ とし，各 $i\in P^+$ に対して，$\hat{V}^i\setminus V^i=\{k_i\}$，各 $i\in P^-$ に対して，$V^i\setminus\hat{V}^i=\{k_i\}$ とする．この性質を利用すると，初期配置と再配置との消費電力量の削減量は

$$
\begin{aligned}
&\sum_{i\in P}\left(cost_i\left(\sum_{k\in V^i} vc_k\right) + base_i\cdot\min\{1,|V^i|\}\right)\\
&\quad - \sum_{i\in P}\left(cost_i\left(\sum_{k\in \hat{V}^i} vc_k\right) + base_i\cdot\min\{1,|\hat{V}^i|\}\right)\\
&= \sum_{i\in P^+\cup P^-}\left(cost_i\left(\sum_{k\in V^i} vc_k\right) - cost_i\left(\sum_{k\in \hat{V}^i} vc_k\right)\right)\\
&\quad + \sum_{i\in P^-} base_i(\min\{1,|V^i|\} - \min\{1,|V^i|-1\})\\
&\quad + \sum_{i\in P^+} base_i(\min\{1,|V^i|\} - \min\{1,|V^i|+1\})\\
&= \sum_{i\in P^+}\left(cost_i\left(\sum_{k\in V^i} vc_k\right) - cost_i\left(\sum_{k\in V^i\cup\{k_i\}} vc_k\right)\right)\\
&\quad + \sum_{i\in P^-}\left(cost_i\left(\sum_{k\in V^i} vc_k\right) - cost_i\left(\sum_{k\in V^i\setminus\{k_i\}} vc_k\right)\right)\\
&\quad + \sum_{i\in P^-} base_i\cdot\max\{0,2-|V^i|\}\\
&\quad + \sum_{i\in P^+} base_i\cdot\min\{0,|V^i|-1\}\\
&= \sum_{(i,j)\in\hat{M}} \tilde{c}(i,j,k_i) \qquad (14)
\end{aligned}
$$

となる．明らかに $k_i\in V^{i\to j}$ なので，$\tilde{c}(i,j,k_i)\le c(i,j)$ となる．一方，M^* は最大重みの U-マッチングなので，$\sum_{(i,j)\in\hat{M}} c(i,j)\le\sum_{(i,j)\in M^*} c(i,j)$ であり，M^* から得られる再配置の方が消費電力の削減量が小さくなることはない．つまり，M^* から得られる再配置も最適な再配置である．□

マッチングに基づく仮想計算機パッキング問題に対する

厳密アルゴリズムを以下に示す．

マッチングベースアルゴリズム（**MBA**）

Step 1.　頂点を物理計算機とする有向グラフ $(P, \tilde{A})$ を構成し，各枝 $(i,j) \in \tilde{A}$ の重み $c(i,j)$ を求める．資源制約を満たしていない物理計算機の集合を $U = \{i \in P \mid \sum_{k \in V^i} vc_k > c_i\} \cup \{i \in P \mid \sum_{k \in V^i} vm_k > m_i\}$ とする．

Step 2.　$(P, \tilde{A})$ の，最大重みの U-マッチング M^* を求める．

Step 3.　各 $(i^*, j^*) \in M^*$ に対して，$\sigma(i^*, j^*)$ を i^* から j^* に再配置する．

Step 2 で最大重みの U-マッチングを求めるのには，$U = \emptyset$ のときの Edmonds の多項式時間アルゴリズム [8] を利用して．$O(|P|^3)$ 時間，または，$O(|P||\tilde{A}| + |P|^2 \log |P|)$ 時間で解くことができる [8], [9], [10].

定理 2. 仮想計算機パッキング問題は $O(|P|^3 + |V||P|)$ 時間で解ける．

Proof.　Step 1 で各枝 (i,j) の重み $c(i,j)$ を求めるのに $O(|V^i|)$ の手間が必要であるから i から出る枝の重みをすべて求めるのに $O(|V^i||P|)$ 時間かかり，Step 1 の計算量は $O\left(\sum_{i \in V} |V^i||P|\right) = O(|V||P|)$ となる．また，$|\tilde{A}| = O(|P|^2)$ なので，Step 2 は上に述べたように $O(|P|^3)$ 時間で最大重みの U-マッチングを求める．Step 3 はマッチングの各枝に対する操作が必要なのでその計算量は $O(|P|)$ となる．　□

4.2　グリーディアルゴリズム

仮想計算機パッキング問題に対するある程度良い解を高速に求めるヒューリスティクとしてグリーディアルゴリズム（GREEDY）を提案する．GREEDY はできるだけ 1 台の物理計算機にたくさんの VM を割り当て物理計算機の稼働台数を削減する．

グリーディアルゴリズム（**GREEDY**）

Step 1　各物理計算機 i（$\in P$）に割り当てられている VM の CPU 使用量の総和とメモリ使用量の総和を計算し，$\sum_{k \in V^i} vc_k > c_i$ もしくは $\sum_{k \in V^i} vm_k > m_i$ となるもの $High$（$\subseteq P$）[*1]とそうでないもの Low（$\subseteq P$）に分類する．

Step 2　$High$ および Low を各物理計算機 i の資源使用率 $util(V^i)$ に応じてソートする．ここで，$util(V^i)$ を次のように定義する．

$$util(V^i) = \sum_{k \in V^i} vc_k/c_i + \sum_{k \in V^i} vm_k/m_i \qquad (15)$$

Step 3　物理計算機 h（$\in High$）に対して，$util(V^i)$ の高い順に次の操作を行う．物理計算機 h に割り当てられている VM の集合 V^h から適切な k' を選択し，Low のいずれかの物理計算機 l（$\in Low$）にマイグレートする．各物理計算機 h（$\in High$）に対して，CPU およびメモリ使用量の超過した分を $oc_h = \max\left\{0, \sum_{k \in V^h} vc_k - c_h\right\}$，$om_h = \max\left\{0, \sum_{k \in V^h} vm_k - m_h\right\}$ とすると，以下の優先順位でマイグレートする VM k' を決定する．

(a)　$vc_k \geq oc_h$ かつ $vm_k \geq om_h$ で *diff* が最小の k（$\in V^h$）

(b)　$vc_k < oc_h$ かつ $vm_k \geq om_h$ で *diff* が最小の k（$\in V^h$）

(c)　$vc_k \geq oc_h$ かつ $vm_k < om_h$ で *diff* が最小の k（$\in V^h$）

(d)　$vc_k < oc_h$ かつ $vm_k < om_h$ で *diff* が最小の k（$\in V^h$）

ここで，*diff* は以下のように算出する．

$$diff = |vc_k - oc_h|/c_h + |vm_k - om_h|/m_h \qquad (16)$$

マイグレート先の物理計算機 l（$\in Low$）は資源使用率の高い物理計算機から順に k'（$\in V^h$）が割当て可能なものを探し，割り当てる．$Low \setminus \{l\}$ とする．k' に対して，割当て可能なマイグレート先がないときは，V^h はそのままで，次の $h \in High$ に移る．

Step 4　Low に含まれる物理計算機のうち，資源の使用率が低い物理計算機 l_{send} から順に，マイグレート先 l_{recv} を決定していく．l_{recv} は資源使用率の高い物理計算機から $V^{l_{send}}$ に含まれるいずれかの VM を割当て可能なものを探して割り当て，$Low \setminus \{l_{recv}\}$ とする．割当て可能なマイグレート先がないときは，$V^{l_{send}}$ はそのままで，次の $l_{send} \in Low$ に移る．

以上の処理により，VM を割り当てる物理計算機数の最小化を試みる．GREEDY では，負荷の急激な上昇により再配置しても資源制約を満たさない場合も再配置を行う．

5.　評価実験

各 VM の CPU 使用量とメモリ使用量が期ごとに変化するとき，各期に仮想計算機パッキング問題を解いて，VM の再配置を行う設定で，200 期間での性能を測る．評価は，VMPP を商用ソルバを用いて解く IP と前節で提案した，MBA，GREEDY の平均消費電力量および平均計算時間で比較する．

5.1　実験環境

評価実験で対象とする設定では，物理計算機数と VM 数は等しく，すべての期にわたって一定とする．物理計算機数（= VM 数）は 50，100，150，200，250，300 と固定する 6 パターンを扱う．物理計算機の CPU のキャパシティ（コア数 × 利用率の上限）は 1.0 とし，メモリキャ

[*1] 集合 $High$ はマッチングベースアルゴリズムで定義した集合 U と同じ集合であるが，アルゴリズムの記述のため，別の記号を用いている．

表 1　パラメータ設定
Table 1　Parameters.

パラメータ	環境設定
アルゴリズム，ソルバ	MBA，GREEDY
	IP（CPLEX）
物理計算機数，VM 数	50 から 300 までの 50 刻み
期間	200
物理計算機の CPU キャパシティ	1.0
物理計算機のメモリキャパシティ	4,096 [MB]
VM の CPU 利用量の avg./max	0.5/1.0
VM のメモリ利用量の avg./max	2,048/4,096 [MB]
VM の CPU，メモリ利用量の変動	Sine カーブ（位相：一様分布，平均周期：$1/12\pi$，一様分布）
$base_i/cost_i(z)$	$53.0/47.0z$

パシティは 4,096 [MB] とする．各期における VM の CPU とメモリの使用量の平均値/最大値はそれぞれ，0.5/1.0，2,048/4,096 [MB] とし，その資源使用量の変動は，Sine カーブで与える．VM のデータについては，実環境での観測データが望ましいが，データ収集は困難なため，今後の課題とする．また，Sine カーブに基づくデータ生成は期ごとにすべての資源使用量が変動するため，実際の運用時に比べて厳しい条件である．消費電力に関するパラメータは，文献 [7] の調査結果をもとに，すべての $i \in P$ で $base_i = 53.0$，$cost_i(z) = 47.0z$ とする．初期配置は，各物理計算機に 1 台ずつ VM をランダムに割り当てたものとする．その他のパラメータ設定も含めて，**表 1** に示す．

実験は，Intel Core i7 CPU（3.4 GHz，8 コア），MacOSX 10.7 64 bit，メモリ 16 GB の計算機上で実行する．上で述べたように $cost_i$ を線形関数で与えるため，VMPP は整数線形計画問題となる．そこで，VMPP を解く商用ソルバとして，CPLEX（IBM ILOG CPLEX Optimization Studio_Academic V12.3，64 bit 版）を用い，スレッド処理を 8 スレッド使用する．また，タイムアウト時間を 600 秒とし，600 秒で最適解が出ない場合は暫定解を出力する．実験は，MBA と GREEDY および，CPLEX を用いて VMPP を解く IP の性能を比較する．CPLEX のインタフェースとして，Python の PuLP ライブラリ [12] を利用する．また，参考までに無償ソルバの 1 つである GLPK（Ver.4.47）[13] を用いた場合の計算時間も調査する．他のアルゴリズムについては，Python 2.6.5 で実装した．MBA の step 2 で最大重みのマッチングを求めるプロシージャは，文献 [9] で示されている $O(|P|^3)$ アルゴリズムを van Rantwijk [11] が実装し，公開しているものを，U-マッチングを出力するように修正したものを用いる．

5.2　実験結果

まず，各アルゴリズムの 1 期間あたりの平均電力消費量と平均計算時間を比較する．MBA はマッチングベースアルゴリズム，GREEDY はグリーディアルゴリズム，IP は VMPP（式 (3)–(10)）を CPLEX で解いた結果である．ただし，VMPP の制約 (8) は x_{ik} と x^0_{ik} の積が 1 ならば物理計算機 i 上に VM k が再配置されたことを意味する性質を利用し，

$$\sum_{k \in V}(x_{ik} + x^0_{ik} - 2x_{ik}x^0_{ik}) \leq 1,\ \forall i \in P \tag{17}$$

としている．また，FIX は，各期で再配置を行わず，1 台の物理計算機上に 1 つの VM を固定した場合の結果を表す．問題が実行不能のとき，MBA や IP では，再配置を行わず，1 期前の配置を用いる．一方，GREEDY では，実行不能時でもできるだけ消費電力量が小さくなるように再配置を行っている．**表 2** に MBA と IP の 200 期での実行不能回数を示す．MBA，IP ともに 50%以上で実行不能になっている．特に IP では，物理計算機数（= VM 数）が 200 以上の問題では，最初の期のみ実行可能で 2 期目以降すべて実行不能になっている．MBA でも最初の 20–30 期あたりまでは実行可能でもその後実行不能に陥り，実行可能な期はまばらになる．

表 2　MBA と IP の実行不能回数
Table 2　Number of infeasible rounds of MBA and IP.

物理計算機数（= VM 数）	実行不能回数 MBA	IP
50	177	195
100	119	197
150	152	198
200	143	199
250	137	199
300	130	199

そこで，MBA や IP でも実行不能時にも再配置を行うように，アルゴリズムの修正をする．IP では VMPP の制約 (5) と (6) をそれぞれ，制約を超過した量を表す非負変数 p_i，q_i を用いて

$$\sum_{k \in V} vm_k x_{ik} - m_i \leq m_i p_i,\ \forall i \in P \tag{18}$$

$$\sum_{k \in V} vc_k x_{ik} - c_i \leq c_i q_i,\ \forall i \in P \tag{19}$$

と置き換え，目的関数 (3) を

$$\sum_{i \in P}\left(cost_i\left(\sum_{k \in V} vc_k x_{ik}\right) + base_i y_i\right) + \sum_{i \in P} base_i \cdot (p_i + q_i) \tag{20}$$

の最小化とした整数計画問題を CPLEX で解いた結果を IPr に表す．

一方，MBA では，実行不能と判定された場合，有向完全グラフの枝の重みを，資源制約の超過率の減少量とすることで再割当てを行うように修正する．ある物理計算機 i か

表 3　1 期間あたりの平均電力消費量と平均計算時間の比較（VM = 50～300）
Table 3　Comparison of average power consumption and computation time per one round (VM = 50∼300).

物理計算機数	平均消費電力量						平均計算時間（秒）				
（= VM 数）	FIX	MBA	IP	GREEDY	IPr	MBA-mod	MBA	IP	GREEDY	IPr	MBA-mod
50	3,814.716	1,951.501	2,817.311	3,076.691	2,854.433	2,527.346	0.018	0.659	0.008	0.479	0.051
100	7,623.835	3,959.415	5,925.185	6,009.985	5,601.287	5,044.060	0.076	13.243	0.026	5.492	0.208
150	11,469.435	5,841.630	9,395.015	8,947.165	8,229.118	7,614.215	0.157	12.360	0.052	10.575	0.472
200	15,259.594	7,752.674	11,755.764	11,814.064	10,842.380	10,124.424	0.280	22.335	0.089	72.597	0.834
250	19,055.052	9,579.447	14,552.893	14,657.112	13,023.482	12,636.487	0.433	21.358	0.133	285.430	1.346
300	22,840.991	11,423.201	17,479.776	17,541.521	15,976.583	15,128.166	0.625	48.815	0.187	530.083	1.943

ら j に VM k を移動させたときの資源制約超過率 $\hat{c}(i,j,k)$ を GREEDY で用いた $util$ を用いて次のように定義する．

$$util(V^i) + util(V^j) - util(V^i \setminus \{k\}) - util(V^j \cup \{k\}). \tag{21}$$

したがって，各枝 $(i,j) \in A$ の新たな重み $c'(i,j)$ を

$$\max\{\hat{c}(i,j,k) \mid k \in V_i^0\} \tag{22}$$

と定義し，(P,A) 上の最大重みマッチングを求めることによって実行不能時でも，できるだけ資源制約超過率を削減するような再割当てを行うことができる．実行不能時の処理を追加した場合の結果を MBA-mod に示す．

表 3 に各アルゴリズムの平均消費電力量と平均計算時間を示す．MBA は IP や GREEDY に比べ，多くの場合で平均消費電力量が小さいことが分かる．MBA と IP の平均消費電力量が異なっているのは，最適解が複数ある場合，異なる再配置になることがあり，かつ IP では，指定時間（600 秒）内に最適解が求められない場合，暫定解を用いているためであると考えられる．なぜならば，$cost_i$ を線形関数としているため，物理計算機の稼働している台数で消費電力量が決まるため，VM をどこに移すかの選択肢が複数ある場合，異なる再配置になる可能性が十分に考えられるからである．IP と IPr を比較すると，ほとんどの場合 IPr の方が平均消費電力量が小さいことから，実行不能時にも再配置を行う意義が分かる．また，MBA と MBA-mod を比較すると，MBA-mod の方が平均消費電力量が大きくなっている．これは，実行不能時，なるべく資源制約の超過量が少なくなるように省電力モードの物理計算機に VM を再割当てしているためと考えられる．計算時間は，GREEDY が圧倒的に短く，MBA と IP を比較すると，MBA の方が計算時間が短い．CPLEX は整数計画問題や線形計画問題に対する汎用ソルバとして企業の生産計画などに使われているが，問題構造を把握し，適切な多項式時間アルゴリズムを構築することで，計算時間を抑え，最適解を得られることが分かった．また，物理計算機数（= VM 数）が 400 以上の問題を MBA と GREEDY で解いた結果を**表 4** に示す．MBA は，物理計算機数（= VM 数）が 1,000 で平均計算時間は 8.5 秒となり，GREEDY の計算時間には劣るものの，十分に実用的な時間であることが分かる．

表 4　1 期間あたりの平均電力消費量と平均計算時間の比較（VM = 400～1,000）
Table 4　Comparison of average power consumption and computation time per one round (VM = 400∼1,000).

物理計算機数	平均消費電力量			平均計算時間（秒）	
（= VM 数）	FIX	MBA	GREEDY	MBA	GREEDY
400	30,507.199	15,273.144	23,375.519	1.145	0.314
500	38,130.399	19,116.119	29,155.379	1.890	0.480
600	45,808.771	22,914.891	34,862.151	2.771	0.651
700	53,408.342	26,723.107	40,640.907	3.892	0.870
800	60,994.286	30,621.311	46,263.201	5.090	1.132
900	68,603.518	34,603.753	52,057.448	6.719	1.431
1,000	76,260.032	38,524.827	57,715.597	8.499	1.718

さらに，整数計画問題 VMPP を解くのに GLPK を用いると，物理計算機数（= VM 数）が 10 のとき，平均計算時間が 0.168 (sec.) であり，物理計算機数（= VM 数）が 20 のときは，200 期間中，600 秒以内で解ける期はなく，暫定解もあまり良い解とは期待できない結果であった．

次に**図 2** に物理計算機数（= VM 数）が 250 のときの FIX と MBA，GREEDY，IP，IPr，MBA-mod の消費電力量の推移を示す．横軸に期間を，縦軸にその期での消費電力量を表す．ほとんどの期間でどのアルゴリズムも消費電力量の増加減の傾向は同じであることが分かる．また，FIX と比べると MBA，GREEDY，IP，IPr，MBA-mod ともに大幅に消費電力量を削減できていることが分かる．また，MBA の結果の上の点は MBA が実行可能である期を示している．

表 5 は各アルゴリズムの消費電力の削減率を表していて，再配置をしなかった場合に比べ，どのアルゴリズムも消費電力を 18%から 50%削減できていることが分かる．

最後に，再配置による性能劣化を比較する．ユーザが感じる性能劣化には様々な要素が絡んでいるが，ここでは，

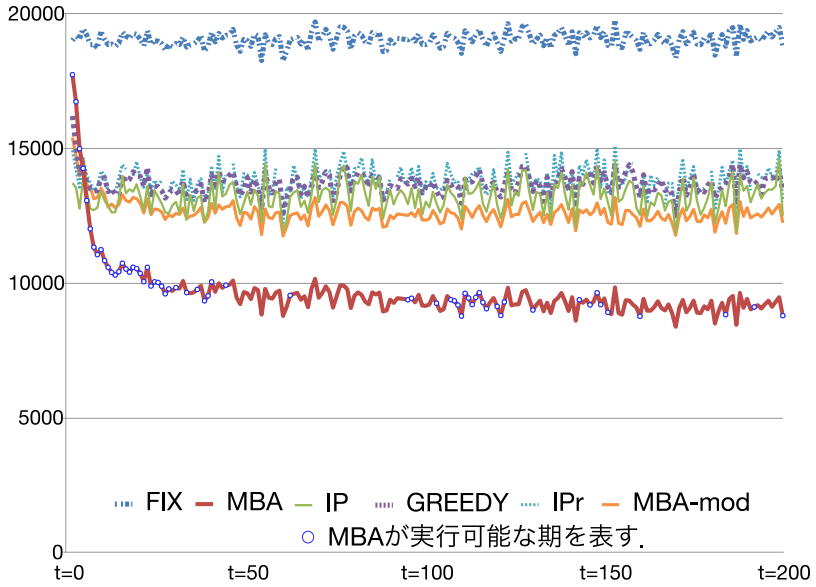


図 2　物理計算機数（= VM 数）250 のときの FIX，MBA，MBA-mod，GREEDY，IP，IPr の消費電力量の推移

Fig. 2　The power consumption fluctuation of FIX, MBA, MBA-mod, GREEDY, IP and IPr in each round at a problem size of 250.

表 5　各アルゴリズムを用いた場合の消費電力削減率

Table 5　Power consumption reduction ratio of each algorithm.

物理計算機数（= VM 数）	MBA	IP	GREEDY	IPr	MBA-mod
50	0.49	0.19	0.26	0.25	0.34
100	0.48	0.21	0.22	0.27	0.34
150	0.49	0.22	0.18	0.28	0.34
200	0.49	0.23	0.23	0.29	0.34
250	0.50	0.23	0.24	0.32	0.34
300	0.50	0.23	0.23	0.30	0.34

表 6　アルゴリズムごとの平均性能劣化数

Table 6　Average performance degradation of user experience.

物理計算機数（= VM 数）	平均性能劣化数				
	MBA	IP	GREEDY	IPr	MBA-mod
50	1.34	8.05	0.17	0.56	0.91
100	1.21	18.39	0.40	1.11	0.93
150	1.16	23.47	0.70	1.59	0.93
200	0.65	34.66	1.07	2.51	0.57
250	0.41	36.68	1.37	3.04	0.36
300	0.61	48.11	1.56	7.67	0.49

性能劣化を表す指標として，CPU およびメモリの使用量の何れかが容量を超えている割合を表す式 (23) で定義される性能劣化数を用い，数値が大きくなるほどユーザ体験が劣化することを意味している．各物理計算機 i（$\in P$）に対して，保持している VM の集合を V^i とする．このとき，性能劣化数は，

$$\sum_{i\in P}\max\left\{0,\sum_{k\in V^i}vc_k-c_i\right\}/c_i + \sum_{i\in P}\max\left\{0,\sum_{k\in V^i}vm_k-m_i\right\}/m_i \qquad (23)$$

で定義される．**表 6** に各アルゴリズムを用いた場合の再配置後の性能劣化数の平均を示す．表 6 を見ると，MBA や MBA-mod 以外のアルゴリズムでは問題サイズの増加に比例して性能劣化数が増加している．一方で，MBA-mod は最も性能劣化数が小さかった．これは，性能劣化が最小となるようなマッチングを出力しているためである．また，IP と IPr では，IPr の方が性能劣化数が小さいが，これは実行不能時の再配置の重要性を示唆している．

提案する MBA は物理計算機がマルチコアのモデルなどにもそのまま拡張できる．一方で，通信通信衝突を回避するために 1 つの計算機が送受信する VM 数の上限を 1 としており，MBA はこの性質を利用していた．送受信数を 2 以上としてより消費電力の削減を試みる方針も考えられるが，送受信数が 2 回以上を許す場合，仮想計算機パッキング問題は，3-分割問題を帰着させることで NP-困難性を示せる．よって，MBA のような効率的な解法は望めず，3 章で与えた整数計画問題をソルバで解くことや，GREEDY のようなヒューリスティックアプローチが良い．また，IP では，$cost_i$ が線形関数なので解くことができたが，MBA や GREEDY では，どのような関数であっても実行することができる．そのため，消費電力が複雑に変化するようなモデルにも適用できる．さらに，提案手法は VM 上のシステムのダウンタイムを最小限にするために高速ポストコピーマイグレーションを用いた VM パッキングシステムへの適用を前提としているが，プレコピーマイグレーションを前

提としたシステムにおいても適用できる．プレコピーマイグレーションでは，メモリイメージの転送時間がVM上のシステムのダウンタイムに直接反映されるため，提案する通信衝突を回避した再配置により高速なメモリイメージ転送が可能となり，ダウンタイムを短縮することができる．

6. おわりに

本研究では，仮想計算機パッキング問題に対して，多項式時間アルゴリズムのマッチングベースアルゴリズムと，ヒューリスティックであるグリーディアルゴリズムを提案した．評価実験では，最適化ソルバと比較し，マッチングベースアルゴリズムはグリーディアルゴリズムには実行時間では劣るものの，実用的な時間でより効果的な消費電力の削減が可能であり，性能劣化も小さいことを示した．また，商用ソルバよりも短時間で最適解を得られることを示した．

今後は，各期での再配置をトレースできるように実験を修正し，各アルゴリズムでの再配置の乖離を検証すること，計算機実験のモデルを，実システムに近づけアルゴリズムの効果を検証すること，大域的な消費電力の削減効果を評価するために多期間仮想計算機パッキング問題に拡張し，提案アルゴリズムの大域的な性能評価を行う．また，提案手法を実VMパッキングシステムに適用し，CPUやメモリなどの資源の使用量に基づく性能劣化に加えて複数の実アプリケーションにおける性能劣化について調査し，提案手法の実用性を示す．

謝辞　本研究の遂行にあたり，有益な示唆をいただいた産業技術総合研究所石井紀代博士に感謝する．また，本研究の一部は，CREST（情報システムの超低消費電力化を目指した技術革新と統合化技術）および，科研費（22510135）の助成を受け実施した．

参考文献

[1] Hirofuchi, T., Nakada, H., Itoh, S. and Sekiguchi, S.: Reactive Consolidation of Virtual Machines Enabled by Postcopy Live Migration, *Proc. 5th Int. Workshop on Virtualization Technologies in Distributed Computing*, pp.11–18 (2011).

[2] Nakada, H., Hirofuchi, T., Ogawa, H. and Itoh, S.: Toward Virtual Machine Packing Optimization Based on Genetic Algorithm, *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, Proc. Int. Symposium on Distributed Computing and Artificial Intelligence*, LNCS, Vol.5518, pp.651–654, Springer (2009).

[3] 中田秀基，竹房あつ子，広渕崇宏，伊藤　智，関口智嗣：仮想計算機パッキングへの最適化手法の適用，電子情報通信学会技術研究報告（CPSY：コンピュータシステム），Vol.110, No.167, pp.55–60 (2010).

[4] 竹房あつ子，中田秀基，広渕崇宏，伊藤　智，関口智嗣：省電力化にむけた仮想計算機パッキングアルゴリズムの提案，電子情報通信学会技術研究報告（CPSY：コンピュータシステム），Vol.110, No.167, pp.73–78 (2011).

[5] Hirofuchi, T., Nakada, H., Itoh, S. and Sekiguchi, S.: Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension, *The 10th IEEE/ACM Int. Conference on Cluster, Cloud and Grid Computing* (*CCGrid2010*), pp.73–83 (2010).

[6] 広渕崇宏，中田秀基，伊藤　智，関口智嗣：高速マイグレーションを利用した仮想マシン配置最適化システムの検討，情報処理学会研究報告（2010-OS-115），pp.1–13 (2010).

[7] Hirofuchi, T., Nakada, H., Itoh, S. and Sekiguchi, S.: Making VM Consolidation More Energy-efficient by Postcopy Live Migration, *Proc. 2nd Int. Conference on Cloud Computing, GRIDs, and Visualization*, pp.195–204 (2011).

[8] Korte, B.H. and Vygen, J.: *Combinatorial optimization: Theory and algorithms*, Springer (2004).

[9] Galil, Z.: Efficient Algorithms for Finding Maximum Matching in Graphs, *ACM Computing Serveys*, Vol.18, No.1, pp.23–38 (1986).

[10] Edmonds, J.: Paths, trees, and flowers, *Canadian Journal Mathematics*, Vol.17, pp.449–467 (1965).

[11] van Rantwijk, J.: Joris_VR, available from ⟨http://jorisvr.nl/maximummatching.html⟩ (accessed 2012-07-24).

[12] PuLP, available from ⟨http://code.google.com/p/pulp-or/⟩ (accessed 2012-07-24).

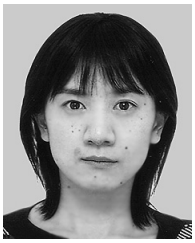
[13] GLPK, available from ⟨http://www.gnu.org/software/glpk/⟩ (accessed 2012-07-24).

[14] Hermenier, F., Lorca, X., Menaud, J.-M., Muller, G. and Lawall, J.: Entropy: A consolidation manager for clusters, *Proc. 5th Int. Conference on Virtual Execution Environments*, pp.41–50 (2009).

[15] Stillwell, M.L., Vivien, F. and Casanova, H.: Dynamic Fractional Resource Scheduling for HPC Workloads, *Proc. Int. Parallel and Distributed Processing Symposium* (*IPDPS*), pp.1–12 (2010).

高橋 里司　（学生会員）

1985年生．2008年山形大学工学部情報科学科卒業．2010年筑波大学大学院システム情報工学研究科博士前期課程修了，2010年より同大学院博士後期課程に在籍，2011年日本学術振興会特別研究員（DC2），電子商取引における最適化アルゴリズムの研究に従事．日本OR学会，日本応用数理学会各会員．



竹房 あつ子　（正会員）

1996 年お茶の水女子大学理学部情報科学科卒業．1998 年同大学大学院理学研究科情報科学専攻修士課程修了．2000 年同大学院人間文化研究科複合領域科学専攻博士課程修了．博士（理学）．同年日本学術振興会特別研究員，2002 年お茶の水女子大学理学部助手．2005 年独立行政法人産業技術総合研究所入所．現在，同所情報技術研究部門研究員．並列分散処理，グリッド，クラウドコンピューティング，スケジューリングに興味を持つ．ACM，電子情報通信学会各会員．

繁野 麻衣子

1991 年東京理科大学工学部経営工学科卒業．1993 年同大学大学院修士課程修了．1995 年東京工業大学大学院理工学研究科情報科学専攻博士後期課程退学．1996 年博士（理学）．1995 年東京工業大学助手．1997 年筑波大学講師を経て，現在，准教授．組合せ最適化の研究に従事．

中田 秀基　（正会員）

1967 年生．1990 年東京大学工学部精密機械工学科卒業．1995 年同大学大学院工学系研究科情報工学専攻博士課程修了．博士（工学）．同年電子技術総合研究所研究官．2001 年独立行政法人産業技術総合研究所に改組．現在，同所情報技術研究部門主任研究員．2001 年より 2005 年度まで東京工業大学客員助教授を兼務．グローバルコンピューティング，並列実行環境に関する研究に従事．ACM 会員．

工藤 知宏　（正会員）

1991 年慶應義塾大学大学院理工学研究科博士課程単位取得退学．東京工科大学助手，講師，助教授を経て，1997 年新情報処理開発機構並列分散システムアーキテクチャつくば研究室長，2002 年より産業技術総合研究所，現在，同所情報技術研究部門副研究部門長．博士（工学）．並列分散処理，通信アーキテクチャに関する研究に従事．電子情報通信学会，IEEE-CS 各会員．

吉瀬 章子

1962 年生．1985 年東京工業大学理工学部経営工学科卒業．1987 年同大学大学院修士課程修了．1990 年同博士課程単位取得退学．1990 年工学博士（東京工業大学）．1990 年筑波大学準研究員．1991 年同大学講師．1993 年同大学助教授．2007 年同大学教授．連続最適化に対するアルゴリズム，サービス産業における最適化モデルの研究に従事．1992 年 INFORMS（米国 OR 学会）Computing Society 賞共同受賞．1993 年 INFORMS（米国 OR 学会）F.W. Lanchester 賞共同受賞．2003 年船井情報科学振興賞共同受賞．2007 年日本 OR 学会文献賞受賞．2011 年日本 OR 学会フェロー．日本 OR 学会，日本応用数理学会，INFORMS，Mathematical Programming，SIAM 各会員．