

## 大規模マルチエージェントシステムにおけるエージェント配置

宮田 直輝<sup>†a)</sup> 石田 亨<sup>†</sup>

Placing Agents in Massively Multi-Agent Systems

Naoki MIYATA<sup>†a)</sup> and Toru ISHIDA<sup>†</sup>

あらまし 近年、多数のエージェントを扱う大規模なマルチエージェントシステムが開発されている。大規模マルチエージェントシステムは、分散化による利点を得るために、複数のエージェントサーバを接続するアプローチがとられる。エージェントが複数のサーバに分散する場合、システムの性能を効率的に改善するためには、エージェントの計算負荷とエージェント間のインタラクションに注目してエージェントを配置する必要がある。本論文では、互いにインタラクションを頻繁に行うエージェント集合であるコミュニティに対する評価をもとに、その配置を決定する Community-based Load Balancing (CLB) アルゴリズムを提案する。CLB は解の精度を保証することができないため、本論文ではシミュレーションを行った。そして CLB は先行研究と比べて、計算コストが大きくなるが、エージェントのインタラクションが構成するネットワークが正則、または小世界ネットワークの場合にインタラクションコストが小さな配置を行うことを示した。

キーワード 大規模マルチエージェントシステム、エージェント配置

### 1. ま え が き

近年、多数のエージェントを扱う大規模マルチエージェントシステムが開発されている [1]。大規模マルチエージェントシステムを構築することで、より多くの利用者にサービスを提供するシステムを構築することや、より複雑な系を対象とするシミュレーションを行うことができる。例えば、数百万規模の交通をマルチエージェントシステムにより再現するシミュレーション [2] や、多数の利用者を想定したサービスをマルチエージェントシステムによって提供するシステム [3] の開発が行われている。こうした大規模なマルチエージェントシステムを構築する際に生じる問題を解決するための研究が、現在進められている [4]。

大規模マルチエージェントシステムにおける問題点の一つは、エージェント数の増加に従ってシステムの性能が低下することである。システムが扱うエージェントの数が増加すると、エージェントの計算負荷や利用者からのアクセス負荷が増加し、リアルタイムな

サービスを提供することや、効率的にシミュレーションを行うことが困難になる。

大規模マルチエージェントシステムの性能が低下する問題に対して、エージェントシステムを分散化するアプローチをとることができる。複数のサーバからなる分散マルチエージェントシステムを構築することで、システムのリソースを増加させることができる。その結果、システムのリソース不足による性能の低下を解決することができる。こうしたシステムの分散配置は、分散 OS やワークステーションのクラスタ化に適用されており、その有効性が示されている [5]。

しかし、サーバを分散配置する場合、分散 OS などに適用されてきた計算負荷に基づく手法では不十分である。エージェントが分散配置されることで、インタラクションがサーバ間で発生し、システムのインタラクションコストは増加する。インタラクションが頻繁に発生する場合、このようなインタラクションコストの増加を無視することはできない。したがって、エージェントの計算負荷とインタラクションコストに基づいて、その配置を決定しなければならない。

本論文では、都市規模の交通シミュレーションや誘導システムといった、数千体、数万体ものエージェントを扱う大規模な分散マルチエージェントシステムに

<sup>†</sup> 京都大学社会情報学専攻, 京都市  
Department of Social Informatics, Kyoto University,  
Yoshida-Honmachi, Kyoto-shi, 606-8501 Japan  
a) E-mail: miyata@ai.soc.i.kyoto-u.ac.jp

において、計算負荷を分散し、通信コストを小さくするようなエージェント配置を決定する手法を提案する。本研究では上記の問題を、各サーバの計算負荷がしきい値より小さく、サーバ間で発生するインタラクションコストを最小にするようなエージェントの配置を求める、エージェント配置問題として定式化する。そして、定式化したエージェント配置問題の最適解を求めることは困難であるため、エージェント配置問題に対する近似解を求める手法を提案する。本研究では、提案手法によるシステムのインタラクションコストの変化を検証するために、シミュレーションを行った。

## 2. 関連研究

分散環境におけるプロセスまたはエージェントの配置手法は、High Performance Computing (HPC) やモバイルエージェント、マルチエージェントの分野で提案されている。これらの手法は、プロセスやエージェントの計算負荷と通信コストに基づいてその配置を決定することで、システムの性能を改善する手法を提案している。

文献 [6] では、分散配置されたノードに対して、あるノードが保持する多数のタスクを、システムの性能が最大となるように割り当てる手法が提案されている。分散されたタスクの負荷は、そのデータ量と分散方法によって定まるため、静的に解析することで適当な配置を行うことができる。一方で、マルチエージェントシステムでは、エージェントが自律的に動作するため、その振舞いは動的に変化し、あらかじめ予測することは困難である。したがって、システムの負荷が動的に変化の中で、システムの性能を改善するように配置を決定しなければならない。

文献 [7], [8] では、エージェントがタスクを実行する際の通信コストを最小にするように、移動を行う手法が提案されている。これらの手法は主に 1 体のエージェントに注目し、その通信コストが小さくなるように移動を決定する。しかし、エージェント間でインタラクションが発生するマルチエージェントシステムにおいては、あるエージェントの移動は、他のエージェントの通信コストに影響を与える。したがって、複数のエージェントの通信コストを考慮してその配置を決定しなければならない。

文献 [9] ~ [11] では、エージェントの計算負荷と通信コストに基づいて、エージェントの配置を決定する手法が提案されている。これらの手法に共通する特徴

として、エージェント単体の評価に基づいて、その移動と移動先を決定していることが挙げられる。例えば [10] が提案する Comet アルゴリズムは、エージェント単体に対する評価をもとに、移動するエージェントとその移動先を決定し、過負荷のサーバの計算負荷がしきい値よりも小さくなるまで移動を反復する。しかし、4.2 で述べるように、エージェントの集合に対する評価をもとに、その配置を決定することで、よりインタラクションコストを小さくすることができる。

## 3. エージェント配置問題

本章では、エージェントが複数のサーバに分散する際に考慮すべき点を述べ、エージェント配置問題として定式化を行う。

### 3.1 分散環境におけるエージェント配置

本研究で考えるマルチエージェントシステムは、ネットワーク上で接続された  $m$  台のサーバに  $n$  体のエージェントが分散配置されているものとする。このマルチエージェントシステム上では、エージェントは必要に応じて相互にインタラクションを行う。互いに異なるサーバに存在するエージェント間のインタラクションは、互いのエージェントが存在するサーバ間でメッセージを交換することで実現可能であるとする。そして、各サーバの性能は同じであり、サーバ間の接続はすべて等価であると仮定する。このようなシステムにおいて達成すべき事柄は、以下の 2 点である。

- 各サーバにエージェントの計算負荷を分散する
- サーバ間で発生する通信コストを最小にする

まず、エージェントの計算負荷を各サーバに分散する必要がある。あるサーバに対して負荷が集中することによって性能が低下することを防ぐため、各サーバの性能が大きく低下しないよう、エージェントを分散配置する必要がある。

次に、サーバ間で発生するインタラクションコストを抑える必要がある。あるエージェントが異なるサーバに存在するエージェントに対してインタラクションを行う場合、インタラクションのためのメッセージをサーバ間で通信する必要がある。したがって、サーバ間のインタラクションが発生することで、システムにおけるインタラクションコストが増加する。こうしたインタラクションコストの増加を抑えるために、頻繁にインタラクションを行うエージェント同士を同じサーバに配置し、サーバ間で発生するインタラクションの頻度を少なくする必要がある。

エージェント間のインタラクションには、頻繁にその相手と頻度が変化する場合と、変動はあるが、統計的にはその変動が小さい場合が想定される。本研究におけるインタラクションは、統計的には変動が小さい場合を想定する。変動が小さい場合、それまでのインタラクションの傾向から、インタラクションコストが小さくなるような配置を行うことができる。一方で、インタラクションの変動が大きい場合、その変動を予測して配置を決定することが必要である。

### 3.2 エージェント配置問題の定式化

表 1 の表記に基づいて、エージェント配置問題を以下のように定式化する。

エージェント集合  $A$  の計算負荷  $W(A)$  は、式 (1) に示すように、エージェント  $a_j$  の計算負荷  $w_j$  の総和で与えられるとする。

$$W(A) = \sum_{a_j \in A} w_j \quad (1)$$

また、表記を簡単にするため、エージェント集合間のインタラクションコストとして、式 (2) で定まる  $P(A_i, A_j)$  を定義する。 $P(A_i, A_j)$  は二つのエージェント集合  $A_i, A_j$  間のインタラクションコストの総和である。

$$P(A_i, A_j) = \sum_{a_k \in A_i} \sum_{a_l \in A_j} p(a_k, a_l) \quad (2)$$

ここで、サーバ  $s_1$  が過負荷であり、他のサーバ  $s_i$  ( $i = 2, 3, \dots, m$ ) は過負荷でないとする。そして、過負荷のサーバから過負荷でないサーバにエージェントを移動することで、各サーバの計算負荷をしきい値以下にすることができるものとする。つまり、初期条件として式 (3), (4), (5) が満たされているものとする。

$$W(A_1) > T_h \quad (3)$$

表 1 表記  
Table 1 Notations.

$a_i$	エージェント $i$ ( $i = 1, 2, \dots, n$ )
$s_i$	エージェントサーバ $i$ ( $i = 1, 2, \dots, m$ )
$w_i$	エージェント $a_i$ の計算負荷
$W(A)$	エージェント集合 $A$ の計算負荷
$p(a_i, a_j)$	$a_i$ と $a_j$ の間のインタラクション頻度
$P(A_i, A_j)$	$A_i$ と $A_j$ の間のインタラクション頻度
$A_i$	移動前に $s_i$ が保持するエージェント集合
$A'_i$	移動後に $s_i$ が保持するエージェント集合
$T_h$	エージェントサーバの計算量しきい値

$$W(A_i) < T_h \quad (i = 2, 3, \dots, m) \quad (4)$$

$$\sum_{1 \leq i \leq m} W(A_i) < mT_h \quad (5)$$

このとき、過負荷のサーバ  $s_1$  から他のサーバ  $s_i$  ( $i = 2, 3, \dots, m$ ) にエージェントを移動して、負荷の分散を行うことを考える。 $s_1$  から  $s_i$  に移動するエージェントの集合を  $M_{1,i}$  とおくと、エージェントを移動した後の各サーバが保持するエージェント集合  $A'_i$  は、式 (6) で与えられる。

$$A'_i = \begin{cases} A_1 \setminus \cup_{2 \leq j \leq m} M_{1,j} & (i = 1) \\ A_i \cup M_{1,i} & (i = 2, 3, \dots, m) \end{cases} \quad (6)$$

上記の式を用いて、サーバの負荷を均等にし、かつサーバ間で発生するインタラクション量を抑えるという目的は式 (7), (8) として定式化することができる。

$$\min \sum_{1 \leq k \leq m} \sum_{k < l \leq m} P(A'_k, A'_l) \quad (7)$$

$$s.t. W(A'_i) < T_h \quad (i = 1, 2, \dots, m) \quad (8)$$

式 (7), (8) を満たす  $M_{1,i}$  ( $i = 2, 3, \dots, m$ ) が、エージェント配置問題における最適なエージェント移動集合である。

$|A_1|$  が大きい場合、すべての配置の中から式 (7) を満たすエージェントの配置を決定することは困難である。なぜならば、 $|A_1|$  体のエージェントに対して、エージェントの移動集合の選び方は  $m^{|A_1|}$  通り存在するからである。そこで 4. で、エージェント配置問題に関する近似アルゴリズムを定める。

## 4. エージェント配置問題に対する近似解

本章では、エージェント配置問題に対して本論文が提案する近似解法と、その解法の特徴について述べる。

### 4.1 Community-based Load Balancing

過負荷になった  $s_1$  は、以下の情報が取得できるとする。

- 他のサーバの計算負荷
- 内部エージェントの計算負荷
- 内部エージェント間のインタラクション頻度
- 各内部エージェントと外部サーバ間のインタラクション頻度

エージェント集合  $A$  の評価値として、 $C(A)$  を定義する。 $C(A)$  は、 $A$  を加えても計算負荷がしきい値を超えないサーバのうち、 $A$  が最もインタラクションを

**Algorithm 1** Community-based Load Balancing

---

**Require:**  $W(A_1) > T_h$ ,  $W(A_i) < T_h (i = 2, 3, \dots, m)$ ,  
 $\sum_{1 \leq i \leq m} W(A_i) < mT_h$

**loop**  
 $cost_{min} \leftarrow \infty$  : minimum  $C(A)$  in the following loop  
 $M_{min} \leftarrow \phi$  : set of agents providing  $cost_{min}$   
 $toservind \leftarrow null$  : index of the server which  $M_{min}$  moves to

**for**  $a_i \in A_1$  **do**  
 $M, s \leftarrow OptimalSet(a_i)$   
**if**  $cost_{min} > C(M)$  **then**  
 $cost_{min} \leftarrow C(M)$   
 $M_{min} \leftarrow M$   
 $toservind \leftarrow$  index of  $s$   
**end if**  
**end for**

**if**  $W(A_1) \leq T_h$  and  $cost_{min} \geq 0$  **then**  
 end loop  
**end if**  
 move  $M_{min}$  to  $toservind$  ( $A_1 \leftarrow A_1 \setminus M_{min}$ ,  
 $A_{toservind} \leftarrow A_{toservind} \cup M_{min}$ )  
**end loop**

---

行うサーバに移動した場合の、インタラクションコストの変化量である。  $A$  を移動可能なサーバが存在しない場合、その評価値は無限大であるとする。

$$C(A) = \begin{cases} P(A, A_1 \setminus A) \\ - \max_{\{m_k | W(A_k \cup A) < T_h\}} P(A_k, A) \\ (\exists m_i | W(A_i \cup A) < T_h) \\ \infty & (\forall m_i | W(A_i \cup A) > T_h) \end{cases} \quad (9)$$

我々が提案する Community-based Load Balancing (CLB) の擬似コードを Algorithm 1 に示す。 CLB はエージェントの移動集合  $M_i$  ( $i = 2, 3, \dots, 4$ ) の近似解を、山登り法で求める。 まず、操作  $OptimalSet$  を定義する。  $OptimalSet$  は 1 体のエージェントを引数にとり、そのエージェントがともに移動する最適なエージェント集合と、その移動先を返す。 CLB は、各エージェントに対して  $OptimalSet$  を適用し、  $C(A)$  が最も小さいエージェント集合を、  $OptimalSet$  により得られる移動先サーバに移動する。 この操作を移動元サーバの計算負荷がしきい値を下回り、かつ評価値が負の (移動によりインタラクションコストが減少する) エージェント集合が見つからなくなるまで繰り返す。

次に、操作  $OptimalSet$  について述べる。  $OptimalSet$  は引数としてエージェントを受け取り、そのエージェントとともに移動する最適なエージェント集合と、移動先サーバを返す。 最適なエージェント

**Algorithm 2** ApproximateOptimalSet ( $a$ )

---

$Move \leftarrow \phi$  : set of agents which move with  $a$   
 $Candidate \leftarrow \{a\}$  : set of candidates for  $Move$   
 $C_{min} \leftarrow \infty$  : minimum  $C(A)$  in the following loop  
 $M_{min} \leftarrow Move$  : set of agents providing  $C_{min}$

**repeat**  
 Select  $a_i \in Candidate$  which has the least  $c(a_i)$   
 $Move \leftarrow Move \cup \{a_i\}$   
 $Candidate \leftarrow Candidate \setminus \{a_i\}$   
**if**  $C_{min} > C(Move)$  **then**  
 $M_{min} \leftarrow Move$   
 $C_{min} \leftarrow C(Move)$   
**end if**  
 Add  $\{a_j | p(a_i, a_j) > 0, a_j \in A_1 \setminus \{Candidate \cup Move\}\}$   
**until**  $W(Move \cup A_i) > T_h (i = 2, 3, \dots, m)$   
**return**  $M_{min}$  and  $s$  (server which  $M_{min}$  moves to in  $C_{min}$ )

---

集合を求めることは、一般的に困難であるため、本論文では  $OptimalSet$  の近似解を求める  $ApproximateOptimalSet$  を提案する。  $ApproximateOptimalSet$  の擬似コードを Algorithm 2 に示す。 まず、移動エージェント集合  $Move$  と、移動候補エージェント集合  $Candidate$  を定める。 初期状態で  $Move$  は空集合であり、  $Candidate$  は引数で与えられたエージェントのみを含む集合である。 そして以下の操作を、  $W(Move)$  がすべてのサーバの許容量を上回るまで反復する。 まず、候補エージェントの中から、式 (10) で定められる評価値  $c(a_i)$  が最も小さいエージェントを選択し、移動エージェント集合に加える。  $c(a_i)$  は、そのエージェントを  $Move$  に加えることによる、  $C(Move)$  の変化量である。

$$c(a_i) = C(Move + a_i) - C(Move) \quad (10)$$

次に、移動エージェント集合に加えたエージェント  $a$  と同じサーバに存在し、かつ  $a$  とインタラクションを行い、  $Candidate$  や  $Move$  に含まれないエージェントを候補エージェントに加える。 以上の操作の反復の中で、最も評価値が小さいエージェント集合を選択し、評価値  $C(A)$  が得られる移動先のサーバとともに返す。

**4.2 特徴**

CLB は、エージェントの配置を決定するために、移動するエージェント集合を考えている。 通常、大規模なマルチエージェントシステムにおいては、過負荷なサーバから他のサーバに負荷を分散するためには、複数のエージェントを移動することが必要である。 複数のエージェントを移動する場合、移動するエージェントの集合を考えることで、単体ごとにエージェントを

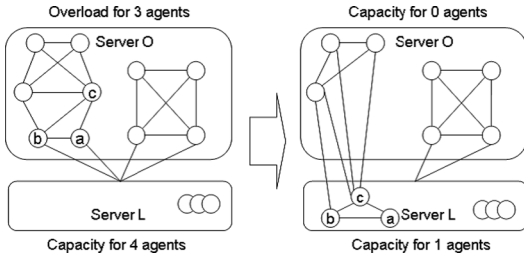


図 1 エージェント単体に対する評価に基づいて配置を決定する場合

Fig. 1 Agent movement based on evaluation of each agent.

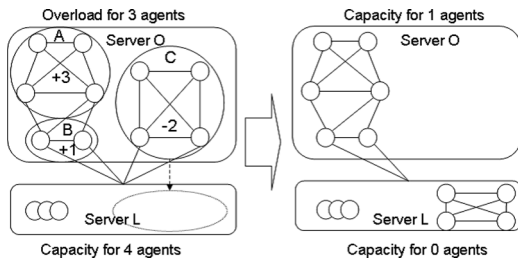


図 2 コミュニティに対する評価に基づいて配置を決定する場合

Fig. 2 Agent movement based on evaluation of each community.

移動した場合よりも、インタラクションコストを小さくすることができる。

エージェント単体を評価する場合と、コミュニティを評価する場合で、移動後のインタラクションコストが異なる例を図 1 と図 2 に示す。これらの図には、エージェントと、その間のインタラクション、サーバが表されている。ここでサーバ A はエージェント 3 体分過負荷であるとし、サーバ B は 4 体分の許容量があるとす。図 1 が、エージェント単体の評価値に基づいて、エージェントの移動を決定する場合である。エージェントの評価値を、移動によるインタラクションコストの変化量とすると、図 1 の場合はエージェント a, b, c の順に移動が行われる。エージェントを移動した後の外部インタラクションコストは、開始前の 4 から 6 に増加する。エージェント a が移動したことで、b, c が続いて移動したが、a が属するコミュニティが、移動先のサーバの許容量よりも大きいため、コミュニティが分割され、インタラクションコストが増加した。

図 2 が、コミュニティに対して評価を行った場合である。A, B, C のコミュニティの評価値は、移動

することによるインタラクションコストの変化量であるとする、それぞれ +3, +1, -2 である。このとき、評価値が最も小さいコミュニティ C を移動することで、インタラクションコストは 4 から 2 に減少する。したがって、エージェント単体の評価に基づく場合と比較すると、インタラクションコストが 4 小さい。コミュニティの評価に基づいて移動を決定することで、コミュニティの分割によるインタラクションコストの増加を防ぐことができる。

## 5. エージェント配置シミュレーション

本章では、CLB の有効性を検証するために行ったシミュレーションについて述べる。

### 5.1 シミュレーション内容

このシミュレーションは、複数のサーバを接続した分散マルチエージェントシステムをシミュレートしたものである。このシミュレーションでは、エージェント数  $n = 1,000$ 、サーバ数  $m = 10$  としている。各エージェント間のインタラクション頻度  $p(a_i, a_j)$  は、小世界ネットワーク [12] の概念を用いて設定した。小世界ネットワークとは、高度に構造化され、かつ任意のノード間の経路長が短いネットワークである。実世界では、人間のコミュニティやインターネットなどに見られる。マルチエージェントシステムは対象とする系を反映するため、エージェント間のインタラクションは小世界ネットワークの性質をもっているといえる。本シミュレーションでは、エージェントの隣接接点を 6 に設定し、隣接接点とのインタラクションコストを 1 とした正則なネットワークを構成し、確率  $p$  で枝を張り替えることで小世界ネットワークを構成した。

まず初期配置として、エージェントをクラスごとに分割し、各サーバに配置する。具体的には、 $s_i$  ( $i = 1, 2, \dots, m$ ) に存在するエージェント集合を  $\{a_j | (i-1)n/m + 1 \leq j \leq in/m\}$  とした。この初期配置は、エージェント移動のために適用する各アルゴリズムに対して共通である。次に、各サーバに対して負荷の平均値をランダム値で定める。そして、各エージェントには、サーバに設定した負荷の平均値にランダム値をかけたものを設定する。エージェントの計算負荷とインタラクションコストには相関がないものとした。本シミュレーションでは、 $T_h$  をエージェントの負荷の総和に 1.1 を掛けた値を設定した。エージェントの計算負荷を定めた後、計算負荷がしきい値  $T_h$  を超えたサーバに対して各手法を適用し、負荷の分散を

行う．そして，式 (11) によって与えられる移動後のシステム全体のインタラクションコストの変化を記録する．

$$\sum_{1 \leq i \leq m} \sum_{i < j \leq m} P(A_i, A_j) \quad (11)$$

以上の操作を 1 回の反復として，負荷分散によるエージェントの移動数と，システム全体のインタラクションコストの比較を行う．

エージェント移動の際に適用したアルゴリズムは，以下の 2 手法である．

- Sequentially Load Balancing
- Community-based Load Balancing

本シミュレーションでは，エージェントをコミュニティの評価に基づいて移動する CLB と比較するために，エージェント単体を評価し，移動を決定する Sequentially Load Balancing (SLB) との比較を行った．SLB の擬似コードを Algorithm 3 に示す．SLB は，式 (12) に基づいてエージェントを評価する．エージェントの評価値は，エージェントを移動することによるインタラクションコストの変化量である．

$$\begin{aligned} \text{gain}(a_i) = & \sum_{a_j \in A_1} p(a_i, a_j) \\ & - \max_{w_i + W(A_k) < T_h} \sum_{a_l \in A_k} p(a_i, a_l) \quad (12) \end{aligned}$$

SLB は，評価値が最も小さいエージェントを選択し，そのエージェントが最もインタラクションを行うサーバに移動する操作を，移動元のサーバの計算負荷がしきい値を下回り，かつインタラクションコストを改善するエージェントが見つからなくなるまで繰り返す．

### Algorithm 3 Sequentially Load Balancing

**Require:**  $W(A_1) > T_h$ ,  $W(A_i) < T_h$  ( $i = 2, 3, \dots, m$ ),  
 $\sum W(A_i) < mT_h$

**Ensure:**  $W(A_i) < T_h$  ( $i = 1, 2, \dots, m$ )

**repeat**

$\text{gain}_{\min} \leftarrow \infty$  : minimum *gain* in the following loop

$a_{\min} \leftarrow \text{null}$  : agent providing  $\text{gain}_{\min}$

**for**  $a_i \in A_1$  **do**

**if**  $\text{gain}_{\min} > \text{gain}(a_i)$  **then**

$a_{\min} \leftarrow a_i$

$\text{gain}_{\min} \leftarrow \text{gain}(a_i)$

**end if**

**end for**

move  $a_{\min}$  to  $s$  (server which  $a_{\min}$  moves to in  $\text{gain}_{\min}$ ) ( $A_1 \leftarrow A_1 \setminus \{a_{\min}\}$ ,  $A_j \leftarrow A_j \cup \{a_{\min}\}$ )

**until**  $W(A_1) < T_h$  and  $\text{gain}(a_i) \geq 0$  ( $\forall a_i \in A_1$ )

## 5.2 シミュレーション結果

図 3 は適用手法ごとにシミュレーションを行い，その結果を記録したグラフである．このグラフの縦軸はサーバ間で発生するインタラクションコストの総和であり，式 (11) で与えられる．横軸は，過負荷のサーバから移動を行ったエージェントの数を表す．このグラフ中には，SLB と CLB を適用した場合のインタラクションのコストを適用回数ごとに記録している．

CLB を適用することで，SLB を適用する場合よりも，インタラクションコストが小さな配置を行うことができる．具体的には，SLB を適用した場合のインタラクションコストの最大値は 168，平均値が 110.45 であるのに対して，CLB を適用した場合は最大値が 117，平均値が 79.77 であった．

次に，小世界ネットワークを構成する際の確率  $p$  を変動させ，手法ごとのインタラクションコストの比較を行った．本論文では，各設定に対して，6 回のシミュレーションを行った．そのシミュレーション結果を表 2 に示す． $I(p)$  は式 (13) で定義され，確率  $p$  で小世界を構成する際の，SLB の CLB に対する改善度を表す．式 (13) における Average interaction cost とは，エージェントの移動数が 10,000 から 20,000 までのインタラクションコストの平均値である．

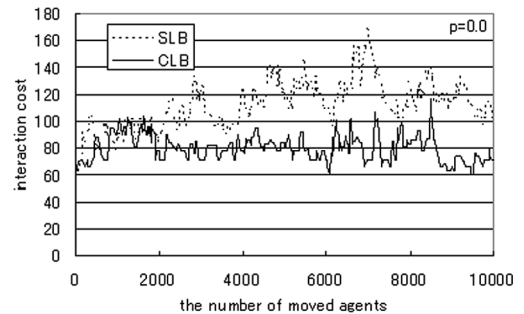


図 3 インタラクションコストの変化  
 Fig. 3 The change of the interaction cost.

表 2 平均インタラクションコストと改善度の比較  
 Table 2 The average interaction cost and improvement.

probability $p$	SLB	CLB	$I(p)$
0	110.946	80.880	0.27099
0.0001	111.296	84.373	0.24190
0.001	111.680	81.719	0.26827
0.01	129.515	105.013	0.18918
0.1	327.989	288.883	0.11923
1.0	1418.101	1422.055	-0.0027

$$I(p) = 1 - \frac{\text{Average interaction cost in CLB}}{\text{Average interaction cost in SLB}} \quad (13)$$

10,000 から 20,000 までの平均値をとった理由は、ある程度エージェントの移動を繰り返した後の、定常状態の平均値を測定するためである。

表 2 より、エージェントのインタラクションが構成するネットワークが正則、または小世界ネットワークの場合に、インタラクションコストはより改善する。改善度はネットワークが正則な場合に最も大きく、SLB のインタラクションコストを約 27%改善している。これは、ネットワークのクラスタリング係数が高いことで、移動元のエージェントにコミュニティが存在する確率が高くなり、CLB が SLB よりも効率的な配置を行うことができることを示している。

しかし、インタラクションコストが減少する一方で、エージェントの配置を決定するために必要となる計算量は大きくなる。ここでは、エージェントがインタラクションを行うエージェントの平均数を  $c$  とし、過負荷のサーバが計算負荷をしきい値以下にするために移動しなければならないエージェントの数を  $N$  とする。SLB では、各エージェントの評価値を求める計算量は  $c$  に比例し、 $s_1$  が保持するすべてのエージェントの評価値を求めるための計算量は  $O(c|A_i|)$  である。そして、最も評価値が小さいエージェントを選択して移動する操作を、移動元のサーバの負荷がしきい値を下回るまで繰り返すため、エージェントの移動に必要な計算量は  $O(Nc|A_i|)$  である。

次に CLB の計算時間を考える。まず、OptimalSet の操作に必要な計算量を  $S$  とする。CLB は各エージェントに対して OptimalSet を適用し、最も評価値が小さいコミュニティを選択して移動を行うため、エージェントの移動に必要な計算量は  $O(NS|A_1|)$  である。本論文で提案した OptimalSet の近似解法である ApproximateOptimalSet に必要な計算時間を求める。Candidate に含まれる 1 体のエージェントの評価値を求める計算時間は  $O(c)$  である。その操作を Candidate に含まれるエージェントに対して反復する。クラスタリング係数が高く、Candidate に含まれるエージェント数を  $O(c)$  とすると、1 体のエージェントを移動エージェント集合に追加するために必要な時間は  $O(c^2)$  である。その反復をコミュニティの移動先サーバがなくなるまで反復するため、その反復回数を

表 3 サーバ数を増加させた場合の、CLB の平均計算時間  
Table 3 The average computation time in changing the number of servers.

サーバ数 (台)	10	20	50	100
CLB の平均計算時間 (分)	0.03	0.05	0.09	0.16

表 4 サーバが保持するエージェント数を増加させた場合の、CLB の平均計算時間  
Table 4 The average computation time in changing the number of agents in a server.

サーバ 1 台のエージェント数 (体)	100	200	500	1000
CLB の平均計算時間 (分)	0.03	0.3	5.8	101

$O(N)$  とすると、総計算時間は  $O(Nc^2)$  となる。よって、CLB の総計算時間は  $O(N^2c^2|A_i|)$  である。

したがって、1 台のサーバが保持するエージェント数が増加するにつれて、 $|A_i|$  と  $N$  が増加し、CLB の計算量は大きく増加する。1 台のエージェントサーバに存在するエージェント数を 100 とし、サーバ数を増加させた場合の、CLB の平均計算時間を表 3 に示す。表 4 には、サーバ数を 10 台とし、サーバ 1 台当りに存在するエージェント数を増加させた場合の CLB の平均計算時間を示す。本計測は CPU 1.7 GHz、メモリ 512 MByte の計算機で、Java を用いて実装されたプログラム上でを行い、エージェントのインタラクションを正則ネットワークとして、エージェントの隣接接点数を 6 とした。これらの結果より、エージェント数に比例してサーバが存在する場合、CLB の計算量の増加は小さい。しかし、サーバ 1 台当りのエージェント数が増加するに従って、CLB の計算量は大きく増加する。

## 6. む す び

本研究では、分散マルチエージェントシステムにおいて、システムの性能を改善するためのエージェント配置に関する議論を行った。分散マルチエージェントシステムを構築する場合、エージェントの計算負荷とインタラクションコストを考慮して、その配置を決定する必要がある。

本研究の主な貢献は、インタラクションコストを小さくするエージェント配置手法である、Community-based Load Balancing (CLB) を提案したことである。CLB の特徴は、互いに頻りにインタラクションを行うエージェント集合であるコミュニティに注目し、コミュニティに対する評価を用いてエージェントの配置を決

定することである。本論文では、エージェント配置に関するシミュレーションを用いることで、エージェントのインタラクションが構成するネットワークが正則、または小世界ネットワークである場合に、エージェント単体に対する評価に基づいて配置を決定する SLB アルゴリズムよりも、インタラクションコストが小さくなることを示した。

一方で、この手法ではエージェントの移動による安定性や収束性を保証することはできない。安定性や収束性は、分散アルゴリズムにおける重要な性質である。こうした性質を保証するためにエージェントの移動時にサーバ間で協調を行う必要があるが、本研究ではその方法に言及していない。サーバ間でエージェントの配置を決定する際に、こうした性質を保証するための協調方法を検討する必要がある。

CLB の計算量も解決すべき問題の一つである。本論文では、エージェント集合に対する評価に基づいて配置を決定する基本的なアルゴリズムとして CLB を提案した。しかし、CLB はサーバ 1 台当りのエージェント数が増加するにつれて計算量が大きく増加するという問題がある。したがって、CLB をより効率的に実行するための手法を提案する必要がある。

謝辞 本研究は、日本学術振興会科学研究費基盤研究(A)(18200009, 2006-2008) 戦略的情報通信研究開発推進制度地域情報通信技術振興型研究開発(2005-2007)の助成を受けて行われた。

## 文 献

- [1] L. Gasser and K. Kakugawa, "Mace3j: Fast flexible distributed simulation of large, large-grain multi-agent systems," AAMAS, pp.745-752, 2002.
- [2] M. Balmer, N. Cetin, K. Nagel, and B. Raney, "Towards truly agent-based traffic and mobility simulations," AAMAS '04: Proc. Third International Joint Conference on Autonomous Agents and Multiagent Systems, pp.60-67, IEEE Computer Society, Washington, DC, USA, 2004.
- [3] H. Tai and G. Yamamoto, "An agent server for the next generation of web applications," DEXA '00: Proc. 11th International Workshop on Database and Expert Systems Applications, p.717, IEEE Computer Society, Washington, DC, USA, 2000.
- [4] T. Ishida, L. Gasser, and H. Nakashima, eds., "Massively multi-agent systems I," First International Workshop, MMAS 2004, Kyoto, Japan, Dec. 2004, Revised Selected and Invited Papers, vol.3446 of Lecture Notes in Computer Science, Springer, 2005.
- [5] I. Ahmad and A. Ghafoor, "Semi-distributed load balancing for massively parallel multicomputer systems," IEEE Trans. Softw. Eng., vol.17, no.10, pp.987-1004, 1991.

- [6] B. Hong and V.K. Prasanna, "Distributed adaptive task allocation in heterogeneous computing environments to maximize throughput," ipdps, vol.01, p.52b, 2004.
- [7] T.-H. Chia and S. Kannapan, "Strategically mobile agents," MA '97: Proc. First International Workshop on Mobile Agents, pp.149-161, London, UK, 1997.
- [8] T. Kawamura, S. Joseph, A. Ohsuya, and S. Honiden, "Quantitative evaluation of pairwise interactions between agents," ASA/MA 2000: Proc. Second International Symposium on Agent Systems and Applications and Fourth International Symposium on Mobile Agents, pp.192-205, London, UK, 2000.
- [9] 能登正人, 沼澤政信, 栗原正仁, "エージェントの移動性を考慮したエージェント間通信のトラフィック量に関する実験と評価," 電学論(C), vol.124-C, no.3, pp.904-911, 2004.
- [10] K.-P. Chow and Y.-K. Kwok, "On load balancing for distributed multiagent computing," IEEE Trans. Parallel Distrib. Syst., vol.13, no.8, pp.787-801, 2002.
- [11] M.-W. Jang and G. Agha, "Adaptive agent allocation for massively multi-agent applications," in T. Ishida, et al. [4], pp.25-39.
- [12] D.J. Watts and S.H. Strogatz, "Collective dynamics of 'small-world' networks," Nature, vol.393, pp.440-442, 1998.

(平成 18 年 5 月 19 日受付, 8 月 9 日再受付)



宮田 直輝

2006 京大・工・情報学卒, 現在, 同大学院社会情報学専攻修士課程に在学中。大規模マルチエージェントシステムに関心をもつ。



石田 亨 (正員)

1976 京大・工・情報工学卒, 1978 同大学院修士課程了。同年日本電信電話公社電気通信研究所入所。ミュンヘン工科大学, パリ第六大学, メリーランド大学客員教授など経験。工博。IEEE フェロー。情報処理学会フェロー。現在, 京都大学大学院情報学専攻教授, 上海交通大学客員教授。自律エージェントとマルチエージェントシステム, セマンティック Web 技術に取り組む。デジタルシティ, 異文化コラボレーションプロジェクトを推進。