Vol.2013-ICS-170 No.11
2013/3/12

リソース推定方法と役割学習を組み合わせたチーム編成の効率化について

早野 真史[1,a)]　浜田 大[1]　菅原 俊治[1,b)]

概要：本研究では，タスク割り当て問題をチーム編成問題と捉え，チーム編成時の自らの役割学習と他のエージェントのリソース・能力の推定を組合せた効率的な割り当て手法を提案する．一般に計算機やインターネットサービスを実現するタスクは異なる能力や機能を要求する複数の部分要素（サブタスク）で構成され，それら全てのサブタスクを実現して初めてそのサービスが提供される．そのため，迅速なサービスの提供には，それらのタスクを適切な能力をもつエージェントに割り当てる必要がある．これまでのチーム編成において，期待報酬の増加やチーム編成の成功率向上を目指して，各エージェントの役割を自律的に学習/決定する手法を提案してきた．この手法では，他のエージェントのリソースは既知と仮定している．しかし，インターネットなどの開放環境では，他のエージェントのリソースをすべて把握することは困難である．そこで本研究は，他のエージェントの能力の情報を仮定せず，代わりに実績からそのリソースを推定する手法と既存の役割学習を組み合わせたチーム編成手法を提案する．評価実験から，リソースの情報を既知とした既存手法と比較し，同等以上の効率が実現されたことを示す．

キーワード：マルチエージェントシステム、強化学習、資源割り当て問題

1. はじめに

近年，グリッドコンピューティング [1] やサービスコンピューティングのように，複数の計算機や，その上で動作するソフトウェアの組み合わせで提供されるサービスが注目されている．これらのサービスは複数の異なる部分的な要求（サービス要素）で構成されており，これら全ての要素を達成してはじめて 1 つのサービスとして提供される [2]．仮に 1 つのサービス要素でも処理が遅れたり，処理に失敗すると，サービス提供不可あるいは遅延を引き起こす．そのため，各要素を，それを実現するために必要な能力やリソースを持つ適切な計算機に割り当てる必要がある．

このような問題に対し，人工知能の分野では，各計算機あるいはその上で動作するソフトウェアを自律エージェントによるチーム編成問題としてモデル化することがある．ここでは，サービスを実現するタスクを実行するために必要なエージェント達をチームと考え，それを自律的な判断かつ動的に編成し，そのチーム内のエージェントにサービス要素に対応したサブタスクを割り当て，処理するモデルである [3]．

迅速かつ必要な質を満たすサービスを実現するには，このようなチーム編成において解決すべき課題がある．たとえば，各サブタスクを実行できる能力を持つエージェントを集め，効率的なチームを編成する必要がある．さらにサービスは，同時に多数要求されることもあるため，エージェントの能力が最大限に発揮でき，かつ重複しないようなタスクの割り当てを実現する必要がある

資源割り当て問題の観点からチーム編成に取り組む研究は多くある．たとえば [4] では，チーム編成のパレート最適解を求める多項式アルゴリズムを提案している．[5] では，エージェントの集合に遺伝的アルゴリズムを適用し，複数のタスクに対応する適切なチームを求める手法を提案した．これらの研究では，エージェントの状態とシステムに存在するタスクをあらかじめ把握している必要があるが，開放環境では要求されるタスクやエージェントの状態を予め予測することは不可能である．[6] では階層構造のエージェントネットワーク環境において強化学習を用いてタスクの割り当てを学習し，効率的にチーム編成を行う方法が提案されている．ここでは過去の経験に基き，階層の下のエージェントにタスクを効率的に割り当てることで，チーム全体として効率を実現した．[7] では [6] の方式を拡張し，学習の結果十分にタスクを割り当てられていないエージェン

[1] 早稲田大学
Waseda Uniersity
[a)] m.hayano@isl.cs.waseda.ac.jp
[b)] sugawara@waseda.jp

トにリンクを追加し，その能力を活用できるようにしている．しかし，どちらもネットワーク構造は階層的と仮定しているため，その応用は限られる．[8] では，全てのエージェントは過去の情報から，どのタスクが誰に提案され割り当てられたかという情報を学習し，その情報を元に，タスクの割り当てに適切と思われるメンバーを選択している．[9] では，[8] のモデルを改良し，エージェントにチーム編成時の役割やチームに加えるべき協調相手を自律的に学習させる方法を提案し，実際に [8] の手法よりも効率が上回ることを示している．しかし，[8] や [9] では他のエージェントのリソースの情報や他のエージェントのメッセージの内容を既知としている．しかし，前と同様インターネットなどの開放環境では，そのソフトウェアのバージョンアップや，ハードウェアとなる計算機の停止や取り換え・更新が非同期に行われる．また，エージェントは，異なる組織やプログラマに実装されることや，主体（人や企業）の目的に合わせて動作していることから，エージェントのリソースや行動決定を既知とできるとは限らない．

本研究では、[9] を拡張し，他のエージェントのリソースを未知としながらも，チーム編成時、特にメンバー選択の過程を工夫し、役割の学習と他のエージェントのリソースを学習を並行して行い，効率的なチーム編成手法の実現を目的とする。

本論文の構成は以下のとおりである．まず，次節でチーム編成の基本的なモデルを説明する．第 3 節では，エージェントの学習手法とリソースを推定する方法，さらにそれを用いたチーム編成の過程を説明する，第 4 節では評価実験を行いその結果について考察し，第 5 節で結論と今後の課題について述べる．

2. 問題設定とモデル

2.1 エージェントとタスク

エージェントの集合を $A = \{1, \ldots, n\}$ とおく．各エージェントは，自身の処理能力に相当したリソースを持つとし，エージェント i のリソースを $H_i = (h_i^1, \ldots, h_i^p)$ とおく．ここで，p はリソースの種類の数であり，h_i^k は非負の実数とする．タスク T は，それを構成するサブタスクの集合 S_T と，完了時に得られる報酬 $u_T \geq 0$ のペア $T = (S_T, u_T)$ で表す．ここで，$S_T = \{s_1, \ldots, s_l\}$ で，s_i は T を実現するためのサブタスクとする．

各サブタスクには，処理するために要求されるリソース $R_{s_i} = (r_{s_i}^1, \ldots, r_{s_i}^p)$ がある．エージェントがサブタスクを処理するためには，エージェントの持つリソースが対応したサブタスクの各リソースを，上回る必要がある．この条件は．エージェント i と，サブタスク s に関して，$h_i^k \geq r_s^k$ と表せる．また，エージェントは

$$h_i^k \geq \sum_{s \in S} r_s^k, \tag{1}$$

を満たせば，集合 S のサブタスク全てを同時に処理できるとするが，S はあるタスク T のサブタスクの部分集合，つまり $S \subset S_T$ とする．これは，i は同時に 2 つの異なるタスクのチームには参加できないとする．

エージェントは s を処理するとそれに対応した報酬を受け取る．本研究では，報酬はリソースの合計と仮定し，

$$u_s = \sum_{k=1}^{p} r_s^k \tag{2}$$

とする．タスク T の報酬 u_T は，S_T に含まれるサブタスクの報酬の和，

$$u_T = \sum_{s \in S} u_s \tag{3}$$

とする．

2.2 チーム編成

タスク T を実行するチームを (G, σ, T) と表す．ここで G はタスク T を処理するエージェントの集合を表す．σ は割り当て関数であり，$s \in T$ を $i \in G$ に割り当てることを $\sigma(s) = i \in G$ と表す．エージェントが次の式を満たすとき，チーム編成に成功したという．

$$h_i^k \geq \sum_{s \in \sigma^{-1}(i)} r_s^k \quad (1 \leq \forall k \leq p), \tag{4}$$

要求されるタスク T は，環境に用意されたキューで管理する．エージェントは，キューの先頭のタスクを参照し，リーダーかメンバーの役割を決める．ここで，リーダーは，参照したタスクに対して主導的にチームを作る役割をもち，メンバーは，自身に送られたチーム参加提案から，1 つ選択し，チームに参加する．

リーダーは，参照したタスク T に含まれるサブタスクのうち，処理可能となるものを自分に割り当て，その残りを処理するためにメンバーの候補を選択する．このとき各サブタスクごとに L 体のエージェントを選ぶ．ここで，L は正の整数とし，チーム参加依頼メッセージ重複度と呼ぶ．選択したメンバーの候補とリーダーの集合を仮チーム G^p とする．リーダーは G^p のメンバーに対してチーム参加提案メッセージを，担当させたいサブタスクとともに送り，その応答を待つ．

メッセージを受信したエージェントは，もしメンバーとしての役割を選択していれば参加するチームを決め，参加表明を行う．チーム参加提案メッセージに対して参加表明のメッセージを返したエージェントと，リーダーの集合を G とする．リーダーは G のメンバーに対して，サブタスクの割り当て σ を決める．具体的な σ の定め方は後で述べる．このようにして得られた (G, σ, T) が式 (4) を満たしたとき，チーム編成が成功したという．このとき，リーダーはメンバーに対してチーム編成の成功を通知しタスク

の処理を依頼する．したがって，チーム参加依頼メッセージ重複度 L が小さいと，あるサブタスクについて参加表明をするエージェントが存在しないことがある．L が大きければ，このようなチーム編成の失敗の確率は減るが，メッセージ数が増える．

チーム編成に成功した後，報酬を受け取る．報酬は，まずリーダーが全ての報酬から一定の割合を受け取り，メンバーはその残りの報酬から，自身の処理したサブタスクの要求リソース量の割合に応じたものを受け取る．もし，チーム編成に成功しなければ，タスクの処理が不可能になったとして，リーダーはチーム編成の失敗を G のメンバーに伝える．

メンバーの役割を選択したエージェントは，受信したチーム参加提案メッセージを参照し，その中に含まれるサブタスクが実行可能なものを選ぶ．そのようなメッセージが複数ある場合は，その中から最大の報酬を期待できるものを選択する．そのメッセージを送ってきたリーダーに，チームへの参加を表明し，それ以外のリーダーにはチームに不参加であることを伝える．

離散時間を導入し，その単位を tick とよぶ．タスクは 1 tick あたり平均 λ のポアソン分布に従い生成され，環境のキューに追加される．この他，メッセージの所要時間も 1 tick とする．

以上のモデルにおいて，本研究では，チーム編成の成功率を向上させる．このために，エージェント全体が受けた報酬の和でこれを評価をする．本研究では，各エージェントは利己的 (self-interested) と仮定しているため，それぞれは自分が得られる報酬を最大化するように行動するが，その判断により，チーム編成が効果的に行われ，その結果システム全体の効率が上がることを目的としている．

3. 提案手法

本章ではまず [9] で提案された学習パラメータについて簡単に説明する．次に，提案するリソース推定法と，それを用いたチームのメンバー決定方法について説明する．

3.1 欲張り度

チーム編成 (G, σ, T) が成功したとき，チーム全体として，タスク T の報酬 u_T を受け取り，リーダー i は，主導的にチーム編成を行った見返りにその一定の割合を自身の報酬とする．この割合を i の欲張り度といい，g_i と表す．したがって i の報酬 u_i は，$u_i = u_T \times g_i$ となる．ここで，T はチームで処理したタスクである．チームのメンバー $j \in G \setminus \{i\}$ への報酬は，処理したサブタスクが全タスクに占める割合の報酬として，以下のように分配する．

$$u_j = (u_T - u_i) \times \frac{\sum_{t \in \sigma^{-1}(j)} u_t}{\sum_{s \in T \setminus \sigma^{-1}(i)} u_s} \tag{5}$$

リーダーの割り当て分を引いてからメンバーの報酬を決めているため，欲張り度 g_i が大きいとメンバーへ分配する報酬は少なくなり，その結果チームへ参加するエージェントが減る可能性がある．

欲張り度 g_i は，チーム編成の成功・不成功の結果に応じて以下の式で更新する．

$$g_i = \alpha_g \times \delta_{success} + (1 - \alpha_g) \times g_i \tag{6}$$

ここで，チーム編成が成功した場合は $\delta_{success} = 1$ とし，失敗した場合は 0 とする．また，α_g の値は欲張り度の学習率であり，0 以上 1 以下の定数とする．

3.2 提案受託期待度

リーダー i が他のエージェント j にチーム参加提案メッセージを送ったとき，それが受託される期待度を提案受託期待度 $e_{i,j}$ と呼ぶ．この値が高ければ，メンバーへの参加の可能性が高いと判断し，優先的にチーム参加を提案をする．提案受託期待度 $e_{i,j}$ は以下の式で更新する．

$$e_{i,j} = \alpha_e \times \delta^j_{accept} + (1 - \alpha_e) \times e_{i,j}, \tag{7}$$

ここで，チーム参加提案メッセージが受託されれば，δ^j_{accept} は 1，拒否されれば 0 とする．α_e は学習率で，0 以上 1 以下の定数とする．割り当て関数 σ 決定時にもこの値の高いものを優先する．リーダーは，提案受託期待度 $e_{i,j}$ と，ε-greedy 戦略にもとづいてチーム参加提案メッセージを送るメンバーを決定する．これらについて，詳しくは第 3.5 節に述べる．

3.3 報酬期待度

報酬期待度とは，エージェントがメンバーとしてチームに参加表明したとき，リーダーから受け取る報酬の期待値のことである．メンバー i のリーダー j に対する報酬期待度を $d_{i,j}$ で表すと，i は受信したメッセージの中で，報酬の期待値が高いメッセージ $\tilde{m}$ を選択する．

$$\tilde{m} = \arg\max_{m \in M} \sum_{s \in S(m)} u_s \times d_{i,l(m)}, \tag{8}$$

ここで，$l(m)$ は，m を送信してきたリーダーを表す．$S(m)$ は，メンバー i が，受信したメッセージ m で要請されたタスク集合を表す．また，メンバーのチーム参加提案メッセージの選択には，ε-greedy 戦略を用る．

報酬期待度 $d_{i,j}$ は受け取った報酬にもとづき，次の式で更新する．

$$d_{i,j} = \alpha_d \times \frac{U}{\sum_{s \in S(\tilde{m})} u_s} + (1 - \alpha_d) \times d_{i,j} \tag{9}$$

ここで，U はサブタスクを処理したことで，実際に得られた報酬である．$\sum_{s \in S(\tilde{m})} u_s$ は，受託したメッセージ $\tilde{m}$ に含まれるサブタスクの報酬の和である．α_d は，報酬期待度

の学習率で 0 以上 1 以下の定数とする．もし，チーム編成に失敗した場合は，タスクを割り当てられないため，U の値は 0 とする．

3.4 役割の選択

リーダーかメンバーの役割の選択を，欲張り度と報酬期待度から決定する．エージェント i は，タスクキューの先頭のタスクを参照し，リーダーとしてもらえる報酬見込み E_i^{leader} と，メンバーとして貰える報酬見込み E_i^{member} を調べる．E_i^{leader} と E_i^{member} はそれぞれ以下の式で求める．

$$E_i^{leader} = \sum_{s \in T} u_s \times g_i \tag{10}$$

$$E_i^{member} = \sum_{s \in S(\tilde{m})} u_s \times d_{i,l(\tilde{m})} \tag{11}$$

エージェントは $E_i^{leader} \geq E_i^{member}$ であればリーダーを選択し，その他の場合はメンバーの役割を選択する．なお，エージェントがチーム参加提案メッセージを受け取っていない場合は，E_i^{member} の値は 0 となる．

3.5 リソース推定方法とメンバー決定方法

3.5.1 リソース推定方法

本研究では，他のエージェントのリソースを未知とするが，代わりにその量を推定する手法を導入する．このため，エージェント i は他のエージェント j に対してリソース推定パラメータ $\tilde{H}_j = (\tilde{h}_j^1, \tilde{h}_j^2, \ldots, \tilde{h}_j^p)$ をもつ．リソース推定パラメータは初期状態では 0 で，チーム参加提案メッセージの返事を受けて更新する．リーダー i がサブタスク s を割り当てるために，j を選択したとする．

(1) i は j にタスク s とともにチーム参加提案メッセージを送る
(2) j がそれに対し受託の返事をしたとき，i はタスクのリソース R_s と推定リソース $\tilde{H}_j$ の各要素を比較し，$r_s^k \geq \tilde{h}_j^k$ ならば，$\tilde{h}_j^k \leftarrow r_s^k$ とする
(3) j が受託を拒否したとき，推定リソース $\tilde{H}_j$ は更新しない．

これらの値は仮チーム編成をする際に用いられるが，その部分は後述する．

3.5.2 仮チームのメンバー決定方法

リーダー i は次のように仮チームのメンバーを決定する．リーダーが処理するサブタスクを除いた残りのサブタスクの集合を S'_T とする．$s \in S'_T$ をこの中で，もっとも報酬の大きなサブタスクとする．リーダーは，提案受託期待度 $e_{i,j}$ の大きい順にそのリソース推定値 $\tilde{H}_j$ と s の要求リソースの各要素を調べ，$r_s^k \leq \tilde{h}_j^k$ $(1 \leq \forall k \leq p)$ のとき，s の担当として j を選ぶ．もし，この条件を満たさなければ，j の次に大きい提案受託期待度を持つエージェントに移る．このようにして s の担当として，L 体のエージェント選択する．もし，このようなエージェントがいなければランダムに s の担当メンバーを決定し，L 体とする．s のメンバーの選択が完了したら，$S'_T = S'_T \setminus \{s\}$ として，上記を繰り返す．ただし，ε の確率で上記のリソース推定値を用いずに S'_T のサブタスクの担当をランダムに L 体定める．アルゴリズムの詳細を図 1 に示す．

Require: T: タスク
  i: リーダー
  L: 各サブタスク毎に選択されるえージェト数
  $S \subset T$: i に割り当てられたサブタスク
  $S'_T = T \setminus S$; $\sigma^p = \emptyset$; $G^p = \emptyset$
  S'_T の要素 s を報酬値にもとづいてソートする．
  for all $s \in T$ **do**
    $l(s) = 0$; $K = K_i \setminus \{i\}$
    for $j \in K$:$e_{i,j}$ を用いて選択する **do**
      if $\tilde{h}_j^k \geq r_s^k$ for $\forall k$ **then**
        $\sigma^p = \sigma^p \cup \{(s, j)\}$ // s を j へ割り当てる
        $G^p = G^p \cup \{j\}$
        $\tilde{h}_j^k = \tilde{h}_j^k - r_s^k$ for $\forall k$
        $l(s) = l(s) + 1$; $K = K \setminus \{j\}$
        if $l(s) \geq L$ **then**
          break
        end if
      end if
    end for
  end for
  for all $s \in T$ が $l(s) < L$ である場合 **do**
    $L - l(s)$ エージェントを K_i からランダムに選択する
    $\sigma^p = \sigma^p \cup \{(s, a'_1), \ldots, (s, a_{L-l(s)})\}$
    $G^p = G^p \cup \{a'_1, \ldots, a_{L-l(s)}\}$
  end for

図 1 提案手法における仮チームのメンバー選択方法
Fig. 1 Determination of Pre-Team Members

3.5.3 タスク割り当てについて

仮チーム G^p のメンバーが決まると，リーダーはチーム参加提案メッセージを送り，メンバーはその返事をする．G^p において，チーム参加提案メッセージを受託したメンバーの集まりを G^0 とする．リーダーは，G^0 の中から，サブタスクを割り当てるメンバーを決定し，σ を定義するが，1 つのサブタスクにつき L だけエージェントを選択し，チーム参加提案メッセージを送っているため，複数の返事が返る場合がある．その場合は，ε-greedy 戦略を用いて，提案受託期待度の大きいメンバーにタスクを割り当てる．ここで，サブタスクをエージェントに割り当て終えたあともサブタスクが残っていた場合，リーダーは，G^0 のリソースに余裕があるメンバーに割り当てる．全てのサブタスクを割り当てることができ，式 (4) を満たすことができたとき，チーム編成が成功となる．

4. 評価実験と考察

4.1 実験環境

提案手法の有効性を調べるための評価実験を行う．本実験では，1 つのサブタスクにつき選択するエージェントの数

表 1 実験におけるエージェント
Table 1 Agemt

パラメータ	値
エージェントの数 $\|A\|$	50
エージェントのリソースの種類数 p	2
エージェントの各リソース量 h_j^k	3 〜 12 のランダム

表 2 実験におけるタスク
Table 2 Task

パラメータ	値
1tick の平均タスク発生数 λ	2(ポアソン分布)
$s \in S_T$ の各リソース量 r_j^k	1 〜 8 のランダム
$\|S_T\|$	3 〜 7 のランダム

表 3 学習パラメータ初期値
Table 3 Initial Values

パラメータ	初期値
欲張り度	0〜1 のランダム
提案受託期待度	0.5
報酬期待度	0.5

表 4 学習率
Table 4 Learning Parameters

パラメータ	学習率
α_g	0.1
α_e	0.05
α_d	0.05

L(チーム参加依頼メッセージ重複度) を 2 とし，$\varepsilon = 0.05$ とする．その他の各種パラメータの設定を表 1〜4 に示す．

比較手法として以下の 3 つを用いた．

既存手法

[9] で提案された手法を既存手法とする．仮チームのメンバーを選択する際，エージェントのリソースは既知としており．リソースを把握した状態でメンバーを決定できる．第 3 節で述べた，欲張り度，提案受託期待度，報酬期待度の学習は行う．

ランダム手法

ランダム手法とは，[9] において，欲張り度，提案受託期待度，報酬期待度の学習はしない，ただし，全エージェントのリソースは既知としているため，メンバーの選択は，対応するサブタスクの処理が可能なものに限る．

Contract Net Protocol(CNP)[10]

本研究の仮定と同様にリソースを既知としないネゴシエーションプロトコルとして CNP と比較する．この場合，全てのエージェントにチーム参加提案メッセージを広報する．ここでもエージェントは利己的，つまり，受信したエージェントは，自分が処理可能なもののうち最大の報酬が得られるものを選び，それに入札する．CNP ではリーダーに相当するマネージャーは固定であり，その数を 5 とした．これはマネージャー数を変えた時に得られる報酬が最大となる値である．

4.2 実験結果

50 tick 毎に得られた総報酬を記録し，それを 30000 tick まで繰り返した．結果を図 2 に示す．

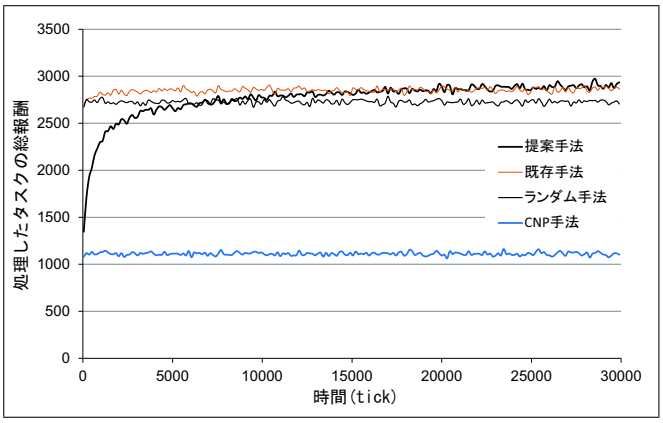

図 2 50 ターンごとの処理したタスクの総報酬の推移
Fig. 2 Transition in numbers of received utilities.

5000 tick まで，処理したタスクの総報酬が急激に増加するが，これはリソースの推定と役割学習によりチーム編成の効率化が達成されたことを示している．さらに，30000 tick 時では，既存手法をやや上回った結果を得ている．

4.3 考察

提案手法は，推定リソースを更新するために仮チーム編成のときに常に ε-greedy 手法を用いる．つまり，5 %の確率で推定リソースを考慮せず，ランダムにエージェントの選択を行う．そのため，図 2 における結果が既存手法よりも 5 %ほど低くなることが考えられた．しかし，最終的な結果はわずかであるが 1.3 %ほど良くなった．これは，次のような理由が推測される．提案手法は，ε-greedy 手法によって，リソースの推定が進んだ場合でも，一定の確率で，ランダムにエージェントを選択する．このとき，提案受託期待度，リソース推定パラメータを考慮したときに，既存手法で選択されないメンバーを選択する場合がある．例えば，それが孤立したエージェントであれば，そのエージェントをメンバーとして，チームで活用できるようになったのではないかと思われる．ただし，詳しい実験が必要である．

提案手法とランダム手法を比較すると，提案手法がやや上回る．これは [9] でも述べている通り，欲張り度，提案受託期待度，報酬期待度の学習によってこのような差が出たことが本研究でも同様に示された．

しかし，CNP は他の手法に比べて非常に低い結果となった．これは，CNP が全てのエージェントに広報メッセージを送ることに起因すると考えられる．エージェントは全てのリーダーからチーム参加提案メッセージを受け取るが，このとき，大きいリソースを持ったエージェントは，より多くの報酬を期待できるチームに参加を試みる．その結果，入札が集中する．このため，能力があるエージェントにタスクが割り当てられず，多くのタスクの処理ができなくなったためと考えられる．CNP では，エージェント

の能力など明確に差があることを想定しており，その状態で集中が起こらないことを仮定している．本実験のように似たエージェントが競合する環境では，十分な効率が得られないことを示している．

5. 結論

本研究では，他のエージェントのリソースを未知とする条件で，効率的なチーム編成を実現する手法を提案した．これは [9] で提案したエージェントの自律的な役割学習により，チーム編成を効率化する手法に加え，リソース推定方法を加えた手法である．実験から，提案手法はリソース情報を未知としたにもかかわらず，既存手法をやや上回る結果を示した．

今後は，既存手法を上回る結果を得た理由を調査するとともに，学習の収束率を向上させ，エージェントの数を増やした大規模な環境にも対応させる．

参考文献

[1] F. Berman, G. Fox, and A.J.G. Hey. *Grid Computing: Making the Global Infrastructure a Reality*. Wiley series on parallel and distributed computing. Wiley, 2003.

[2] Tapia D.I. Bajo J. Corchado, J.M. A multi-agent architecture for distributed services and applications. *Int. Journal of Innovative Computing, Information and Control*, pp. 2453–2476, 2012.

[3] Michael N. Huhns, Munindar P. Singh, Mark Burstein, Keith Decker, Edmund Durfee, Tim Finin, Les Gasser, Hrishikesh Goradia, Nick Jennings, Kiran Lakkaraju, Hideyuki Nakashima, H. Van Dyke Parunak, Jeffrey S. Rosenschein, Alicia Ruvinsky, Gita Sukthankar, Samarth Swarup, Katia Sycara, Milind Tambe, Tom Wagner, and Laura Zavala. Research directions for service-oriented multiagent systems. *IEEE Internet Computing*, Vol. 9, No. 6, pp. 65–70, 2005.

[4] Onn Shehory and Sarit Kraus. Feasible formation of coalitions among autonomous agents in non-super-additive environments. *Computational Intelligence*, Vol. 15, No. 3, 1999.

[5] Jingan Yang and Zhenghu Luo. Coalition formation mechanism in multi-agent systems based on genetic algorithms. *Appl. Soft Comput.*, Vol. 7, No. 2, pp. 561–568, 2007.

[6] Sherief Abdallah and Victor Lesser. Organization-based cooperative coalition formation. In *Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, IAT '04, pp. 162–168, Washington, DC, USA, 2004.

[7] Ryota Katayanagi and Toshiharu Sugawara. Efficient team formation based on learning and reorganization and influence of communication delay. In *Proceedings of the 2011 IEEE 11th International Conference on Computer and Information Technology*, CIT '11, pp. 563–570, Washington, DC, USA, 2011.

[8] Thomas Genin and Samir Aknine. Coalition formation strategies for self-interested agents in task oriented domains. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, Vol. 2, pp. 205–212, 2010.

[9] Dai Hamada and Toshiharu Sugawara. Deciding roles for efficient team formation by parameter learning. In *Proceedings of the 6th KES international conference on Agent and Multi-Agent Systems: technologies and applications*, KES-AMSTA'12, pp. 544–553. Springer-Verlag, 2012.

[10] R. G. SMITH. The contract net protocol: High-level communication and control in a destributed problem solver. *IEEE Transactions on Computers*, Vol. 29, No. 12, pp. 1104–1113, 1980.