# 強化学習とリンクの動的生成を用いた組織の再構成によるチーム編成の効率化

## Effective Team Formation using Reinforcement Learning and Dynamic Reorganization

片柳　亮太　　菅原　俊治
Ryota Katayanagi　　Toshiharu Sugawara

早稲田大学基幹理工学研究科情報理工学専攻
Department of Computer Science and Engineering, Waseda University

We propose a novel and effective method using reinforcement learning and dynamic reorganization for the team formation in multi-agent systems (MAS). A task in MAS usually consists of a number of subtasks that require their own skills, thus they have to be processed in the appropriate team whose agents have the sufficient skills. However, the skills required for tasks and how agents should be organized in MAS is often unknown in advance, thus their organization should be adopted according to the environment where agents are deployed. In this paper, we try to improve the success rate of team formation in the given organizational environment using reinforcement learning. We will show that this method can improve the overall utilities and reduce the failure of team formation.

## 1. はじめに

近年，コンピュータ・ネットワーク上のデータが爆発的に増加している [1]．これに伴い，ネットワーク上の処理負荷も増加傾向にあるため，グリッド・コンピューティング等の分散コンピューティング手法が着目されている．このようなシステムには，

- システム内の計算機の処理性能に個体差がある
- システム全体が一度に処理できるデータ量には限度がある

といった特徴や限界がある．また，システム内の各計算機は与えられた役割に対応したソフトウェアがインストールされ，それに応じた個別の機能を持つ．

一般に，分散環境におけるタスクは複数のサブタスクから構成され，各サブタスクは異なる機能やリソースを要求する．そのため，タスクを処理するには，そのタスクを構成するサブタスクをすべて処理できる計算機のチームを適切に構成し，各計算機にサブタスクを適切に割当てる必要がある．１つでも処理が遅れたり，サブタスクの割当てが行われないと，タスクの遅延，あるいは失敗といった結果をもたらす．従って，これらの条件下でシステムあるいはエージェントが構成する組織において，タスクをいかに効率的に処理するかが重要な課題となる．

[2] では，分散コンピューティング環境をマルチエージェントシステムととらえ，そこでの効率的なタスク処理のために，計算機に自律学習を行わせる手法を提案し，シミュレーションにより評価を行っている．しかし，エージェント全体の能力を考慮するとその能力が十分に引き出されてはいない．本研究では，組織構造の再構成に着目し，一定期間ごとにエージェント間のリンクを動的に生成することで，与えられたタスクをより効率的に処理できるチーム編成を行うアルゴリズムを提案し，本アルゴリズムが全体の効用に与える影響をシミュレーションによって調査する．

## 2. 問題設定

本論文で扱う問題を定式化するため，[2] に基づいて以下の定義を導入する．時間は離散時間とし，その最小単位を「エピソード」と呼ぶ．エピソードの集合は $\mathbf{E} = \{E_1, E_2, ..., E_o\}$ と表す．ここで，$o$ はエピソードの数を表す．

本研究では，初期実験の位置づけから，タスクを処理するエージェントの組織として階層構造を仮定する．階層構造は，効率面から優れた性質を持ち，ある程度以上の規模では，組織がこのような構造を持つことは多い．但し，インターネットのアプリケーションでは，より一般的な構造を持つことがあるが，本論文では対象外とする．組織の例を図 1 に示す．図 1 における葉はタスクを処理する実行エージェントを，根や節はタスクの割当てやチーム編成を行うマネージャを表す（後述）．

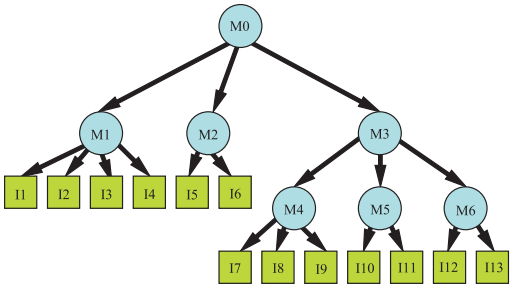



図 1: 階層構造の組織例

環境に存在するタスクの集合 $\mathbf{T}$ を $\mathbf{T} = \{T_1, T_2, ..., T_p\}$ とする．ただし，$p$ は環境に存在するタスクの数を表す．任意のタスク $T_i \in \mathbf{T}$ は，複数のサブタスクによって構成され，$T_i = \{t_1^i, t_2^i, ..., t_q^i\}$ と表せる．ただし，$q$ は $T_i$ に含まれるサブタスクの数を表す．また，任意のタスク $T_i \in \mathbf{T}$ のサブタスク $t_j^i$ は $t_j^i = \{u_j^i, rr_{j1}^i, rr_{j2}^i, ..., rr_{jm}^i\}$ と表す．ここで，$u_j^i$ は $t_j^i$ が組織に与える効用，$rr_{j_k}^i$ は $t_j^i$ が処理に要求するリソース $k$ の量とする．ここで，$T_i \in \mathbf{T}$ は，$T_i$ にチームが割当てられた時に $T_i$ が組織に与える効用の総量 $U_i$ と $T_i$ が要求するリソース $k$ の総量 $RR_k^i$ を用いて $T_i = \{U_i, RR_1^i, RR_2^i, ..., RR_m^i\}$ と表す．$U_i$ と $RR_k^i$ はそれぞれ $U_i = \sum_{t_j^i \in T_i} u_j^i$，$RR_k^i = \sum_{t_j^i \in T_i} rr_{j_k}^i$ で表せる．

組織内の任意のエージェント $A_i$ は，$A_i$ より組織の階層構造の下位にある各エージェントの情報を保持する．エージェ

連絡先: 片柳　亮太，r.katayanagi@isl.cs.waseda.ac.jp

ントは組織内に２種類存在する．階層組織において根や節に位置するエージェントをマネージャと呼び，その集合 **M** を $\mathbf{M} = \{M_1, M_2, ..., M_n\}$ とする．ただし，$n$ は組織におけるマネージャの数を表す．組織に属する任意のマネージャ $M_i$ は受け取ったタスク $T_k$ を下位のエージェントへ受け渡す役割を持つ．その際，$M_i$ は受け渡し先のエージェントに対して適切なサブタスク $t_j^k$ を $T_k$ から分割して渡す．また，階層組織において根に位置するマネージャ $RM \in \mathbf{M}$ を特にルートマネージャと呼ぶ．

組織において葉に位置するエージェントを実行エージェントと呼び，その集合 $\mathbf{I} = \{I_1, I_2, ..., I_l\}$ とする．ただし，$l$ は組織における実行エージェントの数を表す．組織に属する任意の実行エージェント $I_i$ は，マネージャから与えられたサブタスクを処理する役割を持ち，その処理能力は管理するリソースの量に依存する．このことから，$I_i \in \mathbf{I}$ は，$I_i$ が管理するリソース $k$ の総量 $cr_k^i$ を用いて，$I_i = \{cr_1^i, cr_2^i, ..., cr_m^i\}$ と表す．ここで，$m$ は環境下に存在するリソースの種類を表す．本研究において，組織内の任意のエージェント $A_i$ の下位は，全てマネージャであるか，全て実行エージェントであるかの２つの場合のみを想定する．なお，先に述べたリソースについて，本研究においては環境下に存在するリソースの種類を２種類として問題を扱うこととする．

本研究で扱うチームは次のように定義する．あるタスクを実行するために，サブタスクを割当てられたエージェントの集合をチームと呼び，チームの集合 **C** は $\mathbf{C} = \{C_1, C_2, ..., C_{|S|}\}$ と表す．チーム $C_i$ にタスク $T_i \in \mathbf{T}$ が割当てられると，組織に効用 $U_i$ が与えられる．もしタスクの割当てに失敗した場合，$T_i$ は廃棄される．１つのチームに属したエージェントは，そのタスクを構成するサブタスク全ての処理が終わるまで他のチームには属せない．また，サブタスクを処理するエージェントが１体でもかけるとチームは作れず，タスク処理は失敗となる．

以上の定義のもとで，エピソードは次のように進行する．まず，各エピソードにおいて，ルートマネージャ $RM$ は環境から複数のタスクを受け取る．次に，$RM$ を含め，任意のタスク $T_j$ を受け取った任意のマネージャ $M_i \in \mathbf{M}$ は，$T_j$ を下位のエージェントに対して適切なサブタスク $t_{j_k} \in T_j$ に分割し渡す．その後，サブタスク $t_{j_k}$ は最終的に組織の葉に位置する任意の実行エージェント $I_i$ に渡され，$I_i$ は与えられた $t_{j_k}$ の処理を行う．最終的に，$RM$ が環境から受け取った全タスクの処理が終わった時点でエピソード終了となり，各エージェントは次のタスクを受け入れる準備に入る．本研究におけるチーム形成問題は，上記の定義のもと，

1. $\sum_{i|T_i \in \mathbf{S}} U_i$ ができるだけ大きな値をとること
2. チームの集合 **C** において，$C_i \in \mathbf{I}$ がタスク $T_i$ に割当てられるために，以下の条件
   (a) $\forall T_i \subseteq \mathbf{S}$
   (b) $\forall k : \sum_{I_i \in C_i} (cr_{i,k} \times \beta) \geq RR_{i,k}$（$\beta$ は定数）
   (c) $\forall i \neq j : C_i \cap C_j = \emptyset$
   を満たすこと

以上の２条件を満たすタスクの集合 $\mathbf{S} \subseteq \mathbf{T}$ と，**S** を実行するチームの集合 **C** を求めることである．

## 3. 提案手法

本研究では，任意のマネージャ $M_i$ の置かれた状態をニューラル・ネットワークにより表現し，その情報をもとにマネージャに強化学習を行わせることで，タスクの受け渡しを近似的に最適化する．そして，効率的なタスクへのリソースの割当てを実現するとともに，定期的にリンクの動的生成を行うことで，より効率的なチーム編成が行えるよう組織の再構成を行う．以下に，本手法の概要を示す．

### 3.1 Q 学習アルゴリズムによる強化学習

本手法では，マネージャによる下位のエージェントへのタスクの受け渡しについて，Q 学習 [3] を用いた学習を行う．エージェントの学習を表す式は次のようになる．

$$Q(s,a) \leftarrow Q(s,a) + \alpha[r + \gamma max_{a'} Q(s',a') - Q(s,a)]$$

ここで，$Q(s,a)$ は，状態 $s$ で行動 $a$ を取ることの $Q$ 値を表す．$max_{a'} Q(s',a')$ は，状態 $s$ で行動 $a$ を取った結果遷移した状態 $s'$ で，最も高い $Q$ 値を持つ行動 $a'$ を取るときの $Q$ 値である．また，$\alpha$ と $\gamma$ はいずれも 0 より大きく 1 より小さい定数で，学習の割引率を表す．上の式を用いて $Q$ 値を更新していくと，いずれの状態における $Q$ 値もある値に収束する．これによりマネージャは，タスクに対して適切に処理を行うと期待できるエージェントへタスクを受け渡すようになる．

### 3.2 ニューラル・ネットワークによる状態の抽象化

上記で述べた強化学習を実現するには，状態を定式的に表現する必要がある．本研究では，ニューラル・ネットワークを用いて状態表現を抽象化することで，少ない情報量で状態空間を表現している．本研究におけるニューラル・ネットワークを用いた状態表現の概要を図 2 に示す．

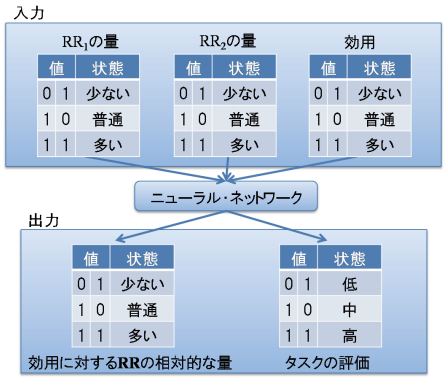



図 2: ニューラル・ネットワークの入力値と出力値

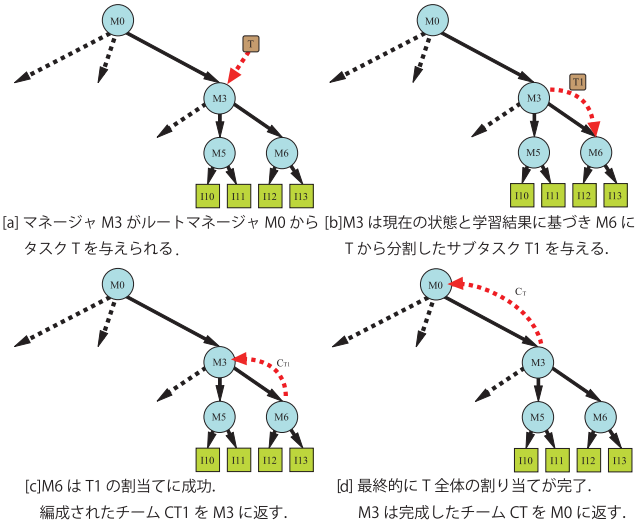
本研究ではニューラル・ネットワークを，タスクの要求する２種類のリソースの量とチーム編成時に得られる効用を各２ビットで表現した値を与えると，効用に対する要求リソースの相対量と，タスクの評価を各々２ビットで表現した値が得られるように訓練する．入出力値は人為的に対応づける．例えば ($RR_1$ の量:$RR_2$ の量:効用 = 少ない:普通:高い) という入力に対し，ニューラル・ネットワークは (効用に対する **RR** の量:タスクの評価 = 少ない:高い) と出力するように訓練を行う．

### 3.3 チーム編成

本研究では，以下の手順によってチーム編成を行う．マネージャはタスクを受け取ると，それまでの学習結果に応じて，タスクを渡すべきエージェントを決定する．次に，そのエージェントの能力に応じてタスクを適切なサブタスクに分割し渡す．マネージャが組織の根から葉に向かって上記の処理を繰り返すことで，最終的には組織の葉に位置する実行エージェントにサ

ブタスクが割当てられる．これらの実行エージェントにより構成されるのがチームである．チームに属する実行エージェントは，タスク全体の処理が終わるまで他のチームに所属することができない．そこで，限られたエージェント数でさらに多くのタスクを処理させる必要がある．マネージャが強化学習によりタスクを渡すべきエージェントを選ぶことで，タスクを処理する効果的なチームの編成を行う [2].

上記の手法に則ったチーム編成の例を図 3 に示す．まず，組織内のマネージャ$M3$ はタスク $T$ を受け取ると，現在の自身の状態をもとに，サブタスクを渡すマネージャを決定する．ここでは下位のマネージャの処理能力から，$T$ を構成するサブタスク $t_1$ が $M6$ に対して適当だと判断し，$T$ から分割して $M6$ に渡すと仮定する．続いて，$M6$ は自身の管理する実行エージェントの能力を見て，処理できるエージェントが存在することを確認し，そのエージェントにタスクを渡す．$M3$ に対しては，編成されたチーム $C_{t_1}$ を返す．このような処理を繰り返し，最終的に $M3$ はチーム編成に成功したら，得られたチーム $C_T$ を，タスク $T$ を渡したルートマネージャ$M0$ に返して処理を終了する．




[a] マネージャ M3 がルートマネージャ M0 から タスク T を与えられる．
[b]M3 は現在の状態と学習結果に基づき M6 に T から分割したサブタスク T1 を与える．
[c]M6 は T1 の割当てに成功. 編成されたチーム CT1 を M3 に返す．
[d] 最終的に T 全体の割り当てが完了. M3 は完成したチーム CT を M0 に返す．

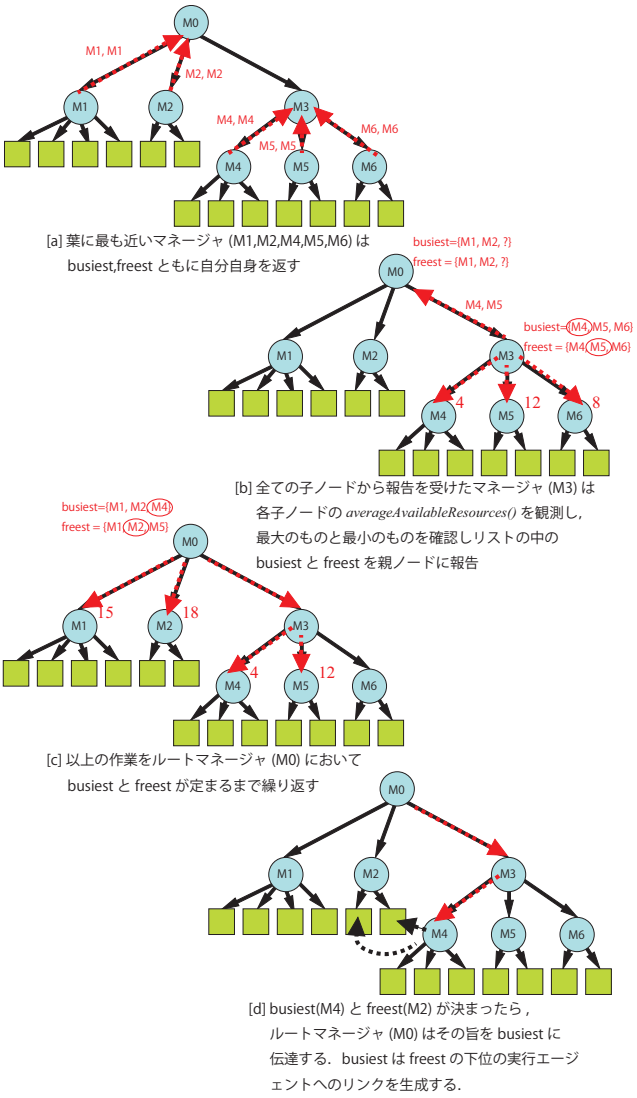
図 3: チーム編成の例

### 3.4 リンクの動的生成を用いた組織の再構成

学習が進むと，エージェントはタスクを渡されやすいものとそうでないものに分かれる．このとき，タスクを渡されやすいものは利用可能なリソース量が平均して少なく，逆にタスクをあまり渡されないものは利用可能なリソース量が平均して多くなる．しかし，各エージェントが管理するリソース量には限界があり，最終的にタスクが廃棄されるなど無駄が生じる．

そこで，本研究では，一定回数のタスクの処理を終えるたびに，組織において実行エージェントの親となるマネージャの中で，利用可能なリソース量の平均が最も少ないもの ($busiest$) と最も多いもの ($freest$) を確認し，$busiest$ から $freest$ の管理する実行エージェントへのリンクを生成する．これにより，タスクを多く渡されるマネージャが利用可能なリソースが増え，タスクの廃棄などの無駄が減少する．

リンクの動的生成の例を図 4 に示す．まず，実行エージェントを直接管理するマネージャたちは自分自身を $busiest$，また $freest$ として返す．図では $M1, M2, M4, M5, M6$ がこれに当たり，図 4[a] において，各マネージャは上位のマネージャに自身




[a] 葉に最も近いマネージャ (M1,M2,M4,M5,M6) は busiest,freest ともに自分自身を返す
[b] 全ての子ノードから報告を受けたマネージャ (M3) は 各子ノードの *averageAvailableResources()* を観測し，最大のものと最小のものを確認しリストの中の busiest と freest を親ノードに報告
[c] 以上の作業をルートマネージャ (M0) において busiest と freest が定まるまで繰り返す
[d] busiest(M4) と freest(M2) が決まったら，ルートマネージャ (M0) はその旨を busiest に伝達する．busiest は freest の下位の実行エージェントへのリンクを生成する．

図 4: リンクの動的生成の様子

を $busiest$ あるいは $freest$ として報告している．報告を受けた上位のマネージャは，その値を配列 $busyList$ や配列 $freeList$ に追加し，下位マネージャからの報告を待つ．図では $M0, M3$ がこの役割を負っている．$M0$ は $M3$ の報告を待つため，$M3$ が先に次の処理を行う．$busyList$ や $freeList$ が完成したマネージャは，リストに含まれるマネージャの中で平均して利用可能なリソースの量が最も低いもの（$busiest$ の場合）あるいは最も高いもの（$freest$ の場合）を選び，上位のマネージャに報告する．図 4[b] においては，$M3$ が完成した $busyList, freeList$ の中から，$busiest = M4, freest = M5$ を確認し，上位のマネージャである $M0$ に報告している．上記のような作業はルートマネージャの $busyList$ や $freeList$ が完成するまで繰り返される．図 4[c] においては $M0$ の $busyList$ と $freeList$ が完成し，組織全体を見た時の $busiest = M4, freest = M2$ であることがわかる．最終的に，ルートマネージャは $busiest$ となったマネージャと $freest$ となったマネージャにその旨を告げ，$busiest$ と $freest$ の下位の実行エージェントとの間に新たにリンクを生成させる．図 4[d] がそれに相当し，$M4$ は $M2$ の管理する実行エージェントに対して新たにリンクを生成する．

# 4. 実験・結果

## 4.1 実験内容

本論文では，図 5 のような構造において，

- 本提案手法
- [2] による手法
- 学習なし (ランダム)

の 3 つのチーム編成手法による効用値の推移を調べ，性能の比較をした．

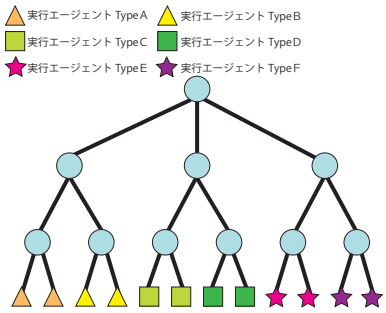



図 5: 実験に用いた組織

図 5 に示す通り，実行エージェントに以下の 6 種類を用意した．種類に応じて利用できるリソース量が異なる．

- **Type A** $cr_A = \{cr_0^A = 2, cr_1^A = 2\}$
- **Type B** $cr_B = \{cr_0^B = 10, cr_1^B = 10\}$
- **Type C** $cr_C = \{cr_0^C = 0, cr_1^C = 30\}$
- **Type D** $cr_D = \{cr_0^D = 1, cr_1^D = 10\}$
- **Type E** $cr_E = \{cr_0^E = 20, cr_1^E = 2\}$
- **Type F** $cr_F = \{cr_0^F = 8, cr_1^F = 0\}$

また，タスクとして 3 種類用意し，6 体のエージェントが協力することで処理できるようにパラメータを設定した．種類により要求するリソースの総量と得られる効用に差がある．3 種類のタスクのパラメータは次の通りである．タスクはエピソード開始時にこれらの中からランダムに選ばれ，複数個生成される．

- **Type A** $T_A = \{U_A = 60, RR_0^A = 48, RR_1^A = 12\}$
- **Type B** $T_B = \{U_B = 86, RR_0^B = 44, RR_1^B = 42\}$
- **Type C** $T_C = \{U_C = 68, RR_0^C = 14, RR_1^C = 54\}$

シミュレーションでは，7 つのタスクを組織の根のマネージャに与えてから 7 つ全ての処理が完了するまでを 1 エピソードとし，30000 エピソードを 1 セットとして 10 セット行い，その平均値を求めた．

## 4.2 結果

本実験の結果を図 6 と表 1 に示す．図 6 に示された通り，今回の条件下においては本研究の提案手法が [2] の手法と比較しておよそ 3 倍の効用を得ている．また，[2] の手法は，学習なし手法と比較して，平均効用では優れた結果を示すものの，表 1 から分かるように，廃棄率に関しては学習なし手法より劣った結果となる．[2] の手法はこのように効用と廃棄率のトレードオフ関係があるのに対し，提案手法はいずれも優れた結果を示している．[2] は少ないタスクで効用を上げる効果があるのに対し，本提案手法はこれに加え，処理できるタスクの量を増加させ，最終的な効用を上げている．

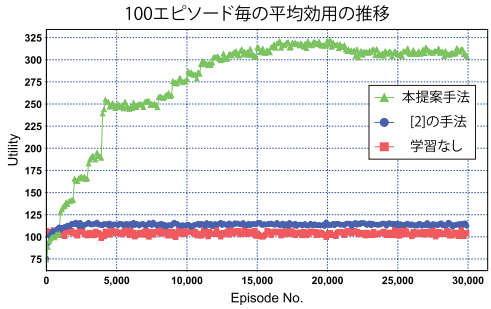



図 6: 平均効用の推移

表 1: タスクの廃棄率

| 手法 | 本提案手法 | [2] の手法 | 学習なし |
|---|---|---|---|
| 廃棄率 | 0.29 | 0.41 | 0.36 |

# 5. 結論

本研究では，システム内のコンピュータの性能をより高く引き出すことで分散処理の効率化を目指し，強化学習と組織構造の再構成によってそれを実現するアルゴリズムを提案した．シミュレーションを用いて，平均効用とタスクの廃棄率について，提案手法が [2] の手法や学習なし手法より優れていることが分かった．

本研究で得られた結果は，あくまでごく一部の状況における結果だけしか保証されていない．そのため，今後は多様な組織構造のもとで評価・検証する必要がある．特に，本論文では階層構造を仮定したが，より一般には，階層構造のグループが何らかのネットワーク構造でつながったもの，あるいはランダムネットワークやスケールフリーネットワークの構造をもつものもある．このような構造における効率化についても調査を行う予定である．また，研究の背景に計算機ネットワーク環境があるものの，現実のネットワークには必ず存在するコミュニケーション（通信）のコストを本研究では考慮していないため，コミュニケーションコストを考慮に入れたアルゴリズムを考案し，評価していく必要がある．

# 参考文献

[1] 総務省．平成２０年版 情報通信白書．

[2] S.Abdallah and V.Lesser. Organization-Based Cooperative Coalition Formation. In *Intelligent Agent Technology 2004.* pages 162-168.

[3] 三上 貞義，皆川 雅章 共訳，強化学習 第１版第５刷，2003.