Vol.2011-MPS-85 No.28
2011/9/16

強化学習を用いたチーム編成の効率化モデルの提案と環境変化に対する評価

佐藤　大樹[†1,†2]　菅原　俊治[†1,†3]

インターネット上のサービスに対応したタスクは，それを構成する複数のサブタスクを処理することで達成される．効率的なタスク処理のためには，サブタスクを対応する能力やリソースを持つエージェントに適切に割り当てる必要がある．我々はこれまで，強化学習とネットワーク構造の再構成により，チーム編成を効率化する手法を提案してきた．しかし，そこで用いた学習は，近隣のエージェントの内部状態を既知としており，必ずしも現実のシステムと合致しない．また，実験で仮定したエージェントの配置も固定的であった．本論文では，提案手法を，他のエージェントの内部状態ではなく，近隣からのメッセージと遅延を考慮した減衰率から報酬を求め，それに基づいて Q 学習するようにモデル化する．次に，初期配置によらず，学習と組織構造の変化を組み合わせ既存手法よりも効率化できることを示す．さらに，タスクの種類の変化についても，効率的なチーム編成が可能なことを実験により評価する．

Efficient Team Formation based on Learning and Reorganization and Influence of Change of Tasks

Daiki Satoh[†1,†2] and Sugawara Toshiharu[†1,†3]

A task in a distributed environment is usually achieved by doing a number of subtasks that require different functions and resources. These subtasks have to be processed cooperatively in the appropriate team of agents that have the required functions with sufficient resources. We showed that the proposed method combines the learning for team formation and reorganization in a way that is adaptive to the environment and that it can improve the overall performance and increase the success in communication delay that may change dynamically. But the machine learning that we used there knows inside state of neighborhood agents and this cannot be assumed in real systems. We propose a method of distributed team formation that uses modified Q-learning with reward based in messages from neighborhood and communication delay. We show that it can improve the overall performance in any initial placement and in environment change of range of task.

1. 序　　論

近年，グリッドやクラウドに代表されるように，インターネット上の多様なリソースとそれを活用したサービスの利用が爆発的に増加している．これらは，サービスにあわせてネットワークに分散した多種多様なリソースを適切に選択し，協調・統合させて実現できるサービス要素の集まりとして実現できる．これらのサービス要素を実現する計算機はそれぞれ異なる処理性能を持ち，そこで処理できるデータ量にも限度がある．また，各計算機は役割に応じたソフトウェアがインストールされ，それぞれ個別の機能を持つ．システムは大量に要求されるサービスを処理するために，各計算機の役割や処理能力に応じてタスクを適切に割当てる必要がある．

サービスは複数のサービス要素で構成されるが，それに対応してサービスを実現するタスクは，各サービス要素を実現するサブタスクの集合としてモデル化できる．したがって，タスクは，機能と負荷状態等に基づいて全てのサブタスクについて，それぞれを処理できる計算機が決定されたとき処理可能となる．しかし 1 つのサブタスクでも処理が遅れたり，リソースの割当てに失敗すると，タスクの遅延，あるいは失敗といった結果をもたらす．さらに，計算機間の通信には無視できない遅延も存在する．このため，システム全体の状況をタイムリーに示す正確な情報を取得することはできない．そのため，計算機が持つ部分的な情報に基づく制御が必要となる．これらの条件の下，計算機に適切な組織構造を導入し，そこで効果的にタスクに応じたチームを作り，能力と状態に合わせて，タスクを適切に割当てることが重要となる．

この課題に対し，3) では，計算機のネットワーク構造を (以下，3) にならい，計算機がなす組織構造をネットワークと捉え，単にネットワークと呼ぶ) 階層構造とし，効率的なタスク処理のために，直下の計算機への適切なタスクの割当てを過去の履歴から強化学習し，効率化する手法を提案している．しかし，この手法には，学習が進むにつれて特定の計算機に頻繁にタスクが渡され，リソースの利用に偏りが生じ，結果としてシステム全体のリソースを十分に活用できないという問題がある．また，6) では，タスク処理をチーム編成問題と捉

†1 早稲田大学 基幹理工学研究科 情報理工学専攻
〒 169-8555 東京都新宿区大久保 3-4-1
†2 E-mail:satoh@toki.waseda.jp
†3 E-mail:sugawara@waseda.jp

Vol.2011-MPS-85 No.28
2011/9/16

え，タスクとリソースの割当ての最適解を求めている．しかし，この手法は指数オーダーであり，現実に応用するのは難しい．また 3) の問題のために，2) ではタスクへのリソース割り当ての学習に加え，そのリソース割り当ての効率化とリソースを最大限に活用できるネットワーク構造への変化を同時に行う手法を提案している．さらに 1) では通信遅延のモデルを導入し，その大きさにあわせてネットワーク構造を変化させることを示した．しかし，要求されるタスクの変化への追従については述べられていない．またそこで用いられる遅延を考慮した学習手法は，複数の計算機の間でそれぞれの状態を共有しており，現実のシステムを考えると不自然である．またその学習手法についても，詳しく述べられていなかった．

そこで本論文では第一に，1) の手法で用いられている強化学習を，行動をタスクの分割と隣接するエージェントへの依頼，複数のエージェントにまたがる報酬の伝搬，として考え，メッセージと自己の状態のみから学習を行うものとして定式化する．さらに，報酬の割引には，エージェントのなすネットワーク間の行為の系列の結果である，遅延の影響を加味したものとして提案する．また，1)2) では，固定したエージェントの配置を初期状態として実験を行っていたが，これを任意のエージェントの配置に置き換え，そこからでも組織ネットワーク構造を適切に変化させ，効率的なチーム編成が可能なことを示す．さらに，タスクの種類といった環境の変化についても，3) などの従来手法より効率的なチーム編成が可能なことを実験により示す．

本論文の構成は以下の通りである．まず，次の節で，エージェントとタスクをモデル化し，本論文で扱う課題を明記する．第 3 節では，提案するエージェント間にまたがる強化学習について定式化し，具体的な学習法について述べる．第 4 節では，実験を通して，提案手法が初期状態によらず既存手法よりも効率化できること，またタスクの種類といった環境の変化についても効率的なチーム編成が可能であることを示す．第 5 節では，研究のまとめと今後の研究課題を述べる．

2. 問 題 定 義

2.1 エージェントとタスクのモデル

計算機ネットワークにおいて，計算機上のソフトウェア（これをエージェントと呼ぶ）が 1 回行動する最小単位時間をエピソードと呼ぶ．エピソードの集合を $\mathbf{E} = \{e_1, e_2, \cdots\}$ と表す．エピソードは e_1 から順に進行する．1 エピソードは $10msec$ を想定する．タスクを処理するネットワークは 3) に基づき階層構造を初期状態と仮定する．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の組織がこのような構造を持つことは多い．階層構造の例を図 1 に示す．図 1 中の節や葉は各計算機上で動作するエージェントを表す．

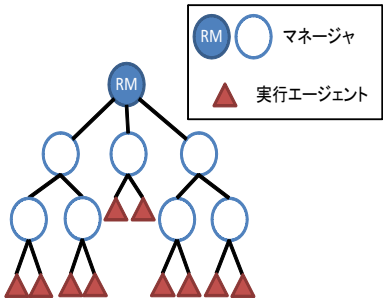

図 1 階層構造の例
Fig. 1 Example of a hierarchical structure.

ネットワーク内のエージェントは，階層構造の直下にあるエージェントが保持するリソース量の大小や，チームを編成中のエージェントのリストのみを保持する．エージェントにはマネージャ M_i と実行エージェント I_l の 2 種類を仮定する．マネージャは階層構造の根や節 (葉を除く) に位置するエージェントであり，直下のエージェントに，タスクを構成するサブタスクの部分集合 (詳細は以下で述べる) を割当てる．根に位置するマネージャを特にルートマネージャ(RM) と呼ぶ．実行エージェントは階層構造の葉に位置するエージェントであり，実際にタスクを処理する．なお，提案手法では，第 3.3 節で説明するリンクの動的生成を行う．それに伴い，エージェント間のネットワークは初期の木構造から変化し，エージェントは階層構造において直上の計算機を複数持つ可能性がある．

一般にネットワークにおいて，通信には遅延が生じる．ここで通信とは，エージェント間のタスクの受け渡しやその成否の報告，リンクの動的生成に必要な情報交換など，エージェント間のあらゆる通信を指す．エージェントは階層構造のネットワーク上で，隣接するエージェントとのみ通信を行う．その際にエージェントが観測できる遅延に関する情報は，隣接するエージェントから応答があるまでのラウンドトリップタイム (RTT) d のみである．RTT のうち，通信遅延が実際にいくつかを知ることはできない．そのため，本モデルにおいて計算機は d の観測値 $Obs(d)$ に基づき通信遅延時間を推測する．なお本論文で述べる実験では簡易化のため隣接する計算機間の RTT の値 d を一定とする．なお，エピソードと対応させ，$d = 1$ は $10msec$ を想定する．

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，$T_i = \{t_1^i, t_2^i, \cdots, t_q^i\}$ と表す．T_i

のサブタスク t_j^i を $t_j^i = \langle u_j, rr_1^j, rr_2^j, \cdots, rr_m^j \rangle$ と表す．ここで，u_j は t_j^i が保持する効用，rr_k^j は t_j^i が要求するリソース k の量とする．また，T_i は，T_i が保持する効用の総量 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 $T_i = \langle U_i, RR_1^i, RR_2^i, \cdots, RR_m^i \rangle$ と表す．ただし，$U_i = \sum_{t_j^i \in T_i} u_j$，$RR_k^i = \sum_{t_j^i \in T_i} rr_k^j$ である．

エピソードごとに，タスクの集合 $\mathbf{T}$ が RM に与えられ，RM は直ちにタスクを 1 つずつ処理する．RM はタスクを構成するサブタスクの集合を第 3.1 節で述べる学習に基づいて部分集合に分け，それぞれを直下の計算機に割当てる．マネージャ M_i は直上の RM あるいはマネージャからサブタスクの部分集合を受け取ると，必要であればさらにそれを分割し，直下の計算機に割当てる．最終的に，実行エージェントは直上のマネージャからサブタスクを割当てられると，保持するリソースをタスクの処理に割り当てる．このとき，当該実行エージェントは対応するタスクのチームに属したとみなす．

実行エージェント I_l は保持するリソース k の総量 cr_k^l を用いて，$I_l = \langle cr_1^l, cr_2^l, \cdots, cr_m^l \rangle$ と表す．あるチームに属する実行エージェントは，同一エピソード中に他のチームには属さないものとする．もし実行エージェント I_l があるチームに属する間に，直上のマネージャ M_i から別のサブタスクの部分集合を渡されると，I_l は M_i に，実行不可であることを報告し，その部分集合を廃棄する．実行エージェントが保持するリソース量は 1 エピソード当たりの処理能力を表す．サブタスク $t_i = \langle u_i, rr_1^i, rr_2^i, \cdots \rangle$ を実行エージェント $I_l = \langle cr_1^l, cr_2^l, \cdots \rangle$ に割当てると，その処理には次の式で表すエピソード数 $D_{t_i}^{I_l}$ がかかるものとする．

$$D_{t_i}^{I_l} = \lceil \max(\frac{rr_1^i}{cr_1^l}, \cdots, \frac{rr_m^i}{cr_m^l}) \rceil$$

ここで $\lceil\ \rceil$ は天井関数であり，また便宜的に $cr_j^l = 0$ のときは次のようにする．

$$\frac{rr_j^l}{cr_j^l} = \infty \ (\text{if } rr_j^l \neq 0), \frac{rr_j^l}{cr_j^l} = 0 \ (\text{if } rr_j^l = 0)$$

なお，以下の実験ではマネージャは 10 エピソード以内に t_i を処理できる実行エージェント I_l を見つけ，それにサブタスクを割当てている．

タスク $T_i = \{t_1^i, t_2^i, \cdots, t_q^i\}$ の全サブタスクが実行エージェントに割当てられた場合，そのタスクのチームが編成されたと言い，T_i の処理は成功とする．タスクの処理に成功すると，計算機のネットワークには T_i が保持する効用 U_i が与えられる．サブタスクのうち，1 つでも実行エージェントが割当てられないと，チームは編成されず，T_i の処理は失敗とする．サブタスクに実行エージェントが割当てられないのは，マネージャの直下に十分なリソースを持つ実行エージェントが 1 体もいない場合か，既に別のチームを編成している場合である．タスクの処理に失敗すると，マネージャはこれを廃棄する．

あるタスク集合 $\mathbf{T}$ に対し，チームの集合 $\mathbf{C}$ を $\mathbf{C} = \{C_1, C_2, \cdots, C_{|\mathbf{T}'|}\}$ と表す．ただし，$\mathbf{T}'(\subset \mathbf{T})$ はチーム編成に成功したタスクの集合とする．本研究におけるチーム編成の効率化とは，上記の定義のもと，$\sum_{i|T_i \in \mathbf{T}'} U_i$ をできるだけ大きくすることである．

3. 提案手法

提案手法では，任意のマネージャ M_i の置かれた状態を抽象表現し，それを基に M_i が Q 学習を行い，効率的なリソース割当てを実現する．さらに，一定エピソード毎にリンクを動的に生成し，チーム編成の効率化が行えるよう組織を再構成する．以下に提案手法の概要を示す．

3.1 タスク割当てに関する強化学習の定式化

提案手法において，全てのマネージャは階層構造の直下の計算機へタスクを割当てるときに Q 学習を行う．第一にこの Q 学習について，報酬の伝搬とその遅延の影響を複数エージェントにまたがる強化学習として，定式化を行う．

マネージャ M_i が，エピソード e においてタスク T_i のサブタスク t_j^i を割当てられると，M_i が置かれた状態 S_e^i(エピソード e におけるマネージャ M_i の状態とする．第 3.2 節にて詳細を述べる) により，直下のエージェントを 1 つ選び，それにサブタスク t_j^i を割り当てる．マネージャ M_i の直下のマネージャ群を $M_{children}^i = \{M_1^i, M_2^i, \cdots\}$ で表すとき (直下が実行エージェント群の場合は $I_{children}^i = \{I_1^i, I_2^i, \cdots\}$ と表す)，受け渡し先のマネージャを $M_c^i \in M_{children}^i$ とし，マネージャ M_i によるマネージャ M_c^i へのサブタスク t_j^i の割り当てを行動 $a_{S_e^i}^c = \langle M_c^i, t_j^i \rangle$ と表す．受け渡し先のエージェントの決定は行動戦略の問題であり，本手法では状態 S_e^i において行動 $a_{S_e^i}^c$ をしたときの Q 値 $Q(S_e^i, a_{S_e^i}^c)$ の 3 乗のルーレット選択により決定する．

マネージャ M_i が直下の実行エージェントにタスクを割り当て，その後他のマネージャエージェントの行動からリソースの割り当てに成功すると，そのサブタスクの効用 u_j を用いて，報酬 $r = u_j \beta^{Obs(d)}$ を獲得し，以下の式で Q 値を更新する．

$$Q(S, a) \leftarrow Q(S, a) + \alpha[u_j \beta^{Obs(d)} - Q(S, a)]$$

なお，リソース割り当て失敗のときは $r = 0$ で Q 値を更新する．この式はマネージャは他のマネージャの状態について未知とし，タスクの効用及び遅延時間のみで Q 値を更新することを反映した．ここで，$S = S_e^i$，$a = a_{S_e^i}^c$ とする．また，α は学習率であり，$0 < \alpha < 1$，β は遅延時間が獲得報酬に与える減衰率であり，$0 < \beta < 1$ を満たす．$Obs(d)$ は M_i がタ

Vol.2011-MPS-85 No.28
2011/9/16

スクを割り当ててからその成否を配下の実行エージェントから受け取るまでの観測時間であり，実行エージェントとの距離によりマネージャが受けとる報酬は減衰するものとして定式化した．また，木構造の中間のマネージャの報酬は自分が割当てられたサブタスクのみを対象としており，ネットワークが全てのサブタスクの割当てに成功して得る効用とは異なることに注意されたい．すべてのサブタスクが実行エージェントに割り当てられ RM がその報告を受けるとチーム編成成功となり，ネットワークはタスク T_i の効用 U_i を獲得する．

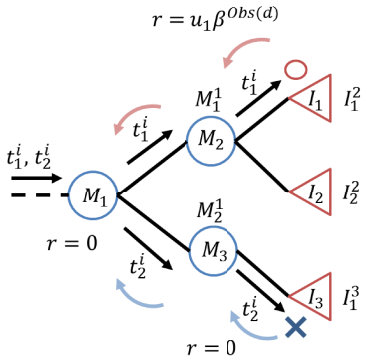

図 2 複数エージェントによるタスクの割り当てと報酬の受け取り
Fig. 2 Task allocations and reward gains among agents.

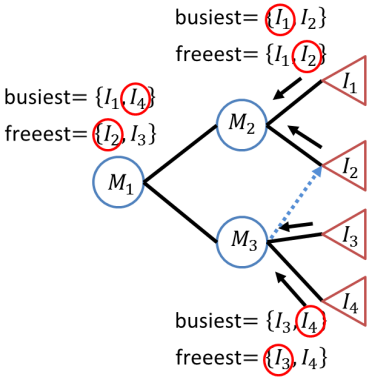

図 3 リンクの動的生成の様子
Fig. 3 An operation of a dynamic linking.

例えば，図 2 のように，M_1 が直上のマネージャから 2 つのサブタスク t_1^i, t_2^i を受け取り，M_1 は自身の状態により，M_2 へ t_1^i，M_3 へ t_2^i を割り当てたとする．同様に，I_1 に t_1^i が割り当てられ，リソース割り当てに成功し，I_3 に t_2^i が割り当てられ，リソース割り当てに一部でも失敗したとき，マネージャ M_1，M_2，M_3 が学習時に伝搬により受け取る報酬はそれぞれ $u_i\beta^{Obs(d)}$，0，0 となる．M_1 から M_2 や M_3 の状態は見えないが，割り当て結果により互いに連携することで報酬が上がるように学習する．また複数のエージェントから構成されているため，エージェント i から他のエージェント j にタスクを割り当てる行動 $a^c_{S_e^i}$ の後の j の状態遷移は観測できない．エージェントでは割り当てを行ったエージェントからのメッセージに基づいて報酬を求めている．またこのために割引率は導入せず，代わりに β がその役割を果たしている．

3.2 状態の抽象表現

Q 学習の効率化のために，計算機の状態空間を単純化する．本研究では，エピソード e におけるマネージャ M_i の状態 S_e^i は，M_i が直上マネージャから割当てられたサブタスク t_j^i の要求リソース量 $rr^j = \langle rr_1^j, rr_2^j, \cdots, rr_m^j \rangle$ と，t_j^i の効用 u_j によって単純に決まるものとする．すなわち，まず一つ目に，3 段階で評価された rr^j の量（種類別）と 3 段階で評価された u_j の量に基づいた，u_j に対する rr^j の量の相対的な多寡と，二つ目に，最終的な t_j^i に対する評価を 3 段階で生成し，これらの組合せを状態 S_e^i と定義する．例えば，t_j^i について，リソースの種類を 2 種類と仮定し，$(rr_1^j, rr_2^j, u_j) =$ (少ない , 普通 , 多い) のとき，状態は (u_j に対する rr^j の相対的な多寡 , t_j^i の評価) = (少ない , 多い) のように単純に 2 変数の組み合わせで表す．詳細については 2) に述べている．

3.3 リンクの動的生成を用いた組織の再構成

学習が進むと，能力とタスクが要求するリソースに応じてタスクを割当てられやすいエージェントとそうでないものに分かれ，リソースの残量に偏りが起こる．逆にリソースの残りが少ない実行エージェント側に多くのタスクが振り分けられ，結果として，タスク割当の失敗や，一部のリソースが使われずに残るという無駄が発生する．これは，エージェントの配置が，あらかじめタスクの構造を意識して作ったものではないからである．またデザイン時に要求タスクの構造や数を予測することもできない．このため提案手法では学習結果に基づくリンクの動的生成を用いて要求タスクにあわせてネットワーク構造を変更させる．

ネットワーク上の実行エージェントで，一定エピソード毎に利用可能なリソース量の平均が最も少ないものを *busiest*，最も多いものを *freest* と呼び，*busiest* の直上のマネージャから *freest* へのリンクを生成する．ただし既にリンクを張られているマネージャと *freest* 間のリンクの動的生成は行わない．また，*busiest* の直上のマネージャが複数存在するときは，管理するリソース量の平均が最も少ないマネージャと *freest* 間にリンクを生成する．また，実行エージェントのリンク数に上限値を導入し，この値を超えてはリンクを生成しないものとする．生成可能なリンク数の設定や，リンクの生成によりリンク数が設定を超える場合にどうするかは戦略切替の問題であり，本論文とは異なる観点からの議論が必要である．

実際には図 3 のように，規定エピソード毎に各実行エージェントが直上のマネージャに利用可能なリソースの平均値を報告し，直上のマネージャがそれを比較し部分的な *busiest*, *freest* を決定する．これを再帰的に行い，RM がネットワーク全体における *busiest*, *freest* を決定する．それを RM は中間マネージャを通して *busiest* の直上のマネージャに報告し，直上のマネージャから *freeset* へのリンク生成が行われる．なお，この間の情報の伝達にも

通信遅延が発生することに注意されたい．

4. 実験と考察

提案手法が 3) の手法 (以下，既存手法と呼ぶ) より，高い効用を獲得することを示している 1)2)．ここで既存手法とは，提案手法においてリンクの動的生成を行わず，第 3.1 節で述べた Q 値の更新時に，報酬に遅延を考慮せず，

$$Q(S,a) \leftarrow Q(S,a) + \alpha[u_j + max_{a'}Q(S',a') - Q(S,a)]$$

上記の更新式を用いた手法に対応する．ここで，S' は S の次の状態，a' は a の次の行動を表し，隣接するマネージャの状態を用いていることに注意されたい．しかし 1) では，実験の初期状態で実行エージェントの配置は固定であった．そこで，ここでは本論文で提案した学習モデルに基づいた Q 学習を実装し，一般的な初期状態から提案手法により効率化可能であることを示す．またタスクの種類が変化したとき，提案手法が環境に適応可能なことを実験的に示す．

4.1 実験環境

先に述べた 2 つの実験について，共通の環境を以下に示す．環境下に存在するリソースは 2 種類に設定し，実行エージェントとタスクのパラメータは次のようにした．実行エージェントは保持するリソースの量によって A〜F の 6 種類に分けた．各々のパラメータは $A = \langle cr_0^A = 2, cr_1^A = 2\rangle$，$B = \langle cr_0^B = 10, cr_1^B = 10\rangle$，$C = \langle cr_0^C = 0, cr_1^C = 30\rangle$，$D = \langle cr_0^D = 1, cr_1^D = 10\rangle$，$E = \langle cr_0^E = 20, cr_1^E = 2\rangle$，$F = \langle cr_0^F = 8, cr_1^F = 0\rangle$ とした．実行エージェントが保持できるリンク数の上限は 10 とした．また，強化学習における Q 値の更新式ではそれぞれ $\alpha = 0.001$，$\beta = 0.9$ を用いた．タスクは要求するリソースの量と効用によって T_1,T_2 の 2 種類を定義した．各タスクのパラメータは次の通りである．

- **タスク** $T_1 = \langle U_1 = 54, RR_0^1 = 12, RR_1^1 = 42\rangle = \{t_1^1, t_2^1, t_3^1\}$
  $= \{\langle u_1 = 4, r_0^1 = 2, r_1^1 = 2\rangle, \langle u_2 = 20, r_0^2 = 10, r_1^2 = 10\rangle, \langle u_3 = 30, r_0^3 = 0, r_1^3 = 30\rangle\}$
- **タスク** $T_2 = \langle U_2 = 41, RR_0^2 = 29, RR_1^2 = 12\rangle = \{t_1^2, t_2^2, t_3^2\}$
  $= \{\langle u_1 = 11, r_0^1 = 1, r_1^1 = 10\rangle, \langle u_2 = 22, r_0^2 = 20, r_1^2 = 2\rangle, \langle u_3 = 8, r_0^3 = 8, r_1^3 = 0\rangle\}$

タスク T_1 は実行エージェント A, B, C が，タスク T_2 は実行エージェント D, E, F がチームを編成することで処理できるようにし，チーム編成手法の性能による差が明確となるように設定した．なお，タスクはエピソード開始時に RM に与えられるが，タスクの種類や量は実験により異なる．

ネットワーク構造は，遅延の効果を明確にするために図 4 のような非対称な構造とした．また，上述の 6 種類の実行エージェントそれぞれ 2 体ずつ生成し，葉マネージャの直下にランダムに配置した．また，ネットワークの通信遅延は 1 エピソードした．これは，第 3 章で述べたよう，あらゆる情報がネットワークのリンク 1 つを伝搬するのに 1 エピソード経過することを表す．

各実験とも，30000 エピソードを 1 セットとして，ネットワークが 100 エピソードあたりに獲得した効用の推移をデータとして取得した．取得したデータは 100 セットの平均値である．なお，提案手法においては 1000 エピソードごとにリンクの動的生成を行った．

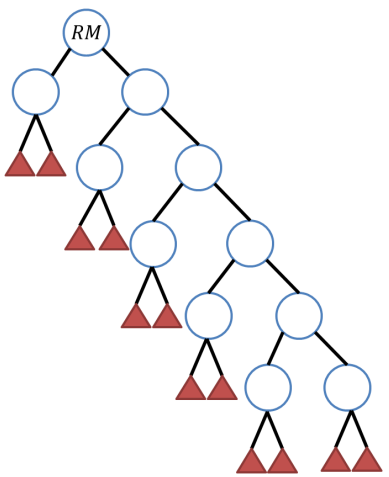

図 4 ネットワーク構造
Fig. 4 An agent network.

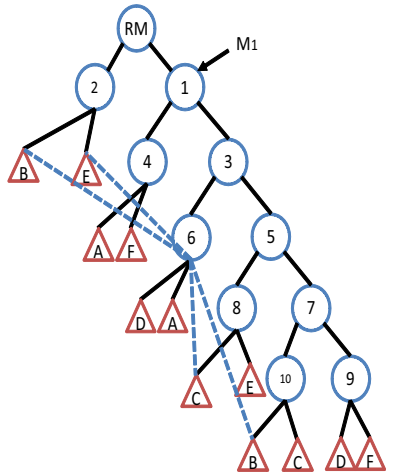

図 5 エピソード 5000 時のネットワーク構造
Fig. 5 A network at 5000 episodes.

4.2 一般環境における提案手法の獲得効用

初期の実行エージェントの配置がランダムである場合の，一定時間あたりの獲得効用の推移を調査した．比較として，既存手法とランダムにタスクを割り当てるランダム手法（リンクの動的生成や学習を行わない）を採用した．タスクは 1 エピソード毎に $T_1, T2$ を 2 つずつ RM に与えた．100 エピソードあたりの獲得効用値の推移を図 6 に示す．

図 6 から，提案手法の獲得効用が一番高く，最終的に既存手法の 150%程度の効用を獲得できたことが分かる．一例として提案手法における 5000 エピソード時のネットワークを図 5 に示す．破線は動的に生成されたリンクを表す．これから 5000 エピソード時点で動的リ

Vol.2011-MPS-85 No.28
2011/9/16

ンクの生成により 4 本の追加リンクが張られており，1000 エピソードから 5000 エピソードまで 1000 エピソード毎に毎回リンクの動的生成がされることが分かる．すなわち，提案手法では図 6 の提案手法の獲得効用値が急激に上昇した 2000 エピソードといった早期に新しいリンクが生成され，その後の獲得効用値の上昇が達成されている．実行エージェントの配置の初期状態にかかわらず，提案手法においてリンクの動的生成が効果的に作用すると考えられる．また，図 5 から，RM からの距離が大きく，リソースが大きい実行エージェント B や実行エージェント C が，RM により近いマネージャ M_6 へとリンクを張っており，初期状態より効率的なチーム編成が可能なネットワーク構造へ変化している．このことは，遅延を含む強化学習とリンクの動的生成が，自律的な組織の再構成に効果的に作用することを示している．組織構造を変化し遅延を含めた学習をすることで，初期状態に依存せずに既存手法より効率的なチーム編成が可能といえる．

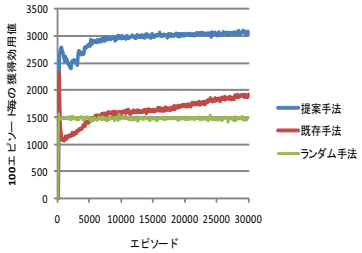

図 6 100 エピソード毎の獲得効用の推移
Fig. 6 Transition of utilities in every 100 episodes.

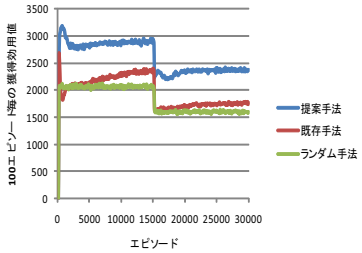

図 7 100 エピソード毎の獲得効用の推移
Fig. 7 Transition of utilities in every 100 episodes.

4.3 環境の変化を想定した実験

次に要求サービスの変化といった環境の変化を想定し，タスクが変化した際の効用の推移を調査する．

タスクの種類の変化を想定し，15000 エピソードまでタスク T_1 を，30000 エピソードまでタスク T_2 を与える実験を行った．このとき，平均 $\lambda = 1.5$ のポアソン分布に従いタスクを与えた．提案手法，既存手法，ランダム手法の獲得効用値の推移の結果を図に示す．

図 7 において，全エピソードを通して提案手法が一番獲得効用が高くなることが分かる．提案手法はタスクの種類が変化しても，環境に追従し，チーム編成を効率化できたといえる．

5. 結　　論

本研究では，1) の手法で用いられたモデルを，報酬の伝搬とその遅延の影響を複数計算機にまたがる強化学習として定式化した．さらに，組織構造を変化させることで初期状態に依存せずに効率化が行われることを示した．また，タスクの種類の変化といった通信遅延の以外の環境変化に対しても，3) などの従来手法より効率的なチーム編成が可能なことを示した．

今後の課題として，計算機ネットワークの過学習を防ぐアルゴリズムの導入がある．長い時間が経過し学習が進んだ場合，またリンクの数が増えた場合，過去の学習結果により外部の環境変化に追従しにくくなることが予想される．第 3.3 で述べたとおり，リンクの上限数や切断条件は切り替え戦略の問題であり議論の余地がある．さらに，7) にあるように，階層構造以外の構造への適用がある．スモールワールドネットワークやスケールフリーネットワークといった大規模な構造においても，リンクの動的生成がアルゴリズムの効率が下がらないような工夫が必要であり，今後の検討課題としたい．

参 考 文 献

1) Ryota Katayanagi and Toshiharu Sugawara, “Efficient Team Formation based on Learning and Reorganization and Influence of Communication Delay”, Computer and Information Technology, IEEE International Conference, 2011
2) 片柳 亮太, 菅原 俊治, “リンクの動的生成を用いたチーム編成の効率化の提案と評価”, 人工知能学会論文誌, Vol. 26, No. 1, pp.76-85, 2011
3) Sherief Abdallah and Victor Lesser, “Organization-based cooperative coalition formation”, Intelligent Agent Technology, 2004. (IAT 2004). Proceedings. IEEE/WIC/ACM International Conference on, 2004
4) Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning”, A Bradford Book, 1998
5) A.Barto and S.Mahadevan, “Recent Advances in Hierarchical Reinforcement Learning”, Discrete Event Systems journal, vol.13, pp.41-77, 2003
6) O.Sheholy and S.Kraus, “Methods for Task Allocation via Agent Coalition Formation”, Journal of Artificial Intelligence, vol.101, pp.165-200, 1998
7) Matthew E. Gaston and Marie desJardins. Agent-organized networks for dynamic team formation. In Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems (AAMAS ’05). ACM, New York, NY, USA, 230-237, 2005