Vol.2018-ICS-190 No.9
2018/3/3

通信遅延がある環境における効率的なチーム編成手法の提案

舟戸 崚也[a)] 早野 真史[b)] 飯島 直輝[c)] 菅原 俊治[d)]

概要：本研究では，マルチエージェントシステムにおけるタスク割り当て問題をチーム編成問題と捉え，そのモデルに通信遅延を導入し，チーム編成時の自らの役割学習と他のエージェントの協調可能性の学習を組み合わせた効率的な割り当て手法を提案する．一般に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求（サブタスク）で構成され，それらすべてのサブタスクを実現することでそのサービスが提供される．そのため，迅速なサービス提供のためには各タスクごとにそれを構成する全サブタスクに適切な能力を持つエージェントを割り当て，一つのチームとして処理する必要がある．これまでのチーム編成問題において，チーム編成の成功率向上を目指し，各エージェントは役割と他のエージェントに対する協調の期待度を自律的に学習する手法がある．しかしその手法では他のエージェントとのメッセージのやり取りにかかる通信遅延を考慮していなかった．現実のネットワークではタスクの処理そのものに加えて通信にも無視できない遅延が発生することがある．そこで本研究ではエージェント間の通信に遅延を加え，モデルを拡張する．このモデルに合致するように既存の学習手法を拡張し，通信の実績からより効率の良いチーム編成を学習する手法を提案する．評価実験から通信遅延を考慮しない既存手法と比較し，効率が改善されたことを示す．

キーワード：マルチエージェントシステム，分散制御，タスクアロケーション

Ryoya Funato[a)] Masashi Hayano[b)] Naoki Iijima[c)] Toshiharu Sugawara[d)]

1. はじめに

近年，グリッドコンピューティング [1] や IoT（Internet of Things）[2] など，多数の異なるエージェント（計算機や機器とそれらの制御ソフトウェア），また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．こうしたサービス（タスク）はそれを構成する多数の部分的な要求（サブタスク）に分割され，それぞれが適切な機器に割り振られて処理されることで提供される．これらのサブタスクが一つでも処理が遅れたり処理に失敗すると，サービスの遅延や提供不可につながる．したがって，迅速かつ確実にサービスを提供するために，あるサービスに対して連携して処理をする機器の集合をチームと捉え，効率的で適切にチームを編成する手法が必要となる．

a) r.funato@isl.cs.waseda.ac.jp
b) m.hayano@isl.cs.waseda.ac.jp
c) n.iijima@isl.cs.waseda.ac.jp
d) sugawara@waseda.jp

このようなチーム編成の問題では，環境が小規模であれば集中制御でも現実的な時間で最適な割り当てができる．しかし将来のネットワークで想定されるような大規模環境では，集中制御ではその負荷が大きくなり現実的でない．そこで分散制御により個々の機器が自律的にチーム編成を行いタスクを処理するモデルがマルチエージェントシステムの枠組みで提案されている [3], [4]．しかし，その多くは通信にかかる時間を考慮していないか一定に設定している．上記の応用例を考えると，通信時間を考慮せずに最適な割り当てを実現するのは非現実的である．例えば，スマートフォンを利用したグリッドコンピューティングや IoT のサービスが提供され始めている [5] が，この場合回線の状態や物理的な距離，ルータのホップ数などに依存した通信時間がかかり，通信によるオーバーヘッドを減らすことが迅速なサービス提供に必須である．

そこで本研究では，[3] のモデルを拡張し，エージェントがメッセージを送受信する際に距離に依存した通信時間を導入し，特に通信遅延が処理時間と同等以上で無視でき

ない状況を想定してチーム編成に与える影響を調べる．具体的には，エージェントに自身の役割 · 行動戦略と自分がチームを組むべきエージェントを学習させ，全体としてのタスク処理数を向上させる．

本論文の構成は以下の通りである．次節で関連研究について述べ，第 3 節ではエージェント · タスク · チームなどのモデルについて述べる．第 4 節でそのモデルにおける提案手法を提案し，第 5 節で実験により提案手法を評価 · 考察する．最後に第 6 節で全体をまとめる．

2. 関連研究

マルチエージェントシステムによる資源割り当てを対象とした研究として，チーム編成問題，提携形成問題，市場モデルに基づく割り当て問題などがあり，これらに関する研究は多く存在する．例えば契約ネット（contract-net protocol）[6] では，エージェントがマネージャと契約者の役割を持ち，マネージャはエージェントにタスクを広報し，契約者は広報されたタスクに入札する．その後マネージャは入札エージェントから最適なエージェントを決め，そのエージェントにタスクを割り当てる．しかし，この方式だとマネージャは全契約者に広報メッセージを送るため，大規模な環境ではメッセージによる負荷が大きいだけでなく，メッセージの待ち時間の増加，競合するエージェントの増加などの課題がある．[3] では複数のサブタスクで構成されるタスクがキューから順番に与えられるモデルで，タスクを処理するチーム編成の効率化を行っている．ここでは，エージェントがリーダーかメンバの役割を持つ．各エージェントは，自らのチーム編成の履歴から自分が適している役割と自分がチームを組むべき相手，また自分が取るべき戦略を学習し，チーム編成を自律的に最適化する．[4] はタスクにデッドラインがあることを想定した場合のチーム編成の効率化，及びタスク処理量の向上を目指したものである．この研究ではチーム編成の効率化だけでなく，エージェントがチームに必要以上に拘束される時間を減らし，リソースの無駄を減じることにも焦点を置いている．しかしこの 2 編では，チーム編成の過程においてエージェント間のメッセージのやり取りが即座に行われることを仮定しており，通信にかかる時間を無視できない環境を考えると適さない．

ビークルフォーメーションや人工衛星などの協調姿勢制御といった移動体群のフォーメーション制御の領域では，マルチエージェントシステムの枠組みで通信遅延を考慮した研究が盛んに行われている [7],[8], [9]．こうした研究は我々がタスクと呼んでいるものを位置 · 姿勢制御と具体化したものと捉えられる．しかし通常こうしたフォーメーション制御の分野では通信の相手が限られている場合を想定しているため，不特定多数が通信可能であり協調相手が適宜入れ替わるインターネットにおいて提供されるサービスに適応できない．

マルチプロセッサシステムやグリッドでの環境におけるタスクスケジューリングの問題でも通信遅延を考慮した研究 [10], [11], [12] があるが，これらの研究はタスクが発生するたびにそれを処理するチームを適宜編成するというものではなく，またタスクとして大規模計算，リソースとしてプロセッサの一種類を想定しているため，様々な種類を持つリソースを必要とするタスクが順次発生しその度に異なるエージェントと連携して処理しなければならないセンサネットワークや群ロボット制御，IoT などに適さない．

本研究では以上を踏まえ，[3] のモデルと手法を拡張し，通信遅延を考慮したチーム編成の一手法を提案する．

3. モデル

3.1 エージェントとタスク

エージェントの集合を $A = \{1, ..., n\}$ と置く．エージェント $i \in A$ は自身の処理能力に相当する p 種類のリソース $H_i = \langle h_i^1, ..., h_i^p \rangle$ を持つ．ただし，$\forall i \in A$ に対して $0 \leq \forall h_i^k,\ \sum_{k=1}^{p} h_i^k > 0$ とする．すなわちエージェントが持つリソースのうち最低でも一種類は $h_i^k > 0$ を満たす．$\sum_{k=1}^{p} h_i^k$ の値が大きいほど，エージェントは高いタスク処理能力を持つ．また，i は自身の役割 $r_i \in \{$ リーダー, メンバ $\}$ を自律的に決定する．ここで，リーダーはチーム編成を，メンバはチームに参加しタスクを処理する役割とする．さらにエージェント i は $X \times Y$ の大きさのグリッドに配置され，その場所に対応する二次元座標 (x_i, y_i) を持つ．エージェント i と j の距離を m_j^i とし，m_j^i はマンハッタン距離 $m_j^i = |x_i - x_j| + |y_i - y_j|$ とする．

システムに一定時間ごとに追加するタスクを T と表す．T はサブタスクの集合 $S_T = \{s_1, ..., s_m\}$ で構成される．ここで，m はタスク T に含まれるサブタスクの数を表す．各サブタスク s_i は処理するためにリソースを要求し，その要求リソースを $R_{s_i} = \langle r_{s_i}^1, ..., r_{s_i}^p \rangle$ と書く．なお，$0 \leq \forall r_{s_i}^j$ とする．本研究では通信遅延の影響に焦点を当てるため，s_i はただ一種類のリソースを要求すると仮定しタスクの処理を単純化した．すなわち s_i が要求するリソース番号を j とすれば，$0 < r_{s_i}^j,\ r_{s_i}^k = 0\ (k \neq j)$ である．エージェント i は，サブタスクが要求するリソースを持つときそれを処理できる．サブタスクが要求するリソース番号を k とすると，その処理時間は $\lceil r_s^k / h_i^k \rceil$ $[ticks]$ である．タスク T はすべてのサブタスク $s \in S_T$ を処理したとき完了となる．そのためタスク T の処理には，各サブタスクを担当する複数のエージェントからなるチームを組む必要がある．

タスク T は環境で生成され，システムに用意されたキュー Q に追加される．キューのサイズ（最大長）を q_{max} と表し，その値を超えたときタスクは破棄される．本モデルに離散時間を導入し，その単位を $tick$ と呼ぶ．

3.2 チーム

エージェントはチームを編成しタスク T を処理する．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 S_T から G への関数であり，$\sigma(s) = i$ は $s \in S_T$ を $i \in G$ に割り当てることを表す．すべてのサブタスク s についてエージェントの割り当てが完了したとき，チーム編成に成功したとしてタスクを処理できる．

チーム編成は各エージェントが役割を選択することから始まる．役割としてリーダーを選択したエージェントはキュー Q の先頭からタスクを取り出し，その中から自分のリソースで可能なサブタスクがあればそのうち一つを選ぶ．残ったサブタスクひとつにつき，ある戦略に基づき R 体のエージェントを選び，チーム参加要請を送信しその返信を待つ．その後参加要請の返信を確認し，受理を返したメンバエージェントと自身の集合を仮チーム G^P とする．そして G^P の中で再びある戦略に基づき割り当て σ を決定し，割り当てるエージェントを G に入れる．すべてのサブタスクについて割り当てが決定したらチームの成立とみなし，$i \in G$ にチームの成立通知とともにサブタスクを割り当てその処理を依頼し，自分もサブタスクを処理する．一つでも割り当て不可能なサブタスクがあった場合はチーム不成立とし，$i \in G^P$ に不成立通知を送信する．なお，リーダーが自分が処理するサブタスクを選択する際自分のリソースで可能なサブタスクが存在しなければ，そのすべてのサブタスクをメンバに割り当てる必要がある．

役割としてメンバを選んだエージェントは，受信したチーム参加要請を参照し，提案手法で述べる戦略により参加するチームを決め，その参加するチームのリーダーに参加表明を行い，それ以外のリーダーには拒否を伝える． なお，各エージェントが同時に所属可能なチーム数は一つと制限する．このためタスク処理中のエージェントは，参加要請メッセージに対し拒否のメッセージを出す．

3.3 通信遅延

チーム編成の一連の流れにおいて, $\forall i \in A$ と他のエージェントとのメッセージのやり取りに通信遅延 $D_i = \{d_1^i, ..., d_n^i\}$ を設ける．ただし d_j^i は 1 以上の整数で，L を最大の通信遅延として $d_i^i = 0,\ 1 \leq d_j^i \leq L\ \ (i \neq j)$ を満たすとする．エージェント間のメッセージのやり取りには以下の項目が該当する．

・ リーダーから他エージェントへの参加要請
・ リーダーからの参加要請に対する受理 · 拒否返信
・ リーダーから $i \in G$ へのチーム成立通知
・ リーダーから $j \in G^P$ へのチーム不成立通知

d_j^i の値は i と j の距離に依存して定義する．エージェント i と j のマンハッタン距離 m_j^i により，i と j 間の通信遅延 d_j^i を $\lceil m_j^i/(X+Y) \times L \rceil$ とする．$(X+Y)$ はグリッド上で最も遠い 2 点間のマンハッタン距離なので，$m_j^i/(X+Y)$ はそのグリッドにおいて標準化した距離となる．

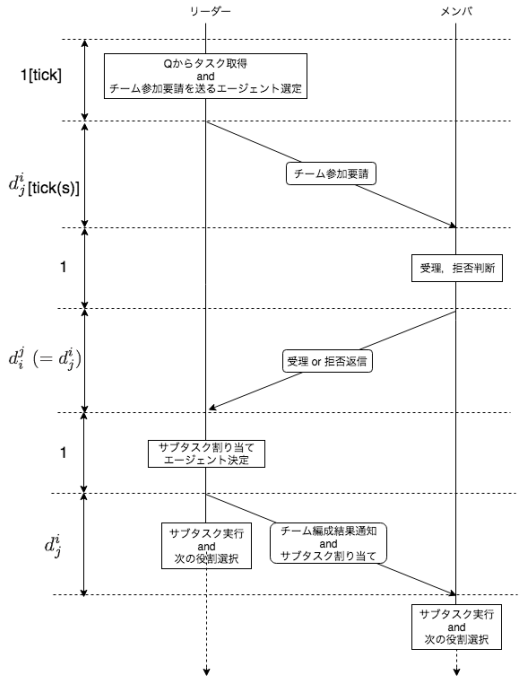


図 1: チーム編成の流れ（サブタスクを割り当てる場合）

3.4 処理の流れと時間経過

本モデルでは，$1[tick]$ ごとに $\lambda > 0$ のタスクが発生して Q に蓄えられ，リーダーの役割を選択したものが順に処理を進める．ここで λ はシステムへのタスクの追加数とし，値が大きいほど負荷が高いことを表す．以降の処理は第 3.2 章で述べたとおりである．

各処理にかかる時間を定義する．リーダーがタスクを取り出してからチーム参加要請を送信するまでと，メンバが要請を受けてからそれに対する返信を送るまでと，リーダーがチーム参加要請に対する返信を受けてからチームを編成しメンバにサブタスクを割り当てるまでを一律で $1[tick]$ とする．また本研究では通信遅延が処理時間と同等かそれ以上であることを想定し，サブタスクの処理とその後の役割選択にかかる時間を合わせて $1[tick]$ とする．図 1 にサブタスクの割り当てを行う場合の処理の流れを示す．

4. 提案手法

[3] では，役割適正値 · 信頼度 · 行動戦略の 3 つの項目をエージェントが学習し，その結果チーム編成を効率化できることを示した． 役割適正値は自分がリーダーとメンバのどちらに向いているかを示し，信頼度は自分がどのエージェントを優先してチームを組むべきかを表し，行動戦略はメンバがあるリーダーからのチーム参加要請を受理するかそれとも拒否するかを決定する指針となるものである．詳細はのちに述べる．

本研究ではモデルに通信遅延を導入したことに伴い信頼度の更新式を変更することで，通信遅延が無視できない分散環境での効率的なチーム編成の手法を提案する．さらに通信遅延を導入したモデルにおいて，一度役割がメンバとなったエージェントは，長時間どのリーダーからの参加要請も受信せず役割を正しく更新しない可能性がある．そのため定期的な役割更新の機構を提案する．以下でエージェントが学習する各種パラメータ，行動戦略，役割更新方法について述べる．

4.1 最短通信所要時間

エージェント i から暫定で最も近いエージェントとの通信時間を最短通信所要時間 Mt_i とする．i が他のエージェント j に返信を期待するメッセージを送信してからその返信を受信するまでの観測時間を Ct_j^i とする．すなわち，Ct_j^i は i がリーダーなら j に参加要請を送ってから j の受理 · 拒否返信を受信するまでの時間であり，i がメンバなら j に受理返信を送ってから j のチーム成立 · 不成立通知を受信するまでの時間である．Mt_i を以下で更新する．

$$Mt_i = min(Mt_i,\ Ct_j^i) \tag{1}$$

Mt_i はより小さい値で更新されることを期待してその初期値を十分大きく設定する．また，Mt_i は後述の信頼度よりも先に更新する．

4.2 役割適正値

エージェント i はチーム編成の成功率の高い役割を選択するために役割適正値を学習する．役割適正値にはリーダー適正値 e_i^l とメンバ適正値 e_i^m の 2 種類があり，それぞれリーダーあるいはメンバとしてのチーム編成成功可能性を表す．すなわち，e_i^l が高いほど i はリーダーとしてチームを編成した時の成功率が高く，e_i^m が高いほど i はメンバとしてチーム参加要請を待ってチームに参加した時の成功率が高いことを表す．これらの値は以下の式で更新される．

$$e_i^l \ = \ \alpha_r \times \delta_r^l + (1 - \alpha_r) \times e_i^l \tag{2}$$

$$e_i^m \ = \ \alpha_r \times \delta_r^m + (1 - \alpha_r) \times e_i^m \tag{3}$$

ここで $0 < \alpha_r (<< 1)$ は役割適正値の学習率である．各エージェントが得る効用に当たる $\delta_r^l,\ \delta_r^m$ は以下で定義される．

$$\delta_r^l = \begin{cases} 1 \text{ (チーム編成成功時)} \\ 0 \text{ (チーム編成失敗時)} \end{cases} \tag{4}$$

$$\delta_r^m = \begin{cases} 1 \text{ (サブタスク処理時)} \\ 0 \text{ (サブタスク未割り当て時)} \end{cases} \tag{5}$$

更新式からわかる通り，自分の役割によって役割適正値の更新タイミングが異なる．エージェント i は役割を選択する際，e_i^l と e_i^m の値を比較し，前者が大きければリーダーを，後者が大きければメンバを，両者が等しければランダムで役割を選択する．ただし，$\varepsilon - greedy$ 戦略を用いて，ε の確率で役割をランダムに選択する．もしリーダーを選択したがタスクが Q にない場合，リーダーはすでに過剰に存在しており，自分はリーダーである必要がない可能性があると判断し，役割適正値をチーム編成失敗時と同じ式で更新し役割を選択し直す．

4.3 信頼度

エージェント i はエージェント $\forall j \in A \setminus \{i\}$ に対する信頼度 c_j^i を持つ．信頼度は自分の行動に対して j が協調してくれる可能性を表す．i がリーダーである場合，チーム参加要請を送るエージェントを決定する際に信頼度の高いエージェントを優先する．また，i がメンバである場合，信頼度の高いリーダーを優先して受理返信を返す．したがって信頼度 c_j^i は，i がリーダーである時は「i が j にチーム参加要請を送った時にそれを j が受諾してくれる期待値」であり，i がメンバである時は「i が j からのチーム参加要請を受信した場合に i が受理を返したとき，i に実際にサブタスクを割り当ててくれる期待値」を意味する．つまり信頼度とは，i と j の役割に関係なく個人として i が j を信頼している度合いである．ただし本モデルにおいては可能な限り通信による遅延を排除したチーム編成を目的とするため，i がリーダーであるかメンバであるかによって j に対する信頼度更新の基準を変える．すなわち，i がリーダーである場合は自分のチーム参加要請を拒否したエージェントの信頼度を下げ，受理したエージェントの中でも応答が早いエージェントを高く評価する．一方 i がメンバである場合は自分にサブタスクを割り当ててくれたリーダーを高く評価し，受理返信をしたのに自分にサブタスクを割り当ててくれなかったリーダー，すなわち自分を裏切ってきたリーダーに対しては自分を拘束した時間 Ct_j^i に比例して信頼度を下げる．以上を元に c_j^i を次のように更新する．

$$c_j^i = (1 - \alpha_c) \times c_j^i + \alpha_c \times \delta_c \tag{6}$$

ここで α_c は学習率であり，$0 < \alpha_c (<< 1)$ の定数である．なお，エージェントの役割によってその更新タイミングと δ_c が異なる．自身がリーダーであるとき，更新タイミングは j からのチーム参加要請受理 · 拒否返信受信時で，

$$\delta_c = \begin{cases} Mt_i / Ct_j^i & \text{(受理返信)} \\ 0 & \text{(拒否返信)} \end{cases} \tag{7}$$

で更新する．Mt_i が先に更新されるので常に $0 \leq \delta_c \leq 1$ である．また，自身がメンバであるとき，更新タイミングは j からのチーム成立 · 不成立通知受信時で，

$$\delta_c = \begin{cases} 1 & \text{(成立通知)} \\ c_j^i \times (1 - Ct_j^i/Mt_i) & \text{(不成立通知)} \end{cases} \quad (8)$$

で更新する．不成立時の c_j^i の更新式を展開して計算すると $c_j^i = (1 - \alpha_c \times Ct_j^i/Mt_i) \times c_j^i$ であり，失敗時には自身を拘束した時間が長いほど信頼度を下げていることがわかる．以上によりリーダーは応答の早いエージェントとチームを組み，メンバは通信時間の長いリーダーに不当に拘束されないようにする．なお，古いチーム編成の履歴を忘れるために信頼度の蒸発を設ける．以下の式で信頼度を蒸発させる．

$$c_j^i = max(c_j^i - \gamma,\ 0)$$

ここで，$0 < \gamma << 1$ は定数である．信頼度の減少は全エージェントが毎 $tick$ 行う．これは蟻コロニー最適化手法 [13] などでフェロモンが蒸発するのと同じ効果がある．

4.4 行動戦略

目先の利益を優先してリーダーからの要請を受理する合理主義と，後で見返りがあることを期待して信頼できるエージェント以外からの要請を拒否する互恵主義の二つの行動戦略を設ける．本節では，合理主義及び互恵主義の詳細，次に互恵主義的行動を行う協調相手（以下，信頼エージェント）の判定方法を述べ，最後に行動戦略の選択について説明する．

合理主義のメンバは，チーム参加要請が一通でもあれば，必ずその中から最も信頼度の高いリーダーのチームに参加する．対して互恵主義のメンバは，信頼エージェント以外のリーダーからの要請は全て拒否し，信頼エージェントからの要請を待ち続ける．これは以下の理由による．本モデルにおいて，メンバエージェントは同時に複数のリーダーからチーム参加要請を受け取ることがある．この場合，その中で最も信頼度の高いエージェントの要請を受理するのが合理的である．しかしこれが常に最善とは限らない．信頼度の学習が十分に進むとチームを組むエージェントはある程度固定化するが，しばしば信頼度がさほど高くないエージェントからのチーム参加要請を受け取ることがある．常に合理的に行動するとすれば，メンバエージェントはその時にどのチームにも参加していなければあまり信頼度が高くないリーダーからの要請でも受けることになる．しかしこうして別のチームに参加している間に自分に対する信頼度が高いリーダーからの要請があった場合，その要請は断らざるを得ず，その結果それまで築いてきた信頼関係が崩れ，長期的に見ればシステムの効率が落ちる恐れがある．よって自分と強固な信頼関係にあると判断できるエージェントがいる場合，そのエージェントからの要請があることを期待して他のエージェントからの要請を断る必要がある．以上の観点から，役割がメンバである場合に，状況に応じて合理主義と互恵主義を切り替えさせる．

信頼エージェントの判定方法を説明する．「エージェント i にとって j が信頼エージェントである」とは，i に対して j が協調的と判断でき，i が互恵行動を選択したときに i は j を優先して協調する相手とみなすということである．本研究では各エージェントは信頼エージェントを上限 X_F 体まで保持する．また，エージェント i の信頼エージェントの集合を F^i と表す．i は，信頼度の上位 X_F 体のエージェントの中から信頼度が T_r を超えるものを信頼エージェントとみなし，F_i に追加する．T_r は信頼エージェントの判定の基準となる閾値である．各エージェントは信頼度を更新するたびに信頼エージェントを見直す．

エージェント i は役割選択時に，合理主義と互恵主義のいずれかを自分の役割適正値と信頼エージェントの有無により決定する．具体的には，行動戦略選択基準の閾値 $0 < T_s < 1$ に対し，$e_i^l > T_s$ かつ $F^i \neq \phi$ または $e_i^m > T_s$ かつ $F^i \neq \phi$ のとき互恵主義を，この式を満たさなければ合理主義を選択する．なお，初期段階では $F^i = \phi$ であるため，全エージェントは合理主義をとる．

4.5 役割更新

適切な役割の学習のために定期的な役割更新の機構を設ける．本モデルにおいて，メンバになったエージェントは自発的には何もせず，リーダーエージェントのチーム参加要請を受けてチームに参加したらサブタスクを処理するという消極的な立場を取っている．そのため，本来であればリーダーになるべきエージェントが役割を適切に学習できずにメンバであり続けてしまう可能性があり，リーダーが不足する恐れがある．そこでメンバエージェント $i \in A$ が最後に役割を選択した時刻を t_v^i とし，その時刻からリーダーエージェントのチーム参加要請を受理するまでの時間が閾値 T_v を超えた場合に，自分はメンバに適さない可能性があると判断し，サブタスク未割り当て時の式によって役割適正値を下げ，改めて役割を選択させる．

5. 実験と考察

5.1 実験環境

本モデル上で提案手法の有効性を示すために評価実験を行う．実験における各種パラメータの設定を表 1,2,3 に示す．

提案手法の評価のために以下の 3 つの手法と比較する．なお，タスクを構成するサブタスク数を 3 ～ 6 のランダムとしており，その分布は正規分布に従うので，タスクは平均して 4.5 個のサブタスクから構成される．したがって平均してエージェント 5 体前後でチームを組むことが想定されるので，学習なしの手法では全エージェント 500 体のうちリーダー数がおよそ 100 体となるようランダムに配置した．学習ありの手法では初期状態でリーダー数とメンバ数

表 1: エージェント

パラメータ	値
エージェント数 $\|A\|$	500
エージェントを配置する空間 $X \times Y$	50×50
リソースの種類数 p	6
エージェントの各リソース量 h_i^k	0 or 1
通信遅延の最大値 L	10
チーム参加依頼提案数 R	2
信頼エージェント判定基準 T_r	0.5
役割更新の閾値 T_v	100
最大信頼エージェント数 X_F	5
行動戦略選択基準 T_p	0.5

表 2: タスク

パラメータ	値
タスクを構成するサブタスク数 $\|S_T\|$	$3 \sim 6$
リソースの種類数 p	6
タスクキューのサイズ q_{max}	500
サブタスクの各リソース量 r_i^k	0 or 1

表 3: 学習パラメータ

パラメータ	値
信頼度 c_j^i の初期値	0.1
役割適正値 e_i^l, e_i^m の初期値	0.5
Mt_j^i の初期値	2147483647 (int 型の最大値)
学習率 α_c, α_e	0.05
ϵ	0.01
信頼度の蒸発値 γ	0.0000001

が同じくらいになるようにランダムに配置し，適切な役割を学習させる．

最短応答優先手法 :

第 4 節で述べた学習を使わず，応答の早いエージェントを優先してチームを組む．役割を学習しないため，各エージェントは最初に定められた役割で最後まで行動する．リーダーエージェントは近くにいるエージェントを優先してチーム参加要請を送信し，メンバエージェントは最初に参照したチーム参加要請に受理を返す．

信頼度優先手法 1（既存手法） :

[3] の手法を本モデルに当てはめたものである．信頼度 · 役割適正値 · 行動戦略を学習する．ただし提案手法とは信頼度の更新式が異なり，式 (6) を

$$\text{リーダー} \quad \ldots \quad \delta_c = \begin{cases} 1 \text{ (受理返信)} \\ 0 \text{ (拒否返信)} \end{cases} \tag{9}$$

$$\text{メンバ} \quad \ldots \quad \delta_c = \begin{cases} 1 \text{ (成立通知)} \\ 0 \text{ (不成立通知)} \end{cases} \tag{10}$$

で更新する．したがって通信遅延は考慮されていない．なお，第 4.5 節で述べた役割の更新は入れない．従って適切に役割が学習できず，リーダー数が減少していくことが予想される．

信頼度優先手法 2 :

信頼度を第 4.3 節で述べた式によって更新し，役割 · 行動戦略も学習するが第 4.5 節で述べた役割の更新は入れない手法である．従来手法とは信頼度の更新式が異なり，提案手法とは役割の更新の有無が異なる．この手法と既存手法を比較することで提案する信頼度更新式の有効性を検証し，この手法と提案手法を比較することで役割更新の有効性を検証する．

5.2 チーム編成成功数の推移

提案手法と比較手法のチーム編成の効率を調査した．ここではタスクの追加数 $\lambda = 1, 5, 10, 15$ の 4 つの場合で，各手法のチーム編成成功数の推移を比較した．$\lambda = 1$ はタスクが少なく競合が発生しにくい環境であり，$\lambda = 15$ は全手法でタスクがキューから溢れ，システムの限界を迎えた（ドロップする）非常に高負荷な環境である．本実験では $7,000[ticks]$ 毎にチーム編成成功数を測定し，10 回の施行の平均を取った．この結果を図 2 に示す．

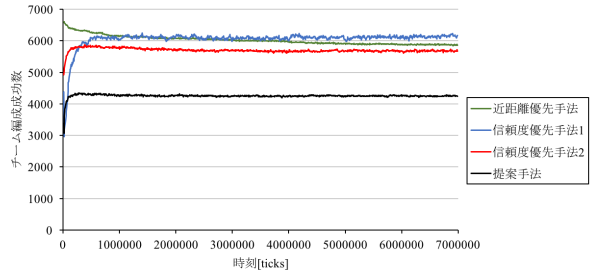
図 2 から，$\lambda = 1$ においては提案手法の結果がもっとも効率が低い．さらに信頼度の更新式に距離を考慮した指標を入れた信頼度優先手法 2 も他の二手法より効率は低い．これは負荷が非常に低いときはタスクの処理に余裕があり，必ずしも近くでチーム編成を組む必要がなかったためである．また負荷が小さいために，学習しない近距離優先手法においてもエージェントの競合が起きず十分な性能を発揮できた．一方で，提案手法と信頼度優先手法 2 の二手法では，追加されるタスク数が少なく学習の機会があまりなく，近くにいるエージェントを優先しようとする制約のため適切な協調関係が築けなかったと考えられる．

しかしその他の条件では距離を考慮した信頼度の更新が有効に働き，提案手法と信頼度優先手法 2 で効率の改善が見られた．また，信頼度優先手法 1, 2 ではチーム編成の成功数が時間とともに減少した．これは第 4.5 節で述べた役割の更新をしないために役割が学習できず，リーダーエージェントが徐々に減少したためである．一方で提案手法では役割更新が有効に働き，チーム編成成功数が高い水準で収束した．

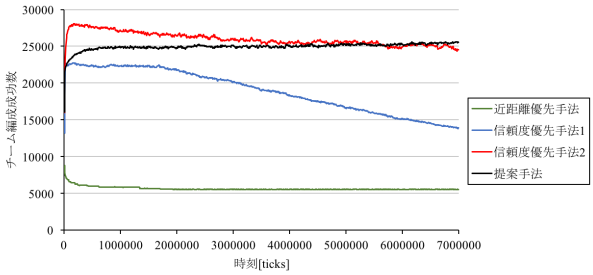
以上から，今回提案した距離を考慮した信頼度の更新式と役割更新の 2 つは概ねチーム編成の効率化に寄与するものの，環境の負荷が非常に低い場合においてはむしろ効率を低下させてしまうことがわかった．よって，より適切な学習機構を考案するか，環境の負荷に応じて手法を切り替える機構を導入する必要がある．

5.3 エージェント構造

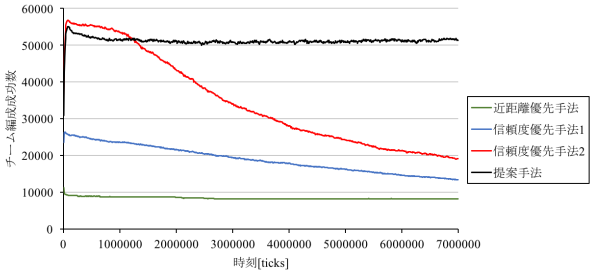
信頼度優先手法 1 と提案手法において得られた，最



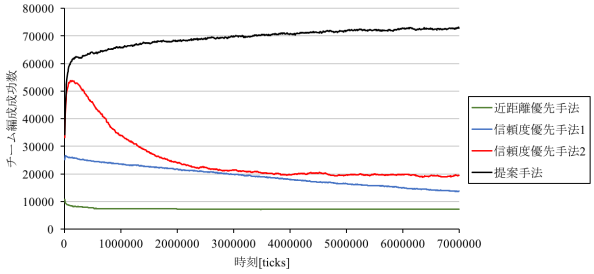
(a) $\lambda = 1$ におけるチーム編成成功数



(b) $\lambda = 5$ におけるチーム編成成功数



(c) $\lambda = 10$ におけるチーム編成成功数

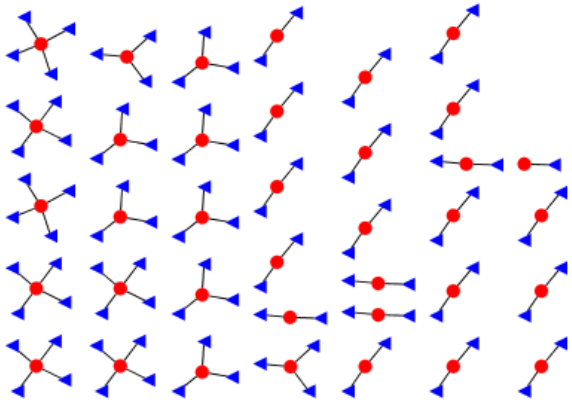


(d) $\lambda = 15$ におけるチーム編成成功数

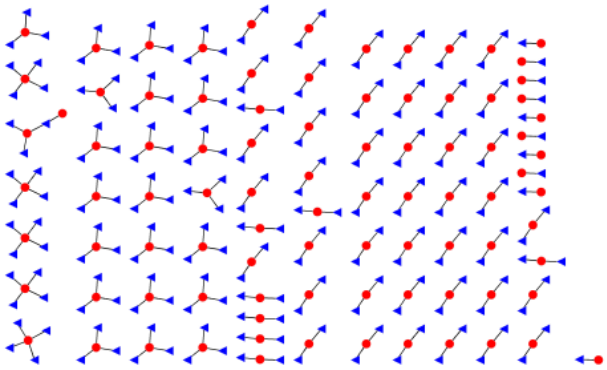
図 2: 様々な負荷 λ におけるチーム編成数

終的なエージェント間のメンバの依頼・受託関係の構造を Cytoscape を用いて可視化した．ここで生成した構造は，最後の $5,000[ticks]$ である $6,995,000[ticks]$ から $7,000,000[ticks]$ における協調関係である．リーダーがチーム参加要請を送り，受信したメンバがそれを受託してチーム編成に成功した回数が $5,000[ticks]$ 中に 100 回以上あった場合に実質的な協調関係があると考え，リンクを生成した．信頼度優先手法 1 と提案手法におけるリーダーと互恵主義のメンバの協調関係のグラフを図 3 に，提案手法におけるリーダーと全メンバの協調関係のグラフを図 4 に示す．図 3 では合理主義のエージェントを除いたことで，信頼度が高く恒常的な協調関係にあるエージェントのグループのみを抽出した．なお，各ノードはエージェントを表し，赤い丸のノードはリーダーエージェントを，緑の四角形のノードは合理主義のメンバエージェントを，青い三角形のノードは互恵主義のメンバエージェントを表す．

図 3a, 3b を比較すると，提案手法の方がチーム数が多く，より効率的に協調関係を結べている．これは信頼度の更新式の変更により応答時間の早いエージェントとチームを組むようになったことで相対的に学習の頻度が増えて効率よく協調関係が結べ，また適切に役割を学習しリーダー数が適切な値に収束したためである．また，図 3 を見ると，リーダーを含めて 3 ～ 5 体のエージェントによるチームが構成されている．これは一つのタスクを構成するサブタスク数の平均が 4.5 個であるためである．さらに図 4 によると，互恵主義のメンバと合理主義のメンバが共存していることがわかる．この合理主義のメンバの存在により，タス



(a) 信頼度優先手法 1 の構造



(b) 提案手法の構造

図 3: リーダーと互恵主義のメンバのみの構造

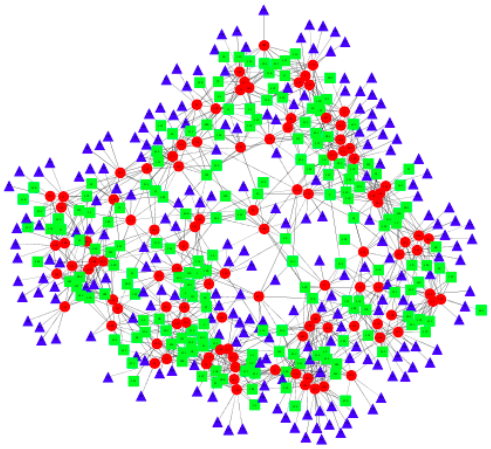


図 4: 提案手法のエージェント構造

クを構成するサブタスク数が多い場合にも対応している．すなわち，恒常的な協調関係にあるチームの大きさ以上の数のサブタスクから成るタスクに対しては，合理主義のメンバが一時的にチームに参加することで不足を補っている．

6. まとめと今後の課題

本研究では，他のエージェントとの通信所要時間を未知とした上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 [3] で提案したエージェントの自律的な学習を，通信遅延が無視できない環境に適用できるように拡張したものである．実験によって，環境の負荷が一定以上の場合では既存手法と比べ提案手法の方がチーム編成を効率化できることを示した．しかし一方で，環境の負荷が小さすぎるような場合には近くを優先する必要がなく，提案手法が他の手法に劣ってしまうという課題も残った．今後は今回明らかになった課題を解決することを目指すと共に，タスクの処理時間が変動する環境を想定してチーム編成の効率化を図り，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] Fran Berman, Geoffrey Fox, and Anthony JG Hey. *Grid computing: making the global infrastructure a reality*. John Wiley and sons, 2003.
[2] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer Networks*, Vol. 54, No. 15, pp. 2787 – 2805, 2010.
[3] Masashi Hayano, Yuki Miyashita, and Toshiharu Sugawara. *Adaptive Switching Behavioral Strategies for Effective Team Formation in Changing Environments*, pp. 37–55. Springer International Publishing, Cham, 2017.
[4] R. Kawaguchi, M. Hayano, and T. Sugawara. Balanced team formation for tasks with deadlines. In *2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT)*, pp. 234–241, Dec 2015.
[5] 総務省. 平成 29 年版情報通信白書.
[6] R. G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Trans. Comput.*, Vol. 29, No. 12, pp. 1104–1113, dec 1980.
[7] 早川朋久, 藤田政之. マルチエージェントシステムとビークルフォーメーション. 計測と制御, Vol. 46, No. 11, pp. 823–828, 2007.
[8] 橘義博, 滑川徹. 通信遅延を考慮した複数剛体の分散協調姿勢制御. *IIC*, Vol. 12, p. 136, 2012.
[9] J. A. Fax and R. M. Murray. Information flow and cooperative control of vehicle formations. *IEEE Transactions on Automatic Control*, Vol. 49, No. 9, pp. 1465–1476, Sept 2004.
[10] 野口智史, 大下福仁, 増澤利光. 通信遅延を考慮したタスクスケジューリングアルゴリズム. Technical Report 122(2003-MPS-047), 大阪大学大学院情報科学研究科コンピュータサイエンス専攻, dec 2003.
[11] 滝沢泰久, 芝公仁, 大久保英嗣. 連続メディア処理における時間制約と通信遅延に適応するタスクスケジューリング. 情報処理学会論文誌数理モデル化と応用（TOM）, Vol. 42, No. SIG05(TOM4), pp. 29–41, may 2001.
[12] 尾高輝, 甲斐宗徳. 通信遅延を考慮したタスクスケジューリングのためのタスク粒度解析. 成蹊大学理工学研究報告, Vol. 44, No. 1, pp. 17–23, 2007.
[13] David Corne, Marco Dorigo, Fred Glover, Dipankar Dasgupta, Pablo Moscato, Riccardo Poli, and Kenneth V. Price, editors. *New Ideas in Optimization*. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.