共同関係に基づく自律的組織化を利用した効率的な分散タスク割当手法の提案

Efficient Distributed Task Allocation based on Autonomous Organizational Formation using Cooperative and Reciprocal Relationships

2018年2月

早野 真史
Masashi HAYANO

共同関係に基づく自律的組織化を利用した効率的な分散タスク割当手法の提案

Efficient Distributed Task Allocation based on Autonomous Organizational Formation using Cooperative and Reciprocal Relationships

早野 真史
Masashi HAYANO

早稲田大学大学院 基幹理工学研究科
情報理工・情報通信専攻
知識ソフトウェア研究

2018年2月

目 次

第1章　序論

本章では，実際に実用化されているサービスをあげ，今後のサービスシステムの展望をふまえた課題を示す．次に，その課題解決に向けたエージェントを利用した代表的なモデルについてまとめる．その後，本研究モデルと提案手法の概要を述べ，最後に本論文の構成を示す．

1.1　背景と本研究の目的

近年，ありとあらゆる「モノ」がネットワークにつながり始めている．PC やスマートフォンに限らず，例えば家電や車にもセンサや通信機能が搭載され，モノ同士の通信や，ユーザ毎の様々な情報収集が可能となってきた．また，これらを利用した新たなサービスが実用化されており，Internet of things (IoT) [1, 2]，サービス (クラウド) コンピューティング [3, 4]，サイバーフィジカルシステム (CPS) [5, 6] と呼ばれている．具体的なサービスとして，複数台のロボットによる自動的清掃やセキュリティ巡回，スマートグリッド [7, 8] などがある．これらは，様々な要素で構成されたサービス要求に対して，モノが持つ情報や処理能力を，適切かつ迅速に組み合わせて割当てて処理し，サービスを提供する [9, 10].

一方で，今後，ネットワークにつながるモノやサービス利用者が更に増加し，システム自体の大規模化・複雑化が予想される．これに伴い，個々のサービス要求に対して，それを実現するために膨大なモノのリソース (例えばモノが持つ情報や処理能力) からより最適な組み合わせを見つけて割当てることは困難になる．このため，トップダウンにシステムを管理する集中的な制御ではなく，計算処理を分散させるボトムアップなサービス提供システムが提案されている．ここでは，要求を受けた (あるいは発見した) デバイス (例えばスマートフォンや，ロボットなど) が，サーバに全ての情報をアップロードして応答を待機するのではなく，デバイスそのもの自体が自律的に連携し，サービス要求を効果的に実現するシステムである．しかし，このようなシステム実現に向けた開発は容易ではない．具体的には，局所的に情報収集と計算処理が行われるため，必ずしも最適な組み合わ

せや割当ができるとは限らず，サービス提供の保障性や質は，集中制御に比べて低くなる可能性がある．特に，大規模な環境になるにつれて課題は複雑になり，適切な割当が難しくなる．

このような分散システムの実現にむけて，マルチエージェントシステムのフレームワークを利用したタスク・資源割当の研究がある．ここでは，ネットワークにつながれたもの (システムを構成するデバイスやアプリケーション) をエージェントとよばれる主体として抽象化し，エージェントをベースにした抽象モデルが提案されている．モデル例として提携形成問題 (coalition structure formation) [11, 12] や，チーム編成問題 (team formation or task-oriented coalition formation) [13, 14, 15] がある．提携形成問題は，与えられたタスク群に対してエージェントが最大の効用となる共同関係の組み合わせの構造を求めるモデルである．チーム編成問題は，マネージャやイニシエータがエージェントを取りまとめる役割を担い，それらが要求されたタスクを処理するエージェントをメッセージ利用して通信し，対応するタスク資源の割当によって共同関係を構築するモデルである．これらのモデルにおいて効率的なエージェントの連携や割当法が提案されており，分散システムへの応用が期待されている．一方で，必ずしも分散環境を想定しておらず，集中的な制御や静的な環境も想定している．例えば，エージェントの提携の効用値を与える特性関数が事前に与えられていたり，エージェントの能力が既知である環境が想定されているものがある．また環境が変化しない静的かつ小規模な環境を想定しているものもある．今後は，システムが大規模化・複雑化することも見込まれ，さらにシステム環境は動的に変化することもありうる．

そこで，本研究はまず第一にボトムアップにサービスを実現するシステムをモデル化し，エージェントによるタスク割当チーム編成モデルを提案する．このモデルは，タスク要求が継続的に発生し，システムを集中管理するマネージャが存在しない分散システムを想定したものである．次に，このモデルにおいて，他のエージェントの能力が未知な状況と大規模環境を想定した手法を提案する．能力が未知な状況に対しては，エージェントが他のエージェントの能力を実績から推定し，さらにタスクを処理するチームメンバの候補を合わせて学習する手法を提案する．

また，大規模化に対しては，２つの観点から手法を提案した．第一に，エージェント数が増加した大規模環境においても効率的にタスク処理をするために，それぞれのエージェントが連携すべきエージェントを，実績に基づき全体から一部のみに制限する手法 (スコープと呼ぶ) を提案し，その有効性を評価した．大規模環境ではシステムすべてのエージェントの存在や状態を知ることができない．例えば，どこに，いつ，どのような

エージェントがネットワークにつながれるか，というような情報を大規模分散システムにおいて常に情報を収集し最新に保つことは不可能である．また，仮にすべてのエージェント情報を保持していたとしても，それら全員とやり取りする通信量は膨大になり，処理すべきデータ量も増える．その結果，連携すべきエージェントの同定に時間がかかり，効率が落ちる．そこでスコープにより協調の相手を制限しても大規模環境で効率的なタスク処理が可能なことを示した．

第二に，上記のスコープを導入したときの課題と得た知見を利用し，ボトムアップにエージェントの堅牢な共同関係を実現する戦略選択エージェントを提案する．スコープの課題は，スコープサイズを静的に事前に決定している点にあった．本来全体のエージェントの数は予測できず，さらにその数や状況も変化する．一方で，スコープから得た知見としては，制限を加えることで学習速度を向上させ大規模環境でも協調すべき相手を早く同定でき，結果としてタスク処理の効率も向上できたことである．この知見を活用し，エージェントの協調関係を相互の実績に基づきボトムアップに創発・組織化させることで，大規模環境やさらに様々な負荷およびその変化に対応できることを示した．具体的な手法としては，人間でも観測される互恵性を元にした互恵戦略を導入し，従来の合理戦略と状況に応じて切り替える「戦略選択エージェント」を提案した．

本研究はこれらの手法によって情報が収集できない未知な環境や，大規模化に対応可能な手法を提案し，さらに分散システムを実用化した際に起きる現象やその解決方法について議論した．

1.2 本論文の構成

本論文の構成は次の通りである．次章で，エージェントとマルチエージェントシステムの特徴および関連研究について議論する．ここでは関連研究の課題を述べ，本研究との違いを明確にする．第 3 章では，本研究のエージェントのタスク割当チーム編成モデルについて述べ，エージェントやタスクの設定とシステム全体の流れについて述べる．第 4 章から第 6 章までは，第 3 章のモデルをベースに導入した手法とその評価実験について議論する．第 7 章では総括として本論文のまとめと今後の課題について述べる．本論文の構成図を図 1.1 示す．

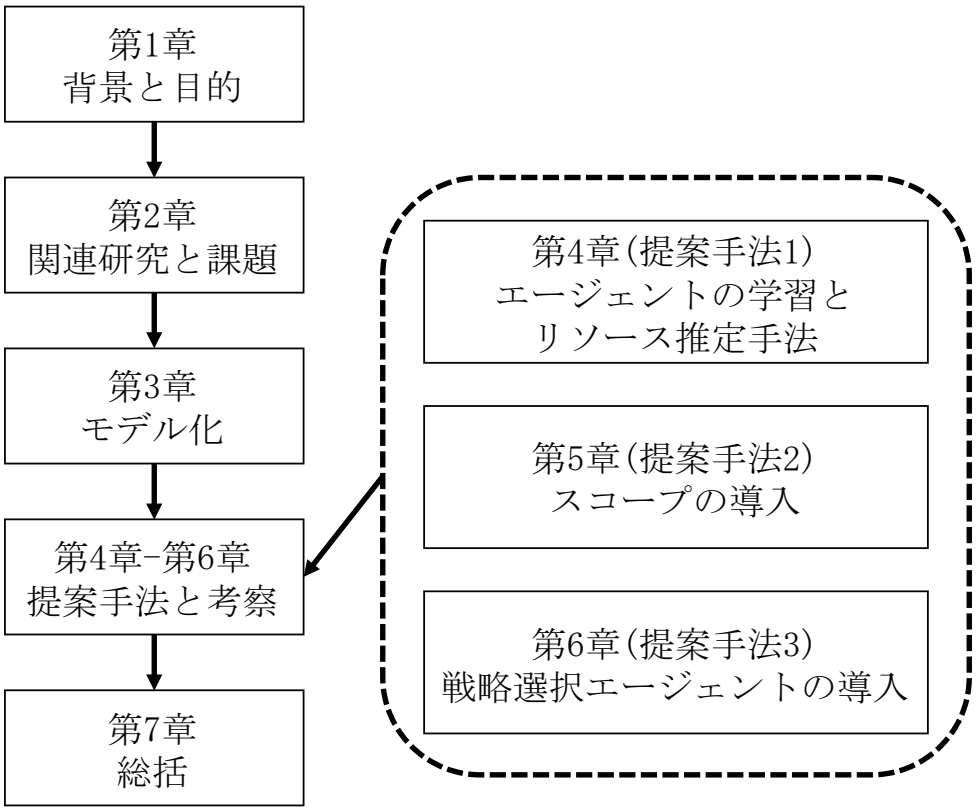
第1章
背景と目的
第2章
関連研究と課題
第3章
モデル化
第4章-第6章
提案手法と考察
第7章
総括
第4章(提案手法1)
エージェントの学習と
リソース推定手法
第5章(提案手法2)
スコープの導入
第6章(提案手法3)
戦略選択エージェントの導入

図 1.1: 論文構成

第2章　マルチエージェントシステムと関連研究

本章では，マルチエージェントシステムの特徴と関連研究を述べる．そのために，まず，分野によって定義が曖昧になるマルチエージェントシステムにおける (シングル) エージェントの概念をまとめる．次に上記で述べたエージェントの疎結合であるマルチエージェントシステムとそれを利用することで解決可能な問題と利点を述べる．その後関連した研究とその課題について述べ，本章のまとめとして本研究の位置づけを明確にする．

2.1　マルチエージェントシステムにおけるエージェント

「エージェント」と呼ばれる言葉は様々な場所で使用されており，その定義は多岐にわたる [16]．そのため，すべての概念を網羅的に包括したエージェントという言葉の定義は難しい [17]．そこで，本研究ではエージェントを以下のように定義する．エージェントとは，自分自身が存在している環境から，感覚器官 (センサ) を用いて情報を収集 (機械であればセンサやカメラ，生物であれば目や鼻が該当) し，その情報を元にアクチュエータを利用して何かしらの行動を起こすものである [18]．特に情報収集と各種判断を自律的に行い (自律性)，目的達成のために合理的に行動する (合理性) 主体をエージェントと考える．本研究のモデルにおける具体的なエージェントの設定については第 3 章で述べる．図 2.1 にエージェントの概要図を示す．またエージェント定義における，センサ，アクチュエータ，自律性，合理性についてそれぞれ以下に述べる．

センサ　センサは，例えば昆虫であれば触覚，人の場合では手足，口，耳といった様々な感覚器官がそれに当てはまる．また，機械などのデバイスであれば，室内温度を感知する温度計，部屋の状態を監視する監視カメラもそれに当てはまる．つまり，エージェントのセンサとは，実世界の環境や相手の状況など，情報が得られる部分に相当する．

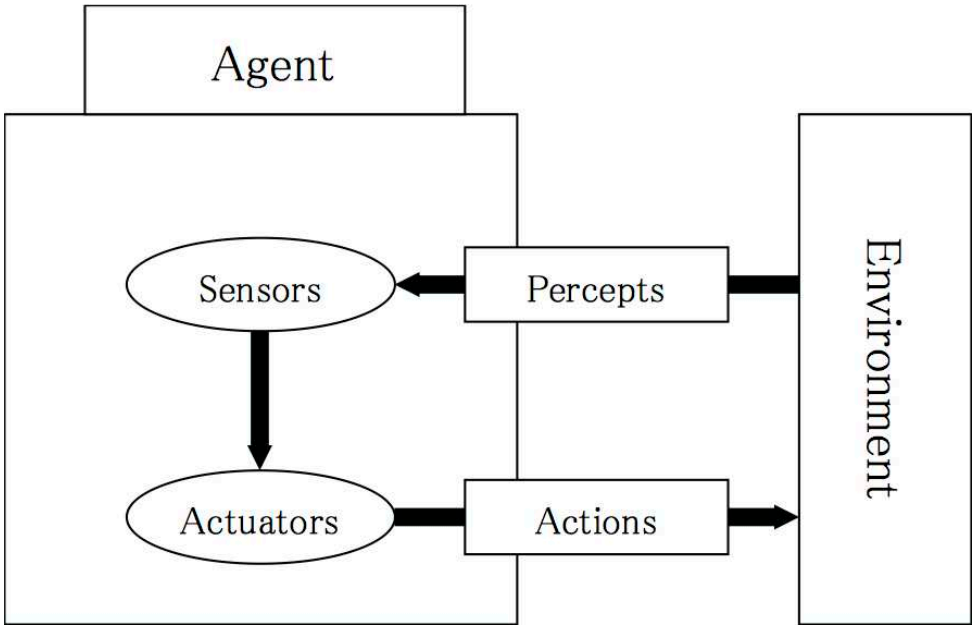

図 2.1: エージェントの概念図

アクチュエータ　アクチュエータは，環境に対して何かしらのアクションを起こせる機能や部分を指す．例えば，生物の場合，センサとして利用する手足や口は，環境においても何かしらアクションを起こすことができるため (例：手でモノを運ぶ，口でものを噛む)，アクチュエータにも該当する．掃除ロボットをエージェントとした場合，移動するためのタイヤやモータ，さらにごみ収集する装置部分がアクチュエータである．

合理性　エージェントの合理性は,「自己の信念を元にゴール (目標) を達成するために行動を起こす [18]」，と定義される．特にここでは，曖昧さを避けるために,「エージェントの設計時に与える評価指標や目的のために，自分の指標をより良くするように行動すること」を「合理性」または「合理的である」と定義する．例えば，掃除エージェントであれば，評価指標として「収集したゴミの量」が考えられ,「収集するゴミの量を多くする行動またはそれを発現するための推論」が合理的な行動と言える．

自律性　エージェントの行動は,「予めエージェントがもつ (あるいは組み込まれた) 知識と，環境から自分で得た知識によって決定される [18]」．このことから，本論文での自律性は,「予め行動指針のみがエージェントに与えられており，自分の判断で行動を選択すること」を自律性と考える．したがって，他のエージェントやシステムからタスクの依頼があっても，必ずしも直ちにそれに取り掛かるとは限らず，後回しにしたり無視すること

もあり得る．

2.1.1 学習エージェント

エージェントは自分自身の振る舞いやその行動決定に至るまでの過程から幾つかに分類される．ここでは特に，本研究分野での議論の中心である学習エージェントについて述べる．

学習エージェントとは，収集した情報や自分の行動結果 (から得た報酬) をエージェント自身に内蔵されている学習器によって学習し，それに基づき自分の行動を修正するエージェントである．人間や昆虫といった生物は，自分の行動指針を経験から学習し，修正するため，学習エージェントといえる．後述するマルチエージェントシステムの研究では，エージェントの学習方法や判断方法，あるいはその行動について議論している．以下にその学習方法とその行動決定方法の代表的な 2 つの手法を述べる．

帰納的な学習に基づく行動決定方法

帰納的な学習に基づく行動決定方法は，エージェントが過去に自分が経験した情報に基づいて行動を決定する方法である．例えばエージェントが状態 s において,「情報 A を収集・感知したため，B の行動をしたら C の結果 (成功や望ましい状況) を得た」，とする．このとき，エージェントは上記を記憶する．その後，C の (ような) 結果を得たいときは，s の状態において B の行動をとる，あるいは情報 A を収集する，と行動を決定するものである．つまり，過去の行動と結果を記憶し，そこから共通的であり最適と思われる行動規則を推論して決定する．帰納的な学習を利用した例としては決定木を利用したエージェントの行動規則の生成 [19] が挙げられる．この学習の欠点は，推論の精度を向上させるために必要な情報量の多さである．エージェントの数や種類が増えるほどその行動 (や状態) は多岐に渡る．そのため，それらを収集する時間も増加し，推論の精度を向上させるのに時間がかかる，あるいは向上できない可能性もある．

強化学習に基づく行動決定方法

強化学習とは，機械学習の一つで，エージェントが試行錯誤を通して環境に適応する学習制御の仕組みである [20, 21]．特徴として，教師付き学習とは異なり，行動結果の正解

不正解をフィードバックする教師が存在せず，代わりにエージェント自身が報酬というスカラ値の情報を元に環境に適応した行動を学習する．強化学習における環境とエージェントの関係性を図 2.2 に示す．

このようなエージェントと環境の関係性はマルコフ決定過程を用いてモデル化するのが有用である．通常のマルコフ決定過程では，環境の状態の集合，エージェントが取りうる行動の集合，環境の状態遷移確率，報酬の期待値，の 4 つが定義される．しかし実際のシステムでは，環境の状態遷移確率や報酬の期待値に関する知識は既知とできるとは限らない．一方で既知としない状態でも，マルコフ性を考慮した場合，ある状態における最適な行動の学習ができることが知られている．その代表的な学習アルゴリズムが Q-learning[21] である．Q-learning におけるエージェントは Q 値と呼ばれるスカラ値をもち，その値を学習することで最適な行動を学習できる．ここで Q 値はある状態 s においてエージェントの行動 a をした際に得られる利得の期待値であり，$Q(s, a)$ で表す．Q 値は以下の行動で学習する．

1. あるエージェントは時刻 t において観測した状態 s_t によって行動 a_t をきめ，その後報酬 u_t を受け取る．
2. 状態遷移後の状態 s_{t+1} を観測する．
3. 以下の式により Q 値を更新する．

$$Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha[u_t + \gamma \max_a Q(s_{(t+1)}, a)] \tag{2.1}$$

   ここで，α は学習率，γ は割引率を表す．
4. 時刻 $t + 1$ へ進め，1 に戻り，これを繰り返す．

本論文のエージェントもこの Q-learning の枠組みを利用して学習するエージェントである．

その他の行動決定方法

エージェントの行動を学習および決定する方法は，エージェントの種類，数，あるいはその目的によって異なり，すべての局面に適用できる最適な手法はない．ここでは割愛するが，他の手法としては遺伝的アルゴリズム [22, 23]，エージェントが得た多数の教師データを元に学習するニューラルネットワーク [24, 25] がある．

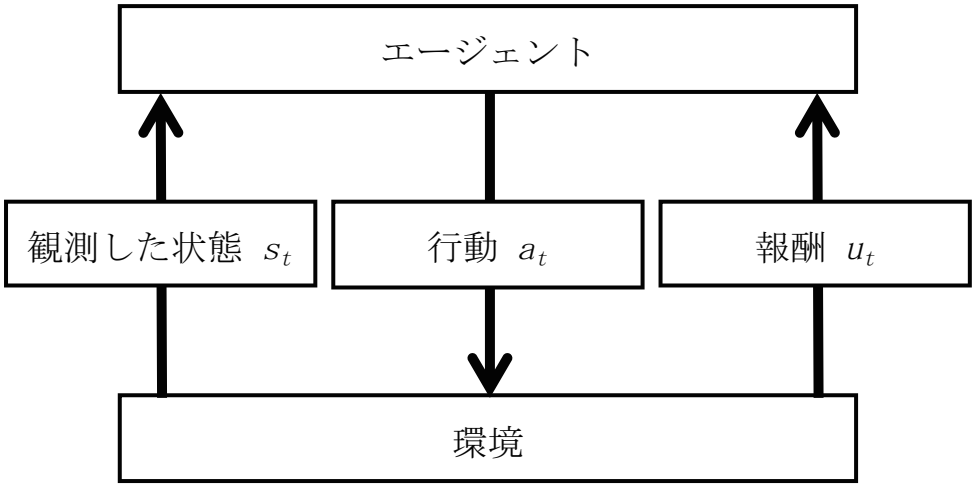

図 2.2: 強化学習の概念図

2.2 マルチエージェントシステム

マルチエージェントシステムとは与えられた課題を解決するために，複数のエージェントがそれぞれ持つ情報や能力を活用し，目的を達成するまでの行動計画を調整し，知的な集団行動を生み出す仕組みのことである [26, 27, 28, 29]．分散コンピューティングを元にした資源割当問題の研究のためのモデルの枠組みや，社会学や経済学における集団行動の心理の解析に用いられる．特にマルチエージェントシステムは広義の分散システムを考察するためのモデルをさす場合がある．マルチエージェントシステムの利点は以下の 5 点がある [30].

問題解決能力の向上　複数箇所に存在するエージェント (ノードと呼ぶ場合もある) や異なる能力をもつエージェントに処理を配分し，各エージェントの能力を組み合わせることでシングルエージェントでは実現できないような高度な仕事を効率的に達成できる．

適応能力　生活の多様化，大規模化により変化し続ける環境において，エージェントがその変化に対応する行動をとることで適切に対処できる．このために前述の学習エージェントが利用されることがある．

ロバスト性　エージェントの機能が故障した場合でも別のエージェントが代役をすることで，システム全体の処理停止を防ぐことができる．

並列性　複数のエージェントが並列して処理することで，システム全体の処理の高速化と負荷分散ができる．これにより，単一のエージェントに負荷が集中して高負荷になることを防ぎ，それによる性能の低下を防ぐことができる．

モジュール性　エージェント単位でモジュール化することで，はじめからシステムの詳細な設計を行わなくても設計済みのエージェントを再利用あるいは組み合わせることで設計コストや時間を低くできる．

これらの利点は単にエージェントを寄せ集めるだけでは得ることはできず，各エージェントがもつ機能を適切に組み合わせ，協調させる必要がある．特に，上記の利点をトップダウンに集中的に管理せず，ボトムアップに分散的に管理することで，効率性と柔軟性を兼ね備えたサービス・システムの実現が期待されている．次節で集中制御の課題と分散的な制御の利点についてまとめる．

2.2.1　集中制御の課題と分散的な制御の利点

シングルエージェントの実際の利用例は，例えばホームエージェントやモバイルエージェントがある．ここでのエージェントに求められる機能は，利用するユーザの満足度を高める情報を収集する，あるいは結果を実現することである．つまり，ユーザとエージェントの１対１の関係であり，その制御は集中制御でも問題ない．一方で，マルチエージェントシステムは，エージェントが複数存在しており，自分以外のエージェントとの関係に依存して自分の行動を変える必要がある．さらに自分の目的達成のために，自分以外のエージェントや環境の情報を収集する必要がある．そのための制御として，集中的な制御がある．集中的な制御は，サーバやクラウド，あるいは情報収集するためのエージェントが，環境の状態やエージェントに関する情報を収集・管理し，それによってシステム全体 (これにはエージェントの行動も含まれる) を制御する方法である．

しかし，実際のシステムでは，以下の課題により全体を集中制御できないケースが多い．

(a) 収集データの増加　ネットワークがオープン環境になり，ありとあらゆるモノが自由にネットワークにつながることが可能であり，その数も増え続けている．そのため，全てのモノの情報を一箇所にすべて集めることは難しくなる．仮に，容量の大きいハードディスクや処理が高機能な計算機を利用して収集や処理ができたとしても，あらかじめその数を予期したメモリの確保はできない．また，デバイスに搭載されるセンサやプロセッサ

の最新版への更新，あるいは物理的な故障による停止など，デバイスの処理能力は変化する．そのためリアルタイムな環境情報を一箇所に収集し続けることは困難である．

(b) 情報の機密性　個人のプライベートな情報 (例えばスケジュール，体重，趣味嗜好) は公にできない．また，企業として秘密にしておくべき機能や技術なども公になることはない．これらのことから，プライバシー，セキュリティの観点からすべての情報やリソースを一箇所に集め，処理することは必ずしも可能なわけではない．一箇所に集められたとしてもハッキングにより情報漏えいのリスクもある．

(c) トラフィックの増加による通信帯域の圧迫　モノの増加につれて，通信量も増加し，通信帯域が圧迫される．また，それによりその他のサービスに影響を与える可能性もある．

集中制御のこれらの課題から，サービスに必要な情報を集められず，遅延または失敗となることも考えられる．また，システムの状況を正確に把握できなくなり，ユーザが求めるサービスの質が低下する恐れもある．そのため集中的な制御ではなく，ボトムアップなアプローチ (分散的な制御) が求められている．

分散的な制御とは，サーバやクラウドで一極集中的にシステムを管理するのではなく，個々のエージェントが必要な情報 (例えばその他のエージェント情報) を自分で収集し，さらにそれを利用して自律的に判断や行動をするこで，要求にボトムアップに応える仕組みである．これにより上記の集中管理の課題に対して，以下の形で解決が期待できる．

(a) 収集データが増加しても各エージェントがそれぞれ必要なときに必要な情報を収集し，さらにそれを利用して計算処理をするため一箇所に情報を集約する必要はない．

(b) 情報の機密性に関しては，エージェントが分散して情報を保持するため，情報漏えいのリスクを減らすことができる．さらにエージェント間で取引や交渉をするため，企業の技術を関係者以外に公開する必要性もない．

(c) 通信帯域の圧迫に対しては，直接エージェント間の通信 (例えば M2M [31]) により通信帯域の圧迫を抑えられる．さらにエージェントが自分に必要なエージェントに問い合わせることでサーバ間との通信しない分，応答を速くできる．

そのため，実現に向けた研究が進められている．

2.3 関連研究

マルチエージェントシステムを利用した研究は様々な観点 (例えば集中的か分散的かに関わらず) からなされている．ここでは特に，複数のエージェントが同じ目的・目標を持ち，エージェント間で協力して組織やチームを編成する「協調問題解決」と，個々のエージェントがそれぞれ異なる目標を持つシステムあるいは組織において安定した相互バランスを維持する「交渉と均衡化」の２つに関する関連研究とその課題，また本研究との違いについて述べる [28, 32, 33]．

2.3.1 (分散) 協調問題解決に基づく研究

分散協調問題は，複数のエージェントが相互に連携し，ある特定の目的 (課題の解決やタスク処理) の達成を目指す研究である [34]．ここでは，結果共有，タスク共有，チーム編成問題，分散探索に分類し，それぞれの観点から関連研究の概要とその課題について議論する．

結果共有方式

結果共有方式は，各エージェントが自分の判断で個々に学習した結果を共有 (持ち寄り) し，他のエージェントが得た途中結果やこれから実行しようとする行動を把握することで，自分のこれからの行動内容を修正あるいは学習し，自分の計画とともにシステム全体の目的が達成されるように振る舞う方式である．分散環境におけるエージェントは，その他のエージェントの学習状況や振る舞い，状態といった個々の情報，システムの全体の負荷状況や稼働状況といった全体の情報を個別には得られないため，結果共有方式では相手の進捗状況や計画を随時メモリ上で共有することでその課題を解決している．代表的な研究としては，黒板モデルを利用した Hearsay-II 音声認識システム [35, 36] がある．一方で，本手法は，大規模な分散環境を想定した場合，途中結果を共有するため黒板 (共有メモリ) へのアクセス集中がボトルネックになる．

タスク共有方式

タスク共有方式は，共通の目的を達成するための負荷や要求を複数のエージェントに分散して割当てる方式である．つまり，要求されたタスクを副問題 (サブタスク) に分類し

(あるいは予めサブタスクに分類されていることもある)，そのサブタスクの割当戦略が重要な課題となる．この方式を応用した代表的な研究として，コントラクトネットプロトコル (Contract Net Protocol，契約ネット)(以後，CNP) があげられる [37]．CNP とは，分散環境におけるエージェント間の契約交渉を，個々のエージェントが相互通信することで解決するための手法である [37, 38, 39, 40]．

このモデルでは，要求タスクを割当てるマネージャと，提案されたタスクを選択し処理をするエージェントによって構成され．マネージャはエージェントに対してタスク処理の交渉 (提案) をする．これによりタスク処理の失敗を減らすことができる．CNP は以下のように交渉，契約を結ぶ．

1. マネージャは存在する全エージェントに対して，タスクを広報する．(タスクアナウンスメントフェーズ)
2. 提案を受けたエージェントは入札を希望するタスクを自分が持つ指標 (例えば獲得報酬，処理時間) に基づき決定し，その意思をマネージャに通知する．(入札フェーズ)
3. マネージャは実際にタスクを割当てるエージェントを，自分がもつ指標 (例えば．エージェントの能力，処理時間) により選択し，その結果を通知して契約を結ぶ．(契約フェーズ)
4. 契約したエージェントは割当てられたタスクを処理し，処理の完了と結果をマネージャに通知する．(タスク完了報告フェーズ)
5. マネージャは，上記の行動を繰り返す．

CNP の交渉の流れを図 2.3 に示す．この手法の特徴は，相手のリソース情報を必要とせず，通信できればタスクを適切に (少なくとも初期状態よりは適切に) 割当てられる点である．

CNP は，各エージェントの能力や状態の情報を保持する必要がない点で優れ (例えば [41, 42])，分散環境では，効率的な割当ができると考えられる．しかし，CNP はその割当 (広報) 対象を，全エージェントにしており，エージェントの増加に伴い広報のコスト (例えば時間) が増加するため，大規模な環境では割当の完了や通知に遅延が発生する可能性もある．さらに，エージェントの増加によってタスクの依頼メッセージも増加するため，入札フェーズから契約フェーズにおいて契約するエージェントを発見する時間がかかり，システム全体の効率が低下するおそれもある．また，負荷の高い状況では十分な入札メッ

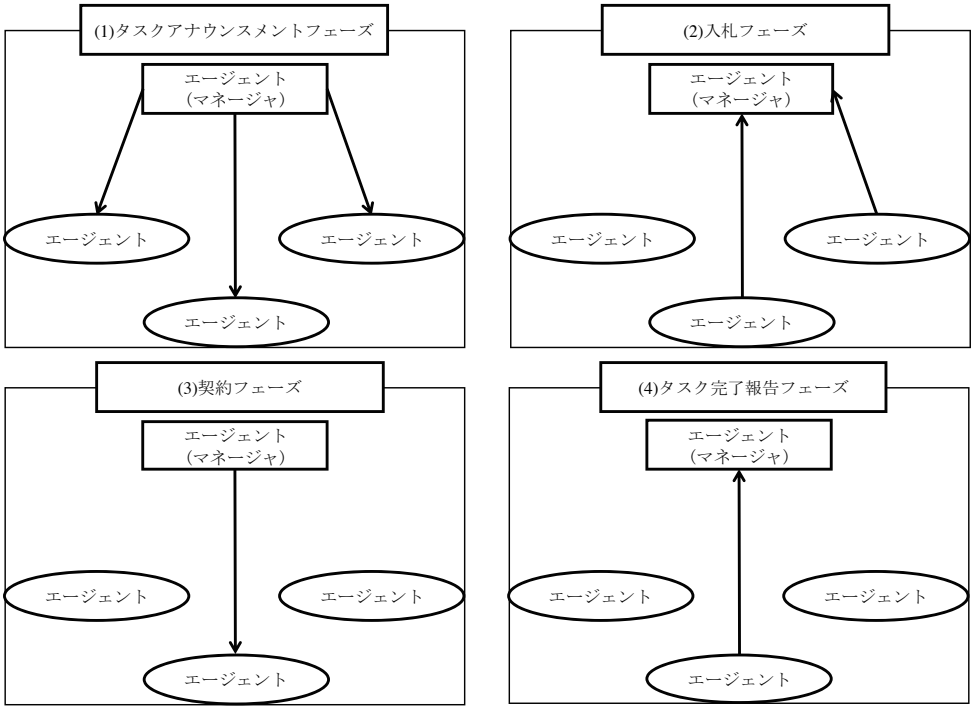

図 2.3: CNP の概念図

セージが得られず，システムの性能が引き出せない [43]．また，これに対し，メッセージを制限するなどの工夫もあるが (例えば [44])，ボトルネックが通信量にあることを想定している．現在は通信帯域は広く，ボトルネックも各ノードでの処理の負荷へ移り，想定する状況が変わってきている．さらにタスクアナウンストメントの順序にも影響するという課題もある．本研究のモデルは，タスクの割当依頼を全体に広報するのではなく，信頼できる一部の相手のみに依頼する．そのためエージェントが増加しても，CNP のように全体と通信をする必要がなく，上記で述べた広報のコストや探索コストに関する課題を解決している．

CNP の他に，エージェント間のつながり (構造) を予め定義し，要求されたタスクを割当てる資源割当の研究もある．例えば，エージェント間の構造を木構造とした研究では，タスクを下層のエージェントにどのように割当てるかに焦点をあてて研究している [45, 46]．この研究では，下層のエージェントのタスク処理状況や割当状況を学習し，タスクを渡すべきエージェントを学習し，システムとしてのタスク処理の効率を向上させている．また，Sherief Abdallah and Victor Lesser[46] では下層への割当方法のみが勘案されていたが，Ryota Katayanagi and Toshiharu Sugawara[47] では，エージェントのネットワーク構造を動的に再構築する仕組みを導入し，処理能力に余裕のあるエージェントに新たにリ

ンクを生成する手法を導入している．Kazuki Urakawa and Toshiharu Sugawara[48] では，リンクの生成ではなく，エージェントにかかる負荷状況に応じてリンクを切断してタスク処理の効率化と負荷分散を実現させた．また，木構造以外のネットワーク構造として，複雑ネットワーク [49] を想定した研究もある．

これらの研究は，予め想定したエージェント構造を前提としているが，以下の 2 つ課題が挙げられる．まず，そのエージェント構造が常に最適であるとは限らない点である．実際のシステムではエージェントの能力や数，さらにシステムにかかる負荷は変化する．そのため，その変化に応じた組み合わせや連携に組み直す必要がある．次に，予め決定した構造から変化に対応しているが，元の構造が木構造でのみを想定しているため，考察が限定されている点である．想定するシステムによって前提となるエージェント構造は異なること (例えば完全グラフ) が考えられる．これらの課題に対し，本研究ではエージェント間のつながりをボトムアップに形成し，適切なつながりをベースとしたタスク割当を実現している．

チーム編成問題

タスク共有方式では,「負荷分散して効率的かつ効果的にエージェントにタスクを割当てる」という観点で評価が行われており，エージェント間の構造に関してはあらかじめ規定されているかもしくは対象ではなかった．しかし，背景でも述べた通り，近年のサービスシステムはオープンになっており，ネットワーク構造を予測して決定することは難しい．そこで，構造をエージェント同士が連携するチームあるいは提携とよび，チーム編成問題，提携形成問題の観点からエージェント間の構造を構成・決定する研究がされている [50]．本論文の目的もこの項目に該当し，効果的な割当をするために，エージェントが協調してチームを編成するモデルである．以下に従来の研究とその課題について述べる．

Onn Shehory and Sarit Kraus[51] ではチーム編成 (つまりエージェントの組み合わせ) のパレート最適解を多項式時間で求めるアルゴリズムを提案している．また，Jingan Yang and Zhenghu Luo[23] 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，複数タスクに対応したチーム編成手法を提案した．これらの研究はエージェントの状態を常に把握できると仮定している．しかしエージェント数の増加にともないその状態の収集は時間がかかり，さらにそれらの情報量は大きくなるため，それを保持し続けるのは困難になる．この課題に対し，本研究ではエージェントに連携相手を制限するスコープを導入すことで，エージェントが持つ情報量を減らしても効率的にタスク処理ができることを示して

いる．Carlos Merida Campos and Steven Willmott[52] では要求されるタスクが予め既知で限られている環境において効率的なチーム編成手法を示した．しかし，タスクに対するエージェントの行動が固定的であり，エージェントは行動を適切に学習をしない．そのため，この手法では，タスクが変化する動的な環境への対応は困難である．

Thomas Genin and Samir Aknine[53] のエージェントは，他のエージェントが受け入れたタスクを記憶し，それに基づく行動決定方法が用いられており，リーダ (要求を割当てるエージェント) として行動するエージェントは過去の情報を参考にチームのメンバ (依頼を処理するエージェント) を決定する．この方法では，メンバがどのタスクを受け入れるかのみを学習しており，自分に適した役割は学習しない．実際のシステムでは，システムの必要とするリーダとメンバの数は負荷状況によって異なる．例えば負荷の小さいときは要求が少ないためリーダの数は少なくても良いが，一方で，負荷が大きくなるほど必要なリーダの数は増える．しかもこれらの数は，システムにかかる負荷が予測できないため予め設定することは不可能である．そのため，リーダやメンバの役割を状況に応じて学習する必要がある．本研究では，エージェントに予め役割を設定せず，エージェントが個々に自分の役割を学習し，状況に応じた役割を選択することが可能である．

Dai Hamada and Toshiharu Sugawara[54] では，Thomas Genin and Samir Aknine[53] の手法に改善を加え，報酬に基づく強化学習を導入することで，エージェントが役割を学習できる手法を提案した．それに加えて，チーム編成の際の協調相手を学習することで効率的なチーム編成手法を提案している．Somchaya Liemhetcharat and Manuela Veloso[55] ではエージェントが他のエージェントとタスクの相性を示すシナジーグラフを持ち，そのグラフを学習することでチーム編成の効率化を達成している．Christian Guttmann[56] では，確率分布で変動するエージェントの性能を表すパラメータを導入し，タスク実行に要する時間が確率的に変動するモデルを提案した．タスクの処理時間における不確実性を考慮に入れながらもチーム編成を効率化できることを示しており，現実にも起こりうる可能性のあるエージェントの能力の変動をモデルに加味している．これらの研究では，エージェントは他の全エージェントの情報 (リソース) を持っており，チーム編成時にそれらを参照して最適なエージェントを発見する．しかし，これらの手法では前述したエージェントに関する情報が十分に把握できない状況 (プライバシーやセキュリティ) には対応できない．そこで本研究では，エージェントの能力を推定学習する手法によってこの課題を解決する．

分散探索方式

マルチエージェントシステムを問題空間の探索問題へと定式化した研究がある．特に，分散制約充足問題 [57] はこの項目に該当する．分散制約充足問題とは，制約充足問題 [58] で設定される変数と制約を分散した複数のエージェントが自律的に管理し，協調して制約を満たす変数の値を決定する問題である．代表的な応用例として，個人のスケジュールに合わせて会議の予約を自動的に実現する会議スケジューリング問題 [59] がある．

この問題に焦点を当てた研究としては，非同期バックトラックアルゴリズム [57] を利用して各エージェントが個々に持つ制約をみたす解を効率的に求める手法が提案されている．この手法は，個々のエージェントが持つ制約 (エージェントの満足度や閾値) がある場合，その満足度を満たす割当を見つけるためのアルゴリズムである．この手法は，各エージェントの満足度を満たす割当を繰り返し，全体が満足するまでこれを繰り返す．そのため，大規模になるほどその探索に時間がかり，解を発見するまで通信するためその通信量も増加する．さらにこの手法を適用する対象は静的な環境を想定しているためヒューリスティックなどを導入して，動的な環境を考慮しつつシステム全体の効率を向上する手法が必要となる．本研究では，最適解ではないが，十分に効率を得られるヒューリスティックな手法としてエージェントの学習を提案した．

2.3.2 交渉と均衡に基づく研究

分散協調問題の枠組みでは，個々のエージェントは共通の目的 (例えばタスク処理) を持ち，それを複数のエージェントに割当てる，あるいは協調して解決する手法が対象であった．一方で交渉と均衡に基づく研究では，個々のエージェントの目的が異なる場合に焦点が当てられ，いかにシステム全体として安定した状態を保てるかということが重要となる．この項目はゲーム理論 [60, 61] が当てはまる．

ゲーム理論では，これまで述べてきたエージェントに相当するプレイヤなるものが存在し，プレイヤの行動とそれに対応する利得を元に数理的な枠組みでモデル化し，そこでの交渉プロトコルの研究がされている．ゲーム理論では，プレイヤ自身の利得を最大化する合理的な行動の選択を前提としている．この合理的な行動は前述のエージェント定義でも議論した合理性の基本的な考え方である．一方で，プレイヤ同士が合理的な行動を取るとお互いに損をするケースもあるため，お互いの利得を可能な限り大きくする行動を取るために交渉し，状況によっては譲歩することもある．具体的な研究として，(繰り返し) 囚人

のジレンマゲームを利用した研究 [62]，最後通牒ゲームを利用したプレイヤの報酬交渉モデル [63] やエージェントの提携形成に着目した研究 [64] がある．これらの研究では，ゲーム理論に基づく数理的な枠組みを利用し，安定した状態 (例えばエージェント間の協調) を維持するプレイヤ戦略学習が提案されている．実際のシステムは処理すべきタスクが発生するため，これらのゲーム理論的アプローチで得た知見 (協調関係の維持) を，チーム編成問題あるいはタスク共有方式へと適用し，タスク処理の性能 (例えば処理数) も調べる必要がある．本研究ではゲーム理論で得られた知見をエージェントの行動戦略へ応用し，エージェントの安定した提携を実現し，それがタスク処理の効率向上に寄与すること示している．

2.4 本章のまとめ

本章はエージェントとマルチエージェントシステムの定義をまとめた．次に，マルチエージェントシステムを利用した従来研究を以下の 2 つに分類してその概要と課題を議論した．

(a) (分散) 協調問題解決に基づく研究

(b) 交渉と均衡に基づく研究

(a) では，分散環境での効率的なタスク処理を実現するために，エージェントが学習結果を共有する手法や，CNP を利用した割当手法が提案されていた．一方で，予めエージェント構造や役割を固定した静的な手法もあり，エージェント数や要求タスク数の増加あるいは変化する動的な環境に必ずしも対応できるとは限らなかった．

(b) ではゲーム理論をベースにしたモデルを利用し，一定の均衡を創発させ，それを安定的に保つ手法が提案されていたが，協調や提携の形成をメインに議論がなされており，エージェント提携がシステムに要求されたタスクを処理できるかを調査する必要があった．

上記の課題を考慮し，本研究では，まず，要求 (タスク) が時々刻々と連続発生する環境を想定 (次のタスクを予測できない) し，エージェントがチームを組んでタスクを処理するタスク割当チーム編成モデルを導入する．このモデルではエージェントは役割や協調相手を自分で選択するため，あらかじめそれらを決定する手法 (例えば CNP や木構造) とは大きく異なる．さらに本研究は，エージェントに役割と協調相手を学習する機能を導入す

る．これにより自分に適した役割と協調相手をチーム編成の結果から学習し，分散環境でのタスク処理成功率の向上を示す．

第3章　エージェントのタスク割当チーム編成モデル

本研究ではエージェントへのタスク割当問題をチーム編成モデルとして捉える．ここではまず，本研究におけるエージェントとタスクを定義する．ここでのエージェントは，前述のエージェント定義を元に，本モデルに適用したものである．そのため，エージェントは合理性を基本とし，タスクを処理する目的で行動する．また，タスクはエージェントが処理するサービス要求を抽象化したものである．実際のサービスではタスクは様々な要素で構成され，それらすべてを処理して初めてサービスが提供される．そのため，それらの要素をここではサブタスクと定義し，複数のエージェントがそれらを処理するためにチームを編成するモデルとした．最後にそのチームを編成するエージェントの行動を，エージェントの役割ごとに分類して述べる．

3.1　はじめに

これまでIoTを利用したサービスの実現に向けた研究は多くある．その一つにfogコンピューティングシステム [65] がある．これはノードに相当するエッジに処理を分散し，必要に応じて情報をサーバに統合させることで，デバイスの増加に伴う通信量の削減を実現し，処理のリアルタイム性を目指している．しかし従来のシステムでは収集する情報量を削減しているが，サーバやクラウドで処理をしており，集中的な制御が必要であった．そこで，本研究は，分散的なシステム制御の実現を目指し，エージェントが要求されたタスクに対してボトムアップに連携し処理するシステムをモデル化した．本モデルでは，処理能力をもつあらゆる主体 (アプリケーション，デバイス) を総じてエージェントと呼ぶ．また，システムへ要求・追加されるサービスをタスク，それを構成する要素をサブタスクとする．タスクに含まれた各種サブタスクを処理するためにエージェントが自律的に振る舞い，チームとしてタスクを処理する．このモデルでは前述したfogコンピューティングシステムのように，分散的な処理を抽象化したものであるが，サーバやマネージャのようにシステム全体を把握する主体を必要とせず，エージェントの各々の自律的判断によりシ

ステムが自動的にタスクを処理するモデルである．

3.2 エージェント

システムに存在する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分の役割と能力に相当するリソースをもつ．役割は，要求を割当てるエージェント (メンバ) を主導的に集めるリーダと，要求を処理するメンバの 2 種類がある．これらの役割はあらかじめ固定的に決定せず，各エージェントが自分に適した役割を自律的に学習し選択する．役割選択については後述する．

エージェント i のリソースを $H_i = (h_i^1, \ldots, h_i^p)$ と表現する．ここで p はリソースの種類を表し h_i^k は非負の実数とする．エージェントは自分のリソース H_i の値が大きいほど高い処理能力を持つことを表す．例えば，エージェント i のリソースが $H_i = (3, 5, 2)$ の場合，リソースの種類は 3 種類であり，最も能力があるリソースは 2 番目となる．また，エージェントは合理性に基づき，得られる利得の最大化を目的に行動する．本モデルでは，後述するタスク処理時に得られる報酬獲得の最大化を目指して行動する．

3.3 タスク

システムへ追加するタスクを T とおく．本モデルにおいて要求されたタスクは環境に用意されたタスクキュー Q で管理する．Q には一定時間毎に $\lambda \geq 0$ 個のタスクが追加される．この λ はシステム負荷 $(workload)$ を表す．そのため，λ の数が大きいほど処理すべきタスクが多くなり，システムにかかる負荷が大きい環境となる．タスク T は，それを構成するサブタスクの集合 S_T と，処理成功時に得られる総報酬 $U_T \geq 0$ のペアで $T = (S_T, U_T)$ と表す．また，T のサブタスクの集合 S_T は $S_T = \{s_1, \ldots, s_l\}$ とし，s_i は T に含まれる各サブタスクを表す．

サブタスク s_i には処理に必要な要求リソースが定義されており，$R_{s_i} = (r_{s_i}^1, \ldots, r_{s_i}^p)$ と表す．エージェントの持つリソースが対応する要求リソースを上回れば，サブタスクを処理できるものとする．この条件は，エージェント i と，サブタスク s に関して，

$$h_i^k \geq r_s^k \tag{3.1}$$

と表せる．例えば，$H_i = (8, 2)$ のリソースを持つエージェント i が存在し，また $R_{s_1} = (5, 1)$ の要求リソースをもつサブタスク s_1 と，$R_{s_2} = (2, 7)$ の要求リソースをもつサブタスク s_2

があるとする．i は要求リソースをすべて上回る s_1 は処理できるが，自分のもつリソースが 1 種類だけ下回っている s_2 は処理できない．また， エージェントは式 (3.1) を満たせばタスク T に含まれる全てのサブタスクを処理できる．

タスク T にふくまれる報酬 U_T は S_T に含まれるサブタスクのリソースの和として

$$U_T = \sum_{s \in S} \sum_{p=1}^{p} r_s \tag{3.2}$$

と表す．エージェントはこの総報酬から自分の貢献度合いによって報酬を得るものとするが，その詳細は後述する．

3.4 チーム

本研究では，処理すべきタスクに対し，あるエージェントの集合を編成する．ここではその集合およびその集合での各サブタスクのエージェントへの割当・処理を含めてチームと呼ぶ．チームに属するエージェントは予め決定せず，タスク要求のたびに動的に編成する．また，エージェントが同時に所属できるチーム数は一つまでと制限する．実際のシステムでも，エージェント間の距離が物理的に離れていて同時に作業ができない場合や，通信に時間がかかりタスクを同タイミングで処理できない場合が考えれられる．そのため今回はこれらを考慮した．

タスク T の処理を実行するチームを (G_T, σ_T, T) と表す．$G_T \subset A$ はタスク T を処理するエージェントの集合を表す．σ_T は割当関数であり，サブタスク $s \in S_T$ をチーム内のエージェント $i \in G_T$ に割当てることを $\sigma_T(s) = i \in G_T$ と表す．チーム編成は，

$$h_i^k \geq \sum_{s \in \sigma_T^{-1}(i)} r_s^k \quad (1 \leq \forall k \leq p), \tag{3.3}$$

を満たしたとき成功と判断する．つまり，タスクに含まれる全てのサブタスクの割当が完了したらチーム編成は成功したとし，同時にタスク処理が行われる．

3.4.1 チーム編成における役割と行動

エージェントは図 3.1 に示す行動を繰り返し，チームを編成しタスクを処理する．以下に各行動を説明する．

リーダの行動

初期状態 (役割選択と依頼)　リーダを選択したエージェントはタスクキュー Q の先頭からタスク T を取得する．このとき，キューにタスクがなければ再度役割選択を行う．タスクを取得したリーダは T に含まれるサブタスクのうち，自分で処理可能なものがあれば一つ選択し，自分に割当てる．残りのサブタスクについて，各サブタスクごとに L (ここで L は正の整数) 体メンバ候補を決定する．メンバ候補決定方法も後述する学習パラメータを利用する．リーダはメンバの候補に対し，担当させるサブタスクの情報が含まれたチーム加入依頼メッセージを送る．ここで L は，チーム参加依頼提案数と呼び，サブタスクの処理を担当するエージェントを確保する確率を上げるために冗長的に依頼する数である．分散環境では，相手の状況に関する情報を常に収集・把握できないため，依頼した相手が処理中，あるいは別の依頼を受けている等を理由に必ずしも依頼が通らないことがあり，それに対するリスクヘッジの必要がある．この値が大きければ大きいほどサブタスク処理の割当成功率は向上するが，その分，各エージェントが受ける依頼が重複する可能性は大きくなるため，チーム参加依頼提案数とメンバの獲得確率 (依頼の成功率) はトレードオフの関係である．本モデルでは実験で比較的高い成功率を示した $L = 2$ を採用した．このとき，リーダとチーム加入依頼メッセージを送ったメンバ候補を仮チームとよび，G_T^p と表す．依頼を終えたリーダは次のチーム編成結果判定と通知行動へ移る．

チーム編成結果判定と通知　リーダは，メンバ候補から参加/不参加の返答メッセージを受け取ったのち，その結果に基いてサブタスクを割当てる．このとき，リーダと参加受託したメンバ候補の集合を G_T^0 と表記する．これは，G_T^p から不参加のメンバ候補を除いた集合である．ここでサブタスク毎に L 体のメンバ候補にチーム加入依頼メッセージを送るため，1 つのサブタスクについて複数のメンバ候補から参加受託の返答を受け取ることもある．この場合にも後述する学習パラメータを元に，複数のメンバ候補のうち 1 体を，実際にサブタスクを割当てるメンバとする．また，受託されないサブタスクがあるときは，G_T^0 のメンバ候補のうち，実行可能かつ，未だサブタスクを割当ていないエージェントをメンバとして選択する．その後，式 (3.3) を満たした場合，G_T^0 のメンバ候補に対して，チーム編成成功の通知と担当するサブタスクを送り，受信したメンバは送られたサブタスクの処理を開始する．サブタスクの割当がないメンバ候補には断りのメッセージを送る．また，式 (3.3) を満たすことができなければ，すべての G_T^0 のメンバ候補にチーム編成失敗の通知を送り，タスクを破棄する．ここで，選択したタスクを破棄せずにキューに戻すなどの設定もあるが，ここではチーム編成の成功率に焦点を当てるため破棄としてカ

ウントする．

タスク処理と報酬　このステップでは，チーム編成に成功したリーダが，自分に割当てたタスクがあった場合はそれを処理し，その他の場合は，すべてのサブタスクの処理が完了するまで待機する．すべての処理が完了したら報酬を分配した後チームを解散し，初期状態へと戻る．報酬の配分は，チームの学習の重要な戦略であり，別途述べる．

メンバの行動

初期状態 (役割選択と依頼選択)　初期状態でメンバの役割を選択したエージェントは，リーダから依頼されたサブタスクのうち自分が処理可能なものを一つ選択し，そのチームへの参加を表明するメッセージを返す．参加しない依頼に対しては断りの返答メッセージを送る．その後，編成結果の受取へ移る．このとき，依頼がなかったメンバは初期状態へ戻り再度役割選択を行う．

編成結果の受取　メンバはリーダからチーム編成結果を受け取り，チーム編成が成功し，サブタスクが割当てられればそれを実行する．一方，チーム編成が失敗もしくは割当がなければ何もせず，初期状態へ戻る．

タスク処理と報酬　メンバもリーダと同様，ここで自分に割当てられたサブタスクを処理し，報酬を獲得したのち初期状態へ戻る．

3.4.2　モデル全体の処理の流れと時間経過

図 3.1 で示したエージェントの各行動は 1 tick 毎に一つ行う．ここで，1 tick とは本研究で用いる時間単位である．単位時間あたりに，システムにはタスクの追加と各エージェントの行動が行われ，これらの繰り返しによりシミュレーションを進めている．またエージェントがやり取りするための各メッセージは，1 往復で M tick，与えられたサブタスク実行の所要時間は全サブタスク共通で K tick とする．したがって，(1) リーダがチーム加入依頼メッセージを送り，その応答を受け取るまでと，(2) チーム編成成功とタスクの完了通知のメッセージにそれぞれ M tick, また (2) の間でサブタスクの処理時間に K tick かかり，全体として $2M + K$ tick かかるものとする．チーム編成が成功しサブタスクの割当があった場合を図 3.2 に示す．

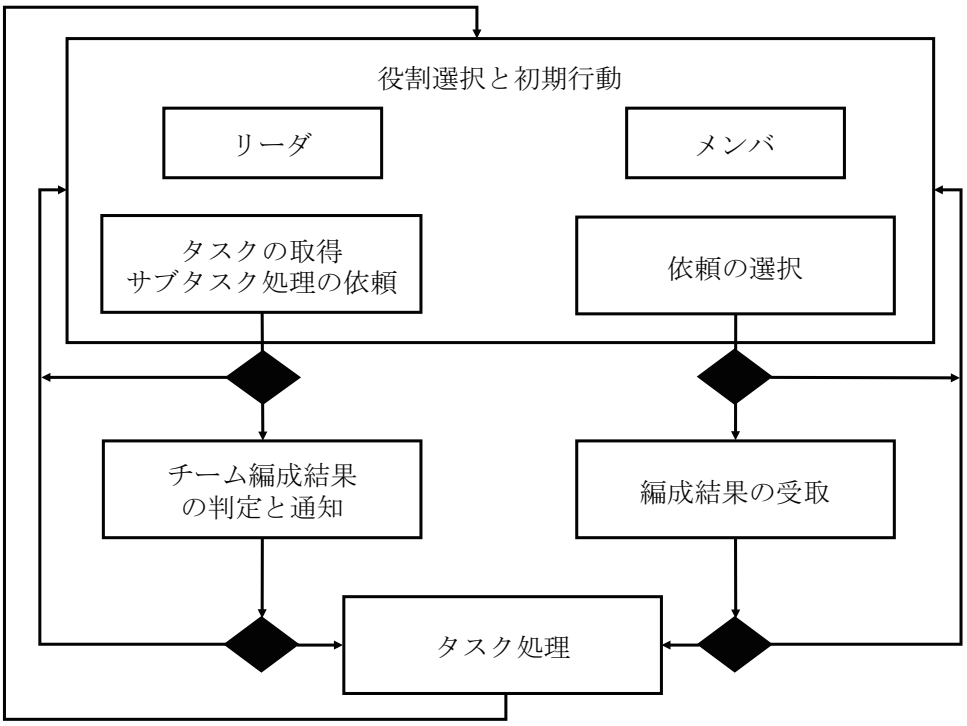

図 3.1: エージェントの行動

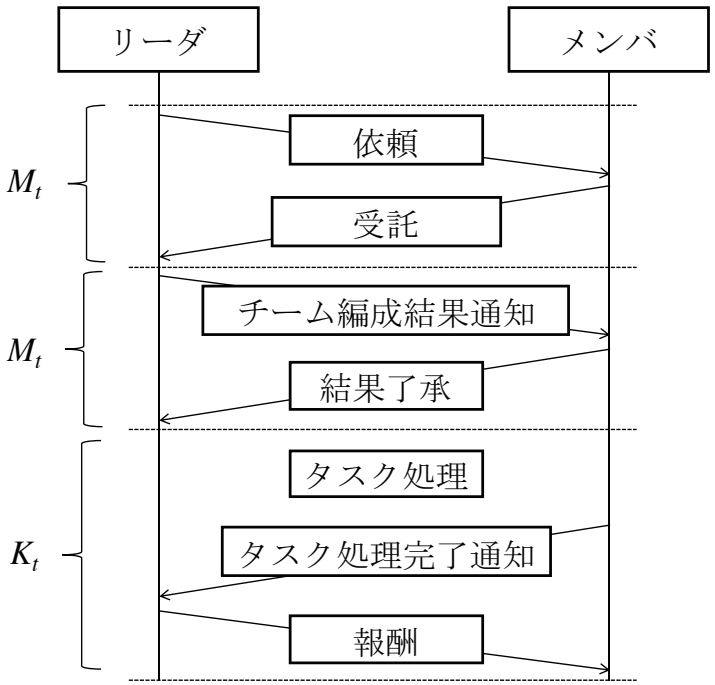

図 3.2: チーム編成成功時に要する時間

3.5 本章のまとめ

本章では，エージェントとタスクを定義し，エージェントのタスク割当チーム編成モデルについてまとめた．本モデルは，予めエージェントの役割やチームを設定せず，後述するエージェントの学習によってそれらを学習する．さらにこのモデルはシステム全体を管理するマネージャやサーバが存在しておらず，システム状況を集中的に制御しない分散的なモデルである．以降の提案手法は本モデルを用いたものであり，エージェントのタスク処理数を効率の指標とする．

第4章　エージェントの役割及び協調相手の学習とリソース推定手法

本章では，エージェントのリソースが相互に想定できない未知環境を想定し，そこでのチーム編成を実現するために，エージェントの学習とリソースの推定学習手法を提案した．ここではまず，リソースが推定できない未知環境を考慮すべき理由を再度まとめる．次にエージェントの学習について述べる．この学習によりエージェントは自分の役割と協調すべき相手を学習し，動的かつ効率的にチームを編成できる．その後，リソースが推定できない未知環境へ対応するためのリソース推定手法を提案する．この手法は予め相手のリソースが収集できなくても，実績に応じて学習・更新する手法である．最後に評価実験によって本提案手法の有効性を示す．ここでの手法および考察は論文 [66] で議論したものである．

4.1　はじめに

近年，情報の重要性と機密性が高まっている．例えば，個々のアプリケーションやデバイスにユーザの情報を保存することがあるが，それらはプライバシーの観点から公開されることは望ましくない．また，サービスを提供する企業も戦略的に，あるいは技術流出を防ぐために，その技術のすべてを公開することはない．一方で，エージェントの能力も予め想定した能力や状況によって変化する場合がある．例えば想定以上の負荷により，エージェントが自分の能力を最大限発揮できない場合が考えられる．こういった環境では，エージェントが相手の能力を把握できない，あるいは予め与えた能力に関する情報が環境の変化や自分の負荷状況によって異なることがある．そのため能力を把握できない未知な環境での対応が必要となる．従来の研究ではリソースをあらかじめ既知とした環境を想定するものが多いが，上記のような問題を考慮すると，実際のシステムに適用できるとは限らない．そこで本手法は，リソースが未知な環境へ対するアプローチとして，エージェントの学習とリソース推定手法の導入をした．エージェントの学習では，エージェントがリーダになるべきか，メンバになるべきかの役割学習とチームを組むべき協調相手を

学習し分散環境での動的なチーム編成を実現した．リソース推定手法では，相手の能力を実績に応じて学習することで相手の能力が未知であっても学習したリソースによってチームが編成できることを示した．

4.2 エージェントの学習

本研究のエージェントは合理的であると仮定し，得られる報酬を最大化を目的に行動する．このためにはチーム編成の成功率を上げるとともに，獲得できる報酬を大きくする行動選択が必要である．この選択をするために欲張り度 (Degree of Greediness)，提案受託期待度 (Expected Ratio of Acceptance of Team Solicitation)，報酬期待度 (Expected Member Utility Division Ratio) の 3 つの学習パラメータをエージェントに定義し，エージェントの役割とチーム編成時の協調相手の選択 (誰に提案をするか，どのチームに参加するか) を学習する．これらの学習により，エージェントの報酬を最大化しようとする合理的な行動が，ボトムアップなチーム編成の成功率を高める．

4.2.1 欲張り度と報酬分配

欲張り度は，リーダがチーム編成成功時に受け取る自分の報酬を学習するもので，リーダ i の欲張り度を g_i と表す．リーダ i がチーム (G_T, σ_T, T) の編成成功時に受け取る報酬 u_i は，

$$u_i = U_T \times g_i \tag{4.1}$$

とする．その後，チームのメンバ $j \in G_T \setminus \{i\}$ への報酬は，割当てられたサブタスクの報酬比に従い

$$u_j = (U_T - u_i) \times \frac{\sum_{t \in \sigma_T^{-1}(j)} \sum_{p=1}^{p} r_t}{\sum_{s \in T \setminus \sigma_T^{-1}(i)} \sum_{p=1}^{p} r_s} \tag{4.2}$$

として分配する．$\sum_{t \in \sigma_T^{-1}(j)} \sum_{p=1}^{p} r_t$ はエージェント j が実際に処理したサブタスクの要求リソースの和である．また，$\sum_{s \in T \setminus \sigma_T^{-1}(i)} \sum_{p=1}^{p} r_s$ はチームのメンバが処理したサブタスクの要求リソースの和である．欲張り度の値が大きいとリーダの獲得報酬は大きくなるが，メンバに分配する報酬も小さくなる．その結果，チーム参加を表明するエージェントが減り，チーム編成成功率が低下する（リーダとしての獲得報酬が 0 になる）．そのため，欲張り度の値をチーム編成の結果により調整し，以下の式で更新する．

$$g_i = \alpha_g \times \delta_{success} + (1 - \alpha_g) \times g_i \tag{4.3}$$

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

4.2.2 提案受託期待度

提案受託期待度とは，エージェントが役割としてリーダを選択したときに，チーム加入依頼メッセージを送るメンバ候補の受託可能性の期待度を表す．この値が高いほどチーム加入依頼メッセージを受託しやすいエージェントと判断でき，リーダはこのようなエージェントに優先的に提案することでチーム編成成功率の向上が期待できる．リーダ i の，あるエージェント j に対する提案受託期待度を $e_{i,\,j}$ と表す．i は j に対する提案受託期待度 $e_{i,\,j}$ をチーム加入依頼メッセージの結果により以下の式で更新する．

$$e_{i,\,j} = \alpha_e \times \delta^j_{accept} + (1 - \alpha_e) \times e_{i,\,j} \tag{4.4}$$

ここで，α_e は提案受託期待度の学習率で 0 以上 1 以下の定数とする．また δ^j_{accept} は，チーム加入依頼メッセージが受託された場合 1，拒否された場合は 0 とする．リーダは提案受託期待度と ε-*greedy* 選択を用いて，仮チームのエージェントや，タスクを割当てるメンバを選択する．詳しいアルゴリズムについては第 4.3.2 節，第 4.3.3 節で述べる．

4.2.3 報酬期待度とチーム加入依頼メッセージ選択

報酬期待度は，エージェントが役割としてメンバを選択したときに，リーダから獲得できる報酬の期待度合いを表すものである．この値が大きいほど，チームに参加した報酬が大きくなると期待できるため，合理的であるエージェントはより多くの報酬を獲得できるであろうチームへの参加を試みる．メンバ i のリーダ j に対する報酬期待度を $d_{i,\,j}$ と表す．ここで，報酬期待度を用いたチーム加入依頼メッセージ $\tilde{m}$ の選択を以下の式で表す．

$$\tilde{m} = \arg\max_{m \in M} \sum_{s \in S(M)} \sum_{p=1}^{p} r_s \times d_{i,\,l(m)} \tag{4.5}$$

ここで M はメンバが自分に受け取ったすべてのチーム加入依頼メッセージの集合とし，$l(m)$ は，i にメッセージ m を送信してきたリーダを表す．$S(M)$ は，i が，受信したメッセージ m において要請されたサブタスク集合を表す．式 (4.5) はメンバは受け取ったチーム加入依頼メッセージにおいて，最も大きいリーダからのメッセージを優先して選択する

ことを意味する．また，メンバによるメッセージ選択は報酬期待度と，ε-$greedy$ 選択を用いる．報酬期待度はリーダから得られた実際の報酬に基づき以下の式で更新する．

$$d_{i,\,j} = \alpha_d \times \frac{u_j}{\sum_{s \in S(\tilde{m})} \sum_{p=1}^{p} r_s} + (1 - \alpha_d) \times d_{i,\,j} \tag{4.6}$$

ここで，u_j はチーム編成に成功しリーダから分配された報酬である．また，$\sum_{s \in S(\tilde{m})} \sum_{p=1}^{p} R_s$ は，選択したメッセージ $\tilde{m}$ のサブタスクの報酬である．もし，チーム編成に失敗した場合は，タスクを割当てられないため，u_j の値は 0 とする．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．

4.2.4 役割選択

エージェント i は初期状態において役割を選択する．その際，リーダあるいはメンバとしての期待報酬を比較して獲得できる報酬が高いと期待できる役割を選択する．i のリーダとしての期待報酬 E_i^{leader} は，タスクキューの先頭のタスクと欲張り度 g_i を元に決定し，以下の式で求める．

$$E_i^{leader} = U_T \times g_i \tag{4.7}$$

i のメンバとしての期待報酬 E_i^{member} は，自分に提案されたサブタスクと報酬期待度を元に決定され次の式で求める．

$$E_i^{member} = \sum_{s \in S(\tilde{m})} \sum_{p=1}^{p} r_s \times d_{i,\,l(\tilde{m})} \tag{4.8}$$

その後，エージェントは上記 2 式を比較し，$E_i^{leader} \geq E_i^{member}$ であればリーダの役割を選択し，その他の場合はメンバの役割を選択する．もし，役割選択のときに，エージェントがチーム加入依頼メッセージを受け取っていない場合は E_i^{member} の値を 0 とする．

4.3 リソース推定手法

ここでは他のエージェントのリソース情報を推定，学習する仕組みとそれを利用したエージェントのメンバ候補選択方法と割当方法について述べる．

4.3.1 リソース推定パラメータとその学習

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を持ち，$\tilde{H}_j = (\tilde{h}_j^1, \ldots, \tilde{h}_j^p)$ と表す．リソース推定パラメータは初期値を 0 とし，チーム加入依頼メッセージが受託されたとき，以下のように更新する．

1. リーダ i は j にサブタスク s とともにチーム加入依頼メッセージを送る．
2. j がそのメッセージを受託した場合，i はサブタスクの要求リソース R_s とリソース推定パラメータ $\tilde{H}_j$ 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 がメッセージを拒否したとき，リソース推定パラメータ $\tilde{H}_j$ は更新しない．

項目 3 はメッセージを拒否した理由に，他のリーダからのメッセージを受け入れた場合が考えられ，必ずしもエージェントの持つリソース不足による拒否とは限らないためである．

リソース推定パラメータはエージェントがリーダとして仮チームのメンバを決定する際に用いられる．従来の手法 [54] であれば，エージェントが持つリソースを参照していたが，本研究ではこのリソース推定パラメータを用いてメンバ候補を決定する．詳しいメンバ候補選択方法は次節で説明する．

4.3.2 メンバ候補決定方法

ここではタスクに含まれるサブタスク処理を依頼するメンバ候補選択方法を述べる．メンバ候補選択アルゴリズムを Algorithm 1 に示す．

まず，リーダの役割を選択したエージェント i はキューからタスク T を選択し，自分が処理できるサブタスク s_i を自分に割当て，$T' = S_T \setminus \{s_i\}$ とする．その後，T' 中の残りのサブタスクの処理を依頼する仮チームのメンバを ε-*greedy* 選択を用いて決定する．ここでは前述の通り，一つのサブタスク s につき，チーム参加依頼提案数 L 体分メンバを決定する．ε の確率では，リーダ i はサブタスク s の処理依頼をするエージェント L 体をランダムに決定する．シミュレーション開始時もリソース推定パラメータや各種学習パラメータが初期値であるため，ランダムにメンバを決定する．また $1-\varepsilon$ の確率で，リーダ i は一つのサブタスク s の処理依頼をするエージェントを，リソース推定パラメータおよび提案受託期待度を用いて次のように決定する．

Algorithm 1 リソース推定パラメータに基づく仮チームのメンバ選択方法

Require: T: タスク
K: エージェント集合
i: リーダ
L: 各サブタスク毎に選択するエージェント数 (チーム参加依頼提案数)
$S'_T \Leftarrow S_T \setminus \{s_i\}$ //i が処理するサブタスクを S_T から除く
$G^p_T \Leftarrow \emptyset$
S'_T の要素 s を要求リソースの和に基づいてソートする.
for all $s \in T$ **do**
  $l(s) \Leftarrow 0$
  $K \Leftarrow K \setminus \{i\}$
  for $j \in K$: $e_{i,\,j}$ を用いて選択する **do**
    if $\tilde{h}^k_j \geq r^k_s$ for $\forall k$ **then**
      $G^p_T \Leftarrow G^p_T \cup \{j\}$ //j を仮チームへ加える
      $K \Leftarrow K \setminus \{j\}$
      $l(s) \Leftarrow l(s) + 1$
      if $l(s) = L$ **then**
        break
      end if
    end if
  end for
  if $l(s) < L$ **then**
    a : K からランダムに選択したエージェント
    $K \Leftarrow K \setminus \{a\}$
    $G^p_T \Leftarrow G^p_T \cup \{a\}$ //a を仮チームへ加える
    $l(s) \Leftarrow l(s) + 1$
    if $l(s) = L$ **then**
      break
    end if
  end if
end for

1. リーダ i は T' のサブタスクを報酬 (要求リソース) の大きい順にソートし，報酬の一番大きいサブタスク s から順に依頼する．$G_T^p = \emptyset$ と初期化する．

2. 提案受託期待度の一番大きく，かつ G_T^p に含まれないエージェントを j とし，s の要求リソースと j のリソース推定パラメータの各要素を比較する．式 (3.1) (ただしリソース推定パラメータを用いる) を満たす場合，j を s の処理依頼をするエージェントとして G_T^p に加える．満たさない場合は G_T^p に加えない．

3. i は j の次に提案受託期待度の大きいエージェント j_2 の調査に移り，同様に条件を調べる．

4. 2 および 3 の試行をチーム参加依頼提案数 L 回繰り返す．送り先となる L 体のエージェントを発見できない場合，残りのエージェントをランダムに選択し，合計で L 体のエージェントに送る．

サブタスク s の担当エージェントを L 体決定した後，次のサブタスク s_2 の調査に移り上記を繰り返す．この行動は，i が選択したタスク T に含まれるサブタスクの数だけ繰り返す．このようにして選ばれたメンバとリーダの集合が仮チーム G_T^p となり，リーダは G_T^p のメンバに対して，割当てるタスク情報を含んだチーム加入依頼メッセージを送り応答を待つ．その後，チーム参加を表明したエージェントとリーダの集合が G_T^0 となる．リーダは G_T^0 のメンバに次に述べる割当関数 σ_T を元にサブタスクを割当てる．

4.3.3 割当関数 σ_T の決定方法

G_T^0 の各メンバに対して，リーダ i は以下のようにサブタスクを割当てる．i は一つのサブタスクにつき L 体のエージェントに対して提案するため，重複してチーム参加提案が受託される場合がある．この場合は，ε-$greedy$ 選択を用いて提案受託期待度の大きい方のエージェントにサブタスクを割当てる．また，全てのサブタスクを割当てた後，処理されないサブタスクが残っていた場合，G_T^0 内の実行可能，かつ提案受託期待度の高いメンバに対して順に割当てる．このとき (G_T, σ_T, T) が式 (3.3) をみたせばチーム編成は成功となる．ただし，メンバ候補決定時と同様，式 (3.3) の判定にはリソース推定パラメータを用いる．このようにして，実際にサブタスクを割当てたエージェントと，リーダの集合がチーム G_T となる．

4.4 評価実験：リソース推定パラメータの有効性評価

エージェントの学習とリソース推定パラメータを組み合わせた提案手法の有効性を評価する．そのために以下の比較手法と処理したタスク数の比較をした．ここではリソース推定パラメータを導入した提案手法を，リソース推定学習および役割学習を導入したチーム編成手法 (Team Formation Method with Resource and Role Learning) と呼び，以降 TFR^2 と表記する．

4.4.1 比較手法

本実験ではリソースを既知とした環境でチームを編成する既存手法 [54] とリソースは既知であるが，役割や協調相手を学習しない NoLearning 手法，コントラクトネットプロトコル (CNP) の 3 手法と比較した．これらの手法について以下に詳細を述べる．

既存手法 (RL 手法)

Dai Hamada and Toshiharu Sugawara[54] で提案された手法を，役割による学習手法 (Role learning 手法) と呼び，RL 手法と表記する．RL 手法は仮チームのメンバを選択するとき，エージェントのリソースを既知としており，リソースを完全に把握した状態でメンバを決定できる．RL 手法のエージェントは，第 4.2 節で述べた，欲張り度，提案受託期待度，報酬期待度の 3 つパラメータを学習する．この手法と本提案手法の比較により，エージェントのリソースが既知，あるいは未知とした環境でのタスク処理の効率を比較できる．

RL 手法の仮チームのメンバ候補も ε-*greedy* 選択を用いるが，$1-\varepsilon$ の確率では以下のように決定する．

1. リーダ i はサブタスクの報酬 (要求リソース) の大きい順にソートし，報酬の大きいサブタスク s から順に依頼する．$G_T^p = \emptyset$ と初期化する．
2. 提案受託期待度の一番大きく G_T^p に含まれないエージェントを j とし，s の要求リソースと j がもつリソースの各要素を比較する．式 (3.1) を満たす場合，j を s の処理依頼をするエージェントとして G_T^p に加える．満たさない場合，G_T^p に加えない．
3. i は j の次に提案受託期待度の大きいエージェント j_2 の調査に移り，同様に条件を調べる．

4. 2 および 3 の試行をチーム参加依頼提案数 L 回繰り返す．L 体発見できない場合は次のサブタスクへ移る．

ε の確率では提案手法同様，サブタスクの担当をランダム決定するが，このときリーダ i はサブタスクの処理が可能なエージェントの中からランダムに決定する．RL 手法の仮チームのメンバ決定のアルゴリズムを，Algorithm 2 に示す．

Algorithm 2 RL 手法における仮チームのメンバ選択方法

Require: T: タスク
K: エージェント集合
i: リーダ
L: 各サブタスク毎に選択するエージェント数 (チーム参加依頼提案数)
$S'_T \Leftarrow S_T \setminus \{s_i\}$ //i が処理するサブタスクを S_T から除く
$G^p_T \Leftarrow \emptyset$
S'_T の要素 s を要求リソースの和に基づいてソートする．
for all $s \in T$ **do**
  $l(s) \Leftarrow 0$
  $K \Leftarrow K \setminus \{i\}$
  for $j \in K : e_{i,\,j}$ を用いて選択する **do**
    if $h^k_j \geq r^k_s$ for $\forall k$ **then**
      $G^p_T \Leftarrow G^p_T \cup \{j\}$ //j を仮チームへ加える
      $K \Leftarrow K \setminus \{j\}$
      $l(s) \Leftarrow l(s) + 1$
      if $l(s) = L$ **then**
        break
      end if
    end if
  end for
  if $l(s) < L$ **then**
    break //次のサブタスクへ移る
  end if
end for

NoLearning 手法

エージェントが自分の役割やチームを組むべき相手を学習しない手法を NoLearning 手法と呼び，NL 手法と表記する．この手法は，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，エージェントのリソース情報は既知とし，その情報を元に実行可能なエージェントにチーム加入依頼メッセージを送る．本手法との比較により学習の効果を調査することができる．

表 4.1: エージェントの設定

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
チーム参加依頼提案数 L	2

CNP

リソース情報を既知としない比較手法としてコントラクトネットプロトコル (CNP) と比較する．CNP には本モデルでのリーダに相当するマネージャエージェントが存在するが，この手法でのマネージャ数は n に固定とした．そのため，本実験ではマネージャ数を手動で変更して実験を行い，CNPn として表記する．例えばマネージャ数 5 体の場合は CNP5 と表記する．今回，マネージャの数を 1, 2, 5, 10 の 4 種類で実験した．

CNP では第 2.3.1 節の図 2.3 でも示した通り，タスク処理のメッセージをシステムに存在するすべてのエージェントに広報するが，本手法ではマネージャもタスクを処理するため，マネージャが処理しないタスクのみとなる．また，CNP のタスク割当方法は以下のように決める．マネージャエージェント i はチームに参加すると返答したエージェントの集合 K の中から，s を処理できるエージェントに割当てる．複数のエージェントから入札 (チーム加入依頼メッセージに対する受託) があった場合，割当てるエージェントはその中からランダムに決定する．

4.4.2 実験設定

実験で用いた各種パラメータ設定を表 4.1，表 4.2，表 4.3 に示す．エージェント数は 50 体とし，各エージェントがもつリソース h_j^k は 3 から 12 の正の整数からランダムに決定する．また，エージェントがチーム編成時に送信するチーム参加依頼提案数 L は 2 とする．一つのタスクに含まれるサブタスクの数 $|S_T|$ は 3〜7 としランダムに決定し，各サブタスクがもつ要求リソース r_j^k は 1〜8 からランダムに決定する．またシステム負荷 λ はポアソン分布で 2 とし，単位時間毎に追加される．

表 4.2: 学習パラメータの設定

パラメータ	値
欲張り度の学習率 α_g	0.1
提案受託期待度の学習率 α_e	0.05
報酬期待度の学習率 α_d	0.05
欲張り度 g_i の初期値	0 〜 1 のランダム
提案受託期待度 $e_{i,\,j}$ の初期値	0.5
報酬期待度 $d_{i,\,j}$ の初期値	0.5
ε-$greedy$ 選択の ε	0.05

表 4.3: タスクの設定

パラメータ	値
1 tick の平均システム負荷 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
一つのタスクに含まれるサブタスクの数 $\lvert S_T \rvert$	3 〜 7 のランダム

4.4.3 実験結果と考察

実験結果

実験結果を図 4.1 に示す．この結果はそれぞれの手法の 50tick 毎のタスク処理数を集計したものである．また，50 回試行の平均値である．提案手法は 5000 tick までにタスクの処理数が急激に増加しているが，これはリソースの推定と各学習パラメータの学習が進み，チーム編成の成功率が向上したことを示している．また，29000 tick から 30000 tick において提案手法は，リソースを既知とした RL 手法を約 0.3 %上回る結果を得ることができた．

CNP 手法は CNP5 が最もスク処理数が多く，CNP1 とした場合は，TFR^2 手法，RL 手法，NL 手法を含む全ての手法の中でもっともタスク処理の効率が低い結果となった．

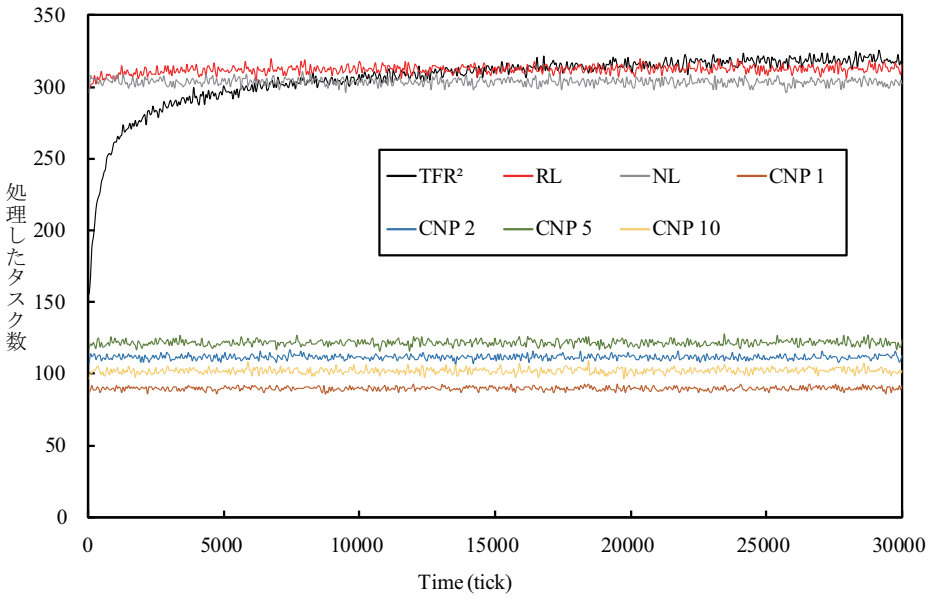

図 4.1: 各手法のタスク処理数の推移

考察

TFR^2 手法について

TFR^2 手法は，リソース推定パラメータの学習ために，ε-*greedy* 選択によって $\varepsilon(5\,\%)$ の確率でリソース推定パラメータを考慮せずランダムにエージェントを選択し提案する．そのため，既存手法に比べてチーム編成の成功率が低くなり，既存手法より 5 % ほどタスク処理数が低くなることが考えられた．しかし，最終的な結果は提案手法が数 % 上回る結果となった．

TFR^2 手法がエージェントのリソース情報を完全に把握できる既存手法を上回った理由として第一に，提案手法の ε-*greedy* 選択によるランダム選択があげられる．本研究におけるリソース推定パラメータの学習戦略は，リソースの推定学習が進んだ場合でも，リソース推定パラメータの学習のために ε の確率でランダムにエージェントを選択する．このとき，通常の確率 $(1-\varepsilon)$ では選ばれることのないエージェント (例えば提案受託期待度が低いエージェントおよびリソース推定パラメータが低いエージェント) を選択する可能性がある．TFR^2 手法はそのエージェントを新たに有効活用することでチーム編成効率を向上でき，結果的に既存手法を上回ったと考える．

第二に，リソース学習の過程で個々のエージェントが提案受託期待度を分散して学習したことが挙げられる．TFR^2 手法は提案受託期待度の学習が進んでいない初期段階では，

リソースの推定も完了していないためリソース情報を考慮せずランダムに依頼する．これによりリソース情報を考慮して依頼する RL 手法よりも依頼が分散される可能性が高くなる．その結果，有用なメンバが各チームに分散され，一つ一つのチームが RL 手法に比べて効率的に編成できる．その結果，タスク処理の効率が向上したと考えられる．

既存手法と NL 手法

既存手法と NL 手法を比較すると既存手法が上回るが，これは，欲張り度，報酬期待度，提案受託期待度の 3 つのパラメータ学習の成果を示している．本実験でも学習パラメータの有用性を示すことができた．

CNP について

CNP は，チーム加入依頼メッセージを全エージェントに送りチーム編成の失敗リスクを減らせるためタスク処理数が高くなると予想されたが，4 つの手法の中で最も効率が低い結果となった．この原因は CNP 手法の全てのエージェントを対象に提案をすることが考えられる．CNP においてタスクを処理するエージェントは，タスクを取得した全てのマネージャからチーム加入依頼メッセージを受け取るが，大きいリソースを持つエージェントは，より多くの報酬が獲得できるチームへの参加を試みる．これは，エージェントが合理的であるため，自分の報酬を最大化しようと行動するためである．その結果，獲得報酬が大きいチーム加入依頼メッセージ (本研究のモデルでは，サブタスクの要求リソースの和が獲得報酬となっているため，要求リソースが大きいサブタスク) に入札が集中する．しかし，サブタスクが割当てられるエージェントは 1 体だけであるため，タスク処理ができないエージェントが多数発生する．そのため，ある程度能力のあるエージェントである関わらずタスクを処理できず，さらに報酬が小さいサブタスクには入札が入らないため，処理されないサブタスクが発生し，図 4.1 の結果になったと考える．また，CNP 手法の前提として，能力に明確に差がある場合を想定しており，本研究のようにエージェントのもつリソースに差があまりない環境では，入札が集中して十分な効率を得られないことも要因として考えられる．

一方で本実験では，マネージャ数を 5 とした CNP5 が CNP 手法の中で最も効率をあげている．入札の集中はマネージャ数の増加とともに起こりうるが，CNP10 や CNP5 に比べて CNP1 や CNP2 は成果をあげられていない．これは，CNP におけるマネージャ数は，チーム数と同数となるため，タスクを効率的に処理するには作業を並列化 (チーム数を増

加) しなければならないが，増加しただけメッセージ数の競合 (同じサブタスクに入札が集中すること) が発生する．そのため，CNP 手法におけるタスク割当の効率性は，依頼の集中と作業の並列性のトレードオフであることを示した．この点で TFR^2 手法は役割を学習しているため適切なリーダ (マネージャ) の数になると考えられる．さらに TFR^2 手法，RL 手法，NL 手法は，チーム参加依頼提案数を 2 としているが，少ないメッセージ数でも効果的に学習およびチーム編成をすることができることを示している．

4.5 本章のまとめ

本章では第 3 章で提案したモデルにおいて，エージェントに学習とリソース推定手法を導入した，実験から，リソースが想定できない未知な分散環境でも効率的にチームを編成し，効果的にタスクを処理できることを示した．また，結果からリソースが把握できる既知環境に比べてわずかではあるが効率を上げることができ，リソースの推定によって分散的にチームを編成できる可能性を示唆した．これは直観に反する結果であるが，これについて考察を加えた．また，集中的に制御する CNP と比較し，本手法は効率的にタスクを処理できたため，分散的に制御する有用性も合わせて示すことができた．

第5章　大規模環境を考慮したスコープの導入とその評価

本章では，多数エージェントが存在する大規模環境に焦点を当てる．前章の知見に基づき，エージェントの学習とリソース推定手法に加えて，エージェントにスコープを導入する．ここではチームを編成するために依頼対象となるエージェントを全体の一部に制限した手法をスコープと呼び，多数エージェントが存在しても個々のエージェントが持つ情報量を減らし効率的なタスク処理を実現した．ここでの手法は [67] で議論した内容に基づいている．

5.1　はじめに

さまざまなモノがネットワークにつながりはじめ，その数は爆発的に増加している．そのため，先にも述べたが集中的な制御によるシステム管理では限界に達しており，分散的な制御によるボトムアップなアプローチが必要となっている．一方で，既存研究では各エージェントがそれぞれ，全てのエージェント情報をもち，それらを参照することでチーム編成時にタスクに応じたエージェントを選択できた．しかし，一般には他のエージェントの情報を仮定することは難しい．この課題に対し，前章でエージェントの能力の一部を学習しながらタスクを割当てる手法を提案したが，この手法ではエージェント数の増加にともない，保持する情報量が増加する．仮に全エージェント情報を保持できる保存領域を確保できても，要求メモリや計算量が膨大になり，同時に学習の効率も低下する．さらに，全体のエージェントとの通信量も膨大になることも考えられる．そこで，エージェント数が増加しても使用メモリや計算量を抑えつつ，効率的なチーム編成を実現するために，本手法では，通信可能なエージェントを一部に制限したスコープを導入し，さらにそのスコープのエージェント要素を入れ替える学習機能を導入する．また，本章で用いるエージェントモデルは，第 3 章で述べたモデルと第 4 章で用いた手法を用い，その追加機能としてスコープを提案した．図 5.1 にこれまでの章と本章の関係性を示す．また，ここでは，スコープの有効性をエージェント数とシステム負荷を変えた実験で評価した．

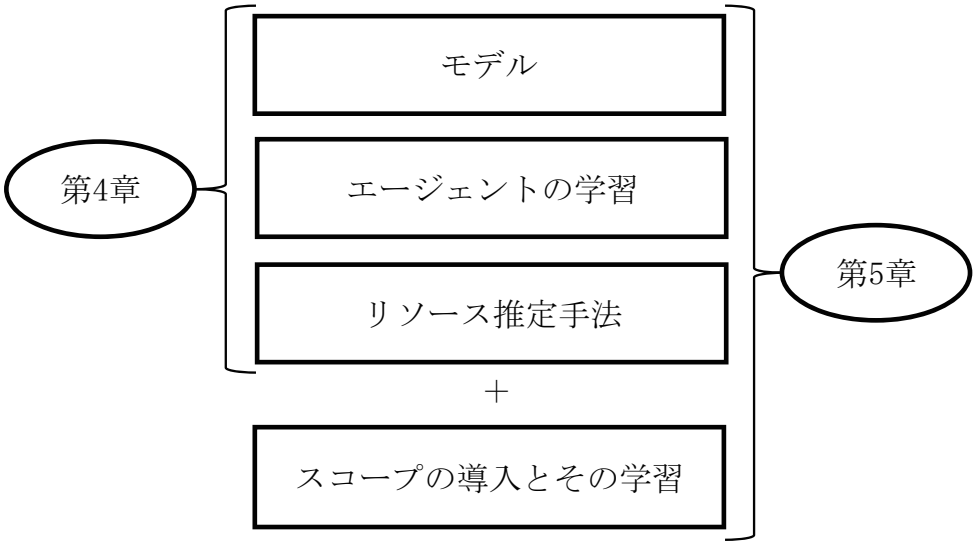

図 5.1: これまでの章と本章の関係性

5.2 スコープ

各エージェントがチームを組むべき対象を一部に制限し，それをスコープと呼ぶ．エージェントはスコープのエージェントをサブタスク処理を依頼する対象とするため，スコープはメッセージを送信する範囲を制限する．

エージェント i のスコープとなるエージェント集合を F_i とし，その要素数を M とする．M は予め決められた正定数であり，スコープサイズと呼ぶ．F_i のエージェントはシステム開始時に，$A \setminus \{i\}$ からランダムに M 体選ぶこととする．i はこの F_i のエージェントに関するリソース推定パラメータと提案受託期待度および報酬期待度を持ち，F_i のエージェントをサブタスク処理を依頼する対象とする．ここで，i がスコープ F_i にエージェント j を選んでいたとしても，j がスコープ F_j に i を入れているとは限らず，エージェントは相互にスコープ要素とは限らない．従来との通信範囲の比較図を図 5.2 に示す．図からも明らかなように，全体のエージェントと通信する場合に比べ，少ないことがわかる．他方，この制限により，十分な数のエージェントに依頼が渡らず効率が下がる可能性もある．

5.2.1 スコープの学習

学習パラメータやリソースの推定学習が進むにつれて提案受託期待度やリソースの低いエージェント，つまりリーダが依頼したときに断る可能性の高いエージェントがスコープ

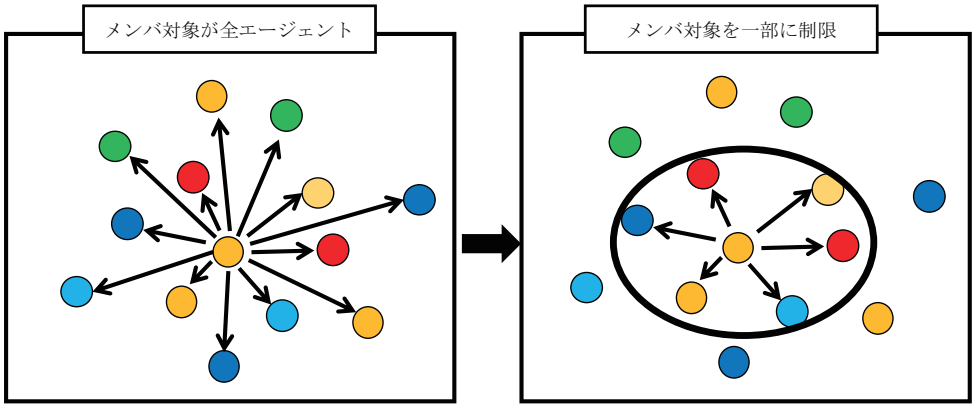

図 5.2: スコープの概念図

内に発生する場合が考えられる．そのため，本手法ではスコープの要素のエージェントを実績に応じて入れ替える．これにより有用なエージェントをスコープに追加し，逆に不要なエージェントは消去することで，スコープのエージェントでチーム編成の成功率が向上するようにした．(スコープの質を上げるとも言える．) スコープの要素の変更法を以下に示す．毎 tick，β $(0 \leq \beta \leq 1)$ の確率でエージェント i はスコープの要素のエージェントを入れ替える．このとき

1. 提案受託期待度が最小のエージェント $j \in F_i$ を選択する．
2. i は j に対する，リソース推定パラメータ $\tilde{H}_j$ および提案受託期待度 $e_{i,j}$ を消去する．
3. i は，F_i から j を削除する．
4. i は，全体のエージェント $A \setminus \{i\}$ からエージェントをランダムに 1 体選び F_i に加える．

として i は，新しいスコープを依頼する対象とし，チーム編成および学習をする．

5.3 評価実験：スコープの効率調査実験

5.3.1 評価実験の概要と比較手法

ここではスコープがタスク処理効率に与える影響を調べるために，エージェント数やシステム負荷を変化させた実験とスコープの入れ替え確率を変化させて実験する．実験は，

表 5.1: 本章での実験について

該当実験	エージェント数	λ	入れ替え確率
実験 1	少数 ($\|A\|$=50)	小さい ($\lambda = 2$)	一定
実験 2	多数 ($\|A\|$=500)	小さい ($\lambda = 2$)	一定
実験 3	多数 ($\|A\|$=500)	大きい ($\lambda = 15$)	一定
実験 4	多数 ($\|A\|$=500)	大きい ($\lambda = 15$)	変更

表 5.2: 実験 1 におけるエージェント

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
チーム参加依頼提案数 L	2

期間内にエージェントが処理に成功したタスクの数で評価する．実験の詳細を表 5.1 に示す．本実験ではスコープサイズ M を変更させ，比較する．ここで $M = |A|$ とした場合は，全エージェントをスコープとするため，第 4 章で提案した TFR^2 手法と同様である．

5.3.2 実験 1:小規模環境におけるスコープの効率調査

実験 1 では第 4 章と同様にエージェントは少なく，さらにシステム負荷も小さい小規模な環境におけるスコープによる影響を調査する．

実験 1 の設定

エージェントの数は 50 体，1 tick 毎にシステム負荷 λ をポアソン分布で 2 とした．本実験では，スコープサイズ M を $15, 20, 30$ および全体をスコープとする $|A|$ とし，4 手法で比較した．また，スコープとするエージェントの入れ替え率は $\beta = 0.0001$ とした．実験で使用した各パラメータを表 5.2〜表 5.5 に示す．

表 5.3: 実験 1 におけるタスク

パラメータ	値
1 tick の平均システム負荷 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
一つのタスクに含まれるサブタスクの数 $\lvert S_T \rvert$	3 〜 7 のランダム

表 5.4: 学習パラメータの設定

パラメータ	値
欲張り度の学習率 α_g	0.1
提案受託期待度の学習率 α_e	0.05
報酬期待度の学習率 α_d	0.05
欲張り度 g_i の初期値	0 〜 1 のランダム
提案受託期待度 $e_{i,\,j}$ の初期値	0.5
報酬期待度 $d_{i,\,j}$ の初期値	0.5
ε-$greedy$ 選択の ε	0.05

表 5.5: スコープの設定

パラメータ	値
スコープサイズ M	15，20，30 および $\lvert A \rvert$
スコープ入れ替え率 β	0.0001

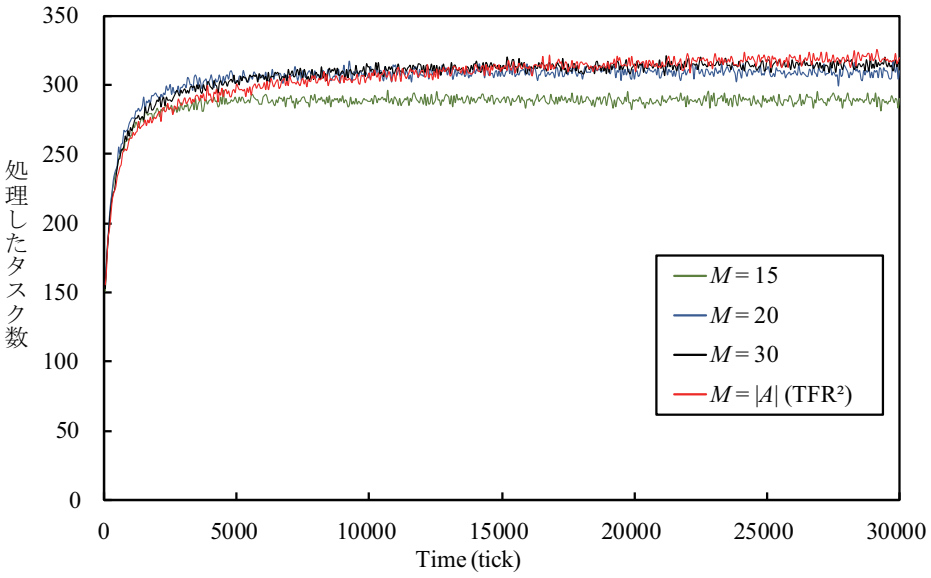

図 5.3: 小規模環境におけるスコープの評価

実験 1 の結果

実験 1 の実験結果を図 5.3 に示す．結果は 50 tick 毎の各手法のタスク処理数を測定し，30000 tick までプロットした．また，実験結果は 50 回試行の平均値である．この結果から，スコープを導入した手法 (M=15,20,30) は M=$|A|$ に比べ，10000 tick までの収束が速く効率的にタスクが処理されることがわかる．一方で，最終的な結果は，M=$|A|$ がわずかに効率が上回る結果となった．

実験 1 の考察

ここでは，収束の速度と最終的な効率の差について議論する．まず収束が速くなった点は，学習対象エージェントの差があげられる．スコープを導入した手法 (M=15,20,30) は，スコープに含まれるエージェントのみを学習すればよいため，全体のエージェントを学習する M=$|A|$ よりもリソースの推定および各学習パラメータの学習を早く進めることができる．その結果，収束を速め，早い段階で効率を向上させることができている．

一方で，最終的な効率の差は，全体のエージェントを学習している M=$|A|$ の方がよい．これは，チームを組むエージェントの質の違いが考えられる．全体エージェントを参照できる M=$|A|$ は，全エージェントから最適なエージェントを学習することができるため，時間はかかるものの，学習を終えた場合は最適なチームを編成でき，しかもその候補はス

コープによって制限した場合に比べて多い．その結果，タスク処理の効率に差が現れたと考える．しかも，この実験では，エージェント数が少ないため，学習が最終的にスコープを導入した手法に追いつき，結果的に効率も上回る結果となった．

これらのことから，スコープを導入した手法は効率の面ではわずかに劣るが，保持する情報量を少なくできる点では評価できる．

5.3.3 実験 2:システム負荷が小さい環境での効率調査

実験 2 では，実験 1 の環境からエージェント数を 10 倍にした 500 体として実験した．また，比較手法はエージェント数の増加に伴い，スコープサイズを変更し，$M = 15,30,50,100$ と $|A|$ の 5 手法で比較し，実験 1 と同様，30000 tick 間の処理したタスク数で評価する．エージェント数およびスコープサイズ以外の条件は，実験 1 の表 5.2〜表 5.5 と同じである．この環境は，エージェント数は多いが要求タスクは少なく，負荷が小さい環境を想定している．

実験 2 の結果

実験 2 の実験結果を図 5.4 に示す．この結果も 50 tick 毎の各手法のタスク処理数を測定し，30000 tick までプロットした．また，実験結果は 50 回試行の平均値である．結果から，スコープサイズに関わらず収束速度および処理したタスク処理数にあまり差が現れず，全ての手法が 5000 tick までにタスク処理数が収束している．

実験 2 の考察

収束速度と効率に差が現れなかった理由として，システム負荷が小さいことが考えられる．ここでの実験は，システム負荷 (タスクの数) に比べ，エージェント数が非常に大きい環境であるため，チーム編を成するエージェントは少なくなる．そのため，依頼した相手が別の依頼を受けている可能性が低くリーダの提案が比較的通りやすいため，チーム編成の成功率が高い環境である．さらに，チーム編成成功率が高い理由として，メンバの役割を選択するエージェントが多くなることもあげられる．これは初期状態のエージェントの役割選択に以下のことが起こるためである．役割は第 4.2.4 節で述べた期待報酬の比較で選択するが，このときリーダとしての期待報酬を表す式 (4.7) はキューにタスクがない

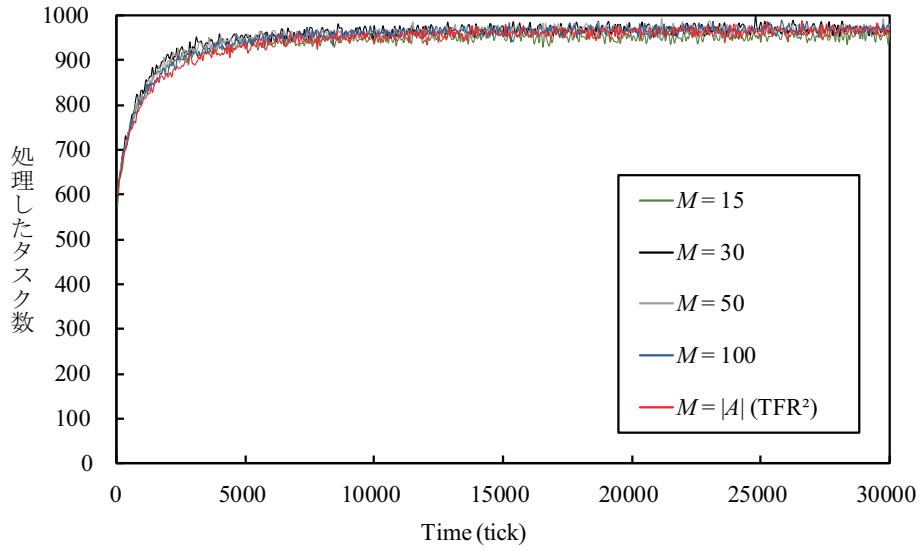

図 5.4: システム負荷が小さい大規模環境におけるスコープサイズ毎のタスク処理数の推移

ため 0 となる．一方でメンバとしての期待報酬を表す式 (4.8) は，提案を受け取った場合は少なくとも 0 ではなくなる．このため，初期状態のエージェントに提案した場合，エージェントはメンバとしての役割を選択することが多くなると考えられる．その結果，リーダに比べてメンバの数が増え，チーム加入依頼メッセージを送信した場合，リーダの役割を選択しているという理由で拒否される可能性が低くなる．さらにタスク数が少ないのでメンバはチーム加入依頼メッセージを重複して受け取る可能性も低く，チーム編成の成効率が向上する．これらの理由により，どの手法も同程度の効率が得られたと考えられる．ただし差は現れないが，実験 1 と同様に，保持できるエージェント情報をスコープとして制限しているにも関わらず，スコープサイズを $M{=}|A|$ とした場合と同程度の効用を得ており，提案手法は計算コストや要求メモリを押さえられた点では勝っている．

5.3.4 実験 3:システム負荷の大きい環境での効率調査

ここでは，エージェント数は 500 体で実験 2 と同数であるが，システム負荷を 15 と増やし，エージェント数，負荷がともに多い大規模な環境においてスコープの有効性を調査する実験を行なった．エージェント数，システム負荷，およびスコープサイズ以外の実験条件は実験 1 の表 5.2〜表 5.5 と同じである．

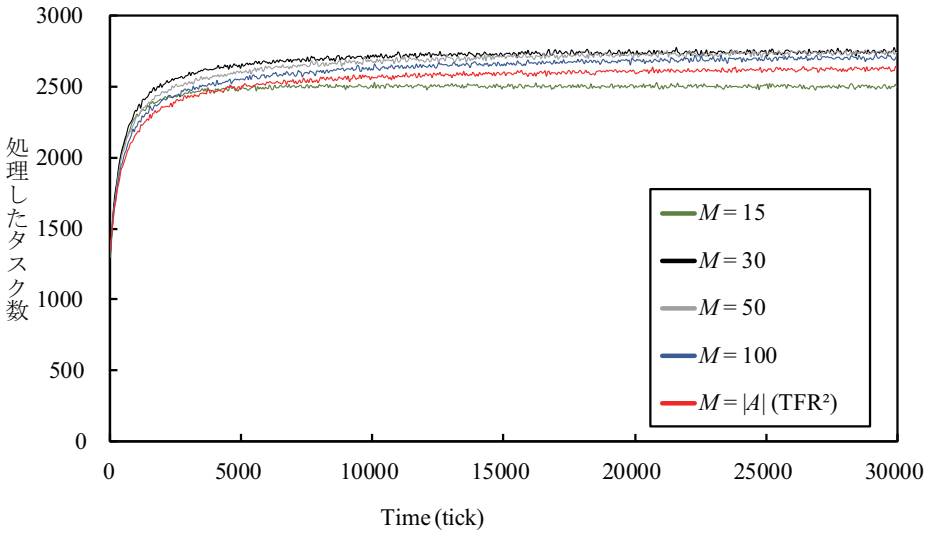

図 5.5: システム負荷が大きい大規模環境におけるスコープサイズ毎のタスク処理数の推移

実験 3 の結果

実験 3 の実験結果を図 5.5 に示す．結果は 50 tick 毎の各手法のタスク処理数を測定し，30000 tick までプロットした．また，実験結果は 50 回試行の平均値である．結果から，全ての手法で 5000 tick まで急激に処理したタスク数が増加しているが，これはエージェントがリソース推定パラメータおよび役割の学習によって，チーム編成効率が向上したためである．特に，$M = 30, 50, 100$ は $M = |A|$ とした場合よりも収束は速く，処理したタスク数も多くなった．一方で $M = 15$ とした場合は，完全にタスク処理数が収束はしているもののその他の手法に比べてその数は小さくなった．

実験 3 の考察

$M = 100$ や $|A|$ に比べて $M = 30, 50$ の収束が速くなったのは 2 つの理由が考えられる．まず第一に学習対象エージェント数の違いである．スコープサイズが大きい (今回は $M = 100$ や $|A|$) 場合，リーダは，メンバ候補を多く保持できるが，それと同時に学習対象となるエージェントも多くなる．そのため，有用なエージェントの学習 (発見) に時間がかかり収束に時間がかかった．実際に，$M = |A|$ とした場合は実験 3 の実験時間内で収束を示すことができていない．それに比べて $M = 30, 50$ とした場合は，学習対象となるエージェントが少なく，早い段階から有用なエージェントを発見できることで収束を速めることができている．ただし，$M = 15$ のようにスコープサイズが極端に小さい場合は

収束は速くなるが効率は悪くなる．このことから学習対象が多すぎると収束速度も遅くなるが，逆に学習対象が少なすぎるとチーム編成の効率自体に影響し，スコープサイズと処理タスク数にはトレードオフの関係があることがわかる．また実験 1 でも学習対象のエージェント数の違いを議論したが，今回の実験ではエージェント数が多いため，全体のエージェントから学習するより一部に制限して学習したほうが，結果的に収束と効率が向上することが明らかになった．本研究では，$M = 30$ が適切なスコープサイズとなったが，エージェントの規模に対する適切なスコープサイズはエージェント数やシステム負荷によって変化すると考える．

第二の理由として，チーム加入依頼メッセージの重複があげられる．今回の実験環境では，エージェント数に加えてシステム負荷も大きいためチーム編成の機会も増える．チーム編成が頻繁に行われることで学習の機会も多くなるが，リーダの有用と判断するエージェントが重複するため，チーム編成成功率が低くなると考えられる．これについては次章で説明する．特に，このチーム加入依頼メッセージの重複による効率低下は，学習対象のエージェントが多くなるほど発生すると考えられる．そのため，全エージェントの提案受託期待度やリソース推定パラメータを調べて最適なメンバに提案できる $M = |A|$ はその他の手法に比べて処理したタスク数が少ない．さらに，チームのエージェント候補が多くなるとチーム編成に必要なエージェントの発見にも時間がかかり，収束速度やチーム編成成功率も低下すると考えられる．今回効率を上げられた $M = 30, 50$ は，スコープによって提案するメンバを押さえることができたため，自然に依頼・被依頼の関係に基づくクラスタを構成し，結果として高いチーム編成効率 (収束速度の向上) を引き出せたと考えられる．

本実験から，大規模なエージェント環境ではスコープの導入により効率的なタスク処理を実現することを示した．

5.3.5 実験 4:スコープのエージェント入れ替え率変更実験

実験 4 ではスコープの学習率を変更し，スコープに含まれるエージェントの入れ替え頻度の変更がチーム編成に与える影響を調べる．本実験では，スコープ入れ替え率 β を 1/100(0.01) と 1/1000(0.001) とおいて実験する．実験 4 の実験環境はスコープの入れ替え確率以外，実験 3 と同様である．

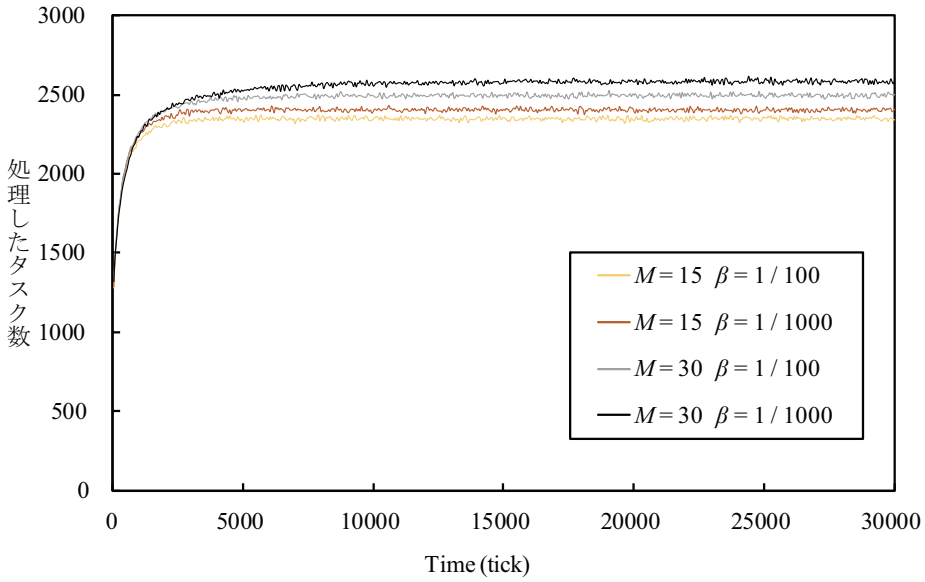

図 5.6: スコープ入れ替え率を変更した場合のタスク処理数の推移

実験 4 の結果

実験結果を図 5.6 に示す．結果は 50 tick 毎の各手法のタスク処理数を測定し，30000 tick までプロットした．また，実験結果は 50 回試行の平均値である．$M = 15$ の場合も $M = 30$ の場合も β を小さくした方が大きいものに比べ収束は速く，最終的な処理タスク数も多くなった．特に，$M = 30$, $\beta = 1/1000$ とした場合はその他の手法に比べ，収束速度やタスク処理数が最も多くなった．

実験 4 の考察

入れ替えを導入した経緯は，リーダにとって提案を受託しにくいエージェントの情報は必要でないため，その情報を破棄しスコープの精度を保つためである．しかし，実験により入れ替えの頻度が多くなると効率自体を下げる結果となった．これはエージェントの入れ替えによって再度学習しなければならないエージェントが多くなったことが考えられる．入れ替え率 $\beta = 1/100$ とした場合，$1/1000$ に比べてスコープのエージェントを頻繁に入れ替えるため，新しくスコープとしたエージェントの提案受託期待度とリソース推定パラメータを学習する必要がある．一方で，有用と判断するエージェントを学習したときには，さらに入れ替える可能性もあるため，処理タスク数は頭うちとなる．本実験では入れ替え頻度は小さい方が効率がよかったが，適切な入れ替え頻度の確率は実験 3 同様，

エージェント数に依存することが予想される．

5.4 本章のまとめ

本章では，第 3 章の提案モデルと第 4 章で提案したリソース推定手法を拡張し，大規模な環境に対応するためのスコープを導入してエージェントの学習範囲とチーム編成対象を絞込み，その評価と考察をした．実験では依頼対象を一部に制限したスコープ手法を，全体を依頼対象とする従来手法と比較し，スコープを導入した手法は，小規模な環境では従来手法と同程度，大規模な環境では効率を向上させる結果を得られた．また，負荷の増加によるメッセージの重複をスコープで抑えることができた．また，各エージェントが保持する情報量を抑えており，メモリ使用量とネットワーク利用効率の観点からも評価できる結果を示した．

第6章　信頼性に基づくエージェント共同関係の促進とその評価

本章では，第5章と同様にエージェント数，システム負荷が増加した大規模分散環境での効率的なタスク処理の実現をめざし，そのための手法を提案する．ここでは第3章で議論したモデルを利用するが，第4章および第5章のエージェントの学習，リソース推定法やスコープは使用せず，新たなエージェントの学習とエージェントの行動戦略を導入した．特にこの行動戦略のために，信頼できるエージェントの有無によって自分の戦略を変化させる戦略選択エージェントを提案した．この手法によりエージェントが相互に信頼関係を構築し，安定したチームを編成できることを示した．これらの結果は [68, 69, 70, 71] にて発表した．

6.1　はじめに

第5章で提案したスコープは，大規模環境に対して効率を落とさずチーム編成が可能であったが，エージェントが個々にもつスコープサイズを手動で決定する点や，その精度を保つためにスコープのエージェントを確率的に入れ替える点がボトルネックであった．実環境では，エージェント数はシステム設計時に予測できず，さらに入れ替えや追加などでその数は一定でないことが考えらる．そのため，制御にトップダウンな仕組みを導入せず，大規模な環境に動的に対応できる手法の提案が必要である．

一方で，スコープによりエージェント学習を制限することで大規模環境でも学習の収束を速められた．このことから，エージェントの相互関係を促進することが重要であると考えた．そこで，本手法ではエージェントの協調関係をボトムアップに形成することで大規模な環境における効率的なタスク処理の実現を目指す．特に，生物学 [72] などですでに議論されている直接互恵性の概念をエージェントへの行動へ適用した．この概念はハムラビ法典等の古代の文化にもある「目には目を，歯には歯を」を応用している [73]．ここでの互恵性とは，自分が相手に対して何らかの行動を起こした結果，相手にとっては利益とな

るが，自分への見返り (利益) が以下のようになる場合が当てはまる．

1. 見返りとなる利益が発生しない．
2. 見返りを得るための代償として自分は損害を被る．
3. 見返りを得るために時間がかかる．

特にこれらの行動は親子間で観測され (子育て)，生物においてしばしば見られる行動である [72]．また，この理論に基づいた行動はゲーム理論 [62, 74, 75] で議論されており，具体的な戦略例としてしっぺ返し戦略が有名である．このしっぺ返し戦略により囚人のジレンマゲームと同様な相互関係がある場合においても安定したエージェントの協調関係を構築・維持することが可能である．

本研究ではこれらの概念を応用し，エージェントの行動戦略として新たに互恵戦略を導入し，以下のように定義した．エージェントは自分に対して協調的な行動を取るエージェントに対してはより協調行動 (ここでは依頼をする，受託する) を，非協調的な行動をとる相手には非協調行動 (ここでは依頼をしない，拒否する) をとる．さらに本手法では，学習によって上記の互恵戦略と従来の合理戦略を状況に応じて選択する戦略選択エージェントを提案した．またここでは戦略選択エージェントに加え，これまで議論した学習を改良し，新しい学習パラメータを導入し，提案手法の有効性を調査した．

本章で用いるエージェントモデルは，第 3 章で述べたモデルを用いるが，第 4 章および第 5 章で議論したエージェント学習，リソース推定手法，およびスコープは用いない．また，提案したエージェントの学習と戦略選択エージェントによってタスク割当とメンバ選択方法を一部変えている．この点については提案手法とともに述べる．図 6.1 にこれまでの章と本章の関係性について示す．

6.2 エージェントの学習の改良

ここでは本章で導入した学習パラメータについて述べる．第 4 章と第 5 章で提案したエージェントの学習における欲張り度，提案受託期待度，報酬期待度の 3 つのパラメータを結合・改良し，新たなパラメータとして，協調期待度，リーダの役割適正値およびメンバの役割適正値をエージェントに導入した．以下にその詳細を述べる．

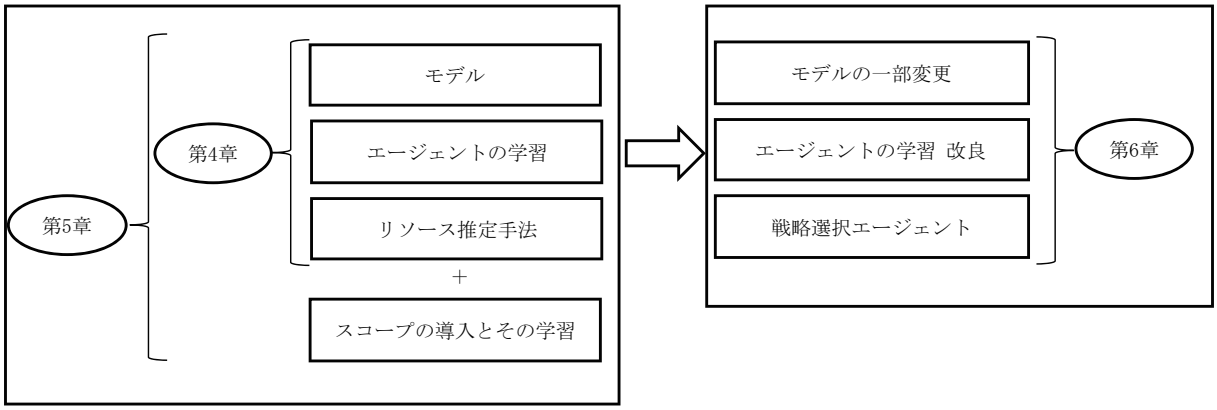

図 6.1: これまでの章と本章の関係性

6.2.1 協調期待度と協調相手の選択

協調期待度は，リーダ・メンバの役割にかかわらず，自分の行動に対する他のエージェントの協調可能性を表す．このパラメータは提案受託期待度と報酬期待度で役割毎に学習していた相手の協調可能性を一つに統合したものとも位置づけられ，この値が大きいものが信頼度が高いと判断し，それに基づいた行動をする．そのためエージェントは協調期待度を用いて以下のように協調相手を選択する．自分がリーダの場合にサブタスクの依頼をする際，協調期待度の大きいエージェントは依頼の受託率が高いと期待できるため (提案受託期待度と同様)，優先的にそのエージェントに依頼する．自分がメンバの場合，サブタスクの依頼を受けた際，協調期待度の大きいエージェントからは実際にタスクが割当てられる (つまり報酬獲得の可能性が高い) と期待できるため (報酬期待度と同様)，優先的にそのエージェントからの依頼を選択する．そのため合理的判断として，協調期待度の大きいエージェントを協調相手として選択する．ある t tick 目におけるエージェント i の j に対する協調期待度 $e_{i\to j,t}$ は，チーム編成結果に基づき，以下の式で更新する．

$$e_{i\to j,t} = \alpha \times \delta_{j,t}^{act} + (1-\alpha) \times e_{i\to j,t-1} \tag{6.1}$$

ここで α は学習率であり，$0 < \alpha < 1$ の値をとる．$\delta_{j,t}^{act}$ は自分が選択している役割に応じて以下のように決定する．i がリーダであれば，サブタスクを依頼したメンバ j がその依頼を受託したとき $\delta_{j,t}^{act}$=1，受託しなければ 0 とする．i がメンバの場合には参加表明したチームのリーダ j に対し，チーム編成が成功し実際に j からサブタスクが割当てられれば $\delta_{j,t}^{act}$=1，その他の場合は 0 とする．この協調期待度に基づくメンバ候補決定方法は後述する．

ここで，学習した協調期待度が古くなる場合がある．たとえば，あるエージェント i と j が一時的な連携によって，協調期待度を学習し，お互いが協調的であると判断したとする．その後 i も学習が進み，j 以外と協調するようになった場合 (逆に j も i 以外と協調することもありうる)，学習の機会が減り，その値を更新しにくくなる．このように一時的な連携だったにも関わらず，協調期待度が更新されず古い情報で維持され続けることを防ぐため，定期的に協調期待度を僅かずつ減少させる．エージェント i の j に対する協調期待度の減少を以下の式で表す．

$$e_{i\to j,t} = \max\{e_{i\to j,t-1} - \alpha_F, 0\} \tag{6.2}$$

ここで $0 \leq \alpha_F (<< 1)$ は定数である．この減少は全エージェントが毎 tick 行う．この減少は，蟻コロニー最適化手法 [76] などでフェロモンが蒸発するのと同じ効果があり，古い情報を忘却させ，学習の最新性を保つ．

6.2.2 協調期待度によるメンバ候補決定方法

ここでは協調期待度を用いたメンバ候補選択方法を述べる．メンバ候補選択アルゴリズムを Algorithm 3 に示す．

まず，リーダの役割を選択したエージェント i は第 4 章，第 5 章と同様にキューからタスク T を選択し，自分が処理できるサブタスク s_i を自分に割当て，$T' = S_T \setminus \{s_i\}$ とする．その後，T' 中の残りのサブタスクの処理を依頼する仮チームのメンバを ε-$greedy$ 選択を用いて決定する．ここでは前述の通り，一つのサブタスク s につき，チーム参加依頼提案数 L 体分メンバを決定する．ε の確率でリーダ i はサブタスク s の処理依頼をするエージェント L 体を s が処理可能なエージェントからランダムに決定する．シミュレーション開始時も各学習パラメータが初期値であるため，ランダムにメンバを決定する．また $1-\varepsilon$ の確率で，リーダ i はサブタスク s の処理依頼をするエージェントを，相手のリソースと協調期待度を用いて次のように決定する．

1. リーダ i は T' のサブタスクを要求リソースの大きい順にソートし，その総和が一番大きいサブタスク s から順に依頼する．$G_T^p = \emptyset$ と初期化する．
2. 協調期待度の一番大きく G_T^p に含まれないエージェントを j とし，s の要求リソースと j のリソースの各要素を比較する．式 (3.1) を満たす場合，j を s の処理依頼をするエージェントとして G_T^p に加える．満たさない場合，G_T^p に加えない．

Algorithm 3 協調期待度を用いた仮チームのメンバ選択方法

Require: T: タスク
K: エージェント集合
i: リーダ
L: 各サブタスク毎に選択するエージェント数 (チーム参加依頼提案数)
$S'_T \Leftarrow S_T \setminus \{s_i\}$ //i が処理するサブタスクを S_T から除く
$G^p_T \Leftarrow \emptyset$
S'_T の要素 s を要求リソースの和に基づいてソートする.
for all $s \in T$ **do**
  $l(s) \Leftarrow 0$
  $K \Leftarrow K \setminus \{i\}$
  for $j \in K$:$e_{i \to j,t}$ を用いて選択する **do**
    if $h^k_j \geq r^k_s$ for $\forall k$ **then**
      $G^p_T \Leftarrow G^p_T \cup \{j\}$ //j を仮チームへ加える
      $K \Leftarrow K \setminus \{j\}$
      $l(s) \Leftarrow l(s) + 1$
      if $l(s) = L$ **then**
        break
      end if
    end if
  end for
  if $l(s) < L$ **then**
    break //次のサブタスクへ移る
  end if
end for

3. i は j の次に協調期待度の大きいエージェント j_2 の調査に移り，同様に条件を調べる．

4. 2 および 3 の試行をチーム参加依頼提案数 L 回繰り返す．送り先となる L 体のエージェントを発見できない場合，次のサブタスクへ移る．

サブタスク s の担当エージェントを L 体決定した後，次のサブタスク s_2 の調査に移り上記を繰り返す．この行動は，i が選択したタスク T に含まれるサブタスクの数だけ繰り返す．このようにして選ばれたメンバとリーダの集合が仮チーム G_T^p となり，リーダは G_T^p のメンバに対して，割当てるサブタスク情報を含んだチーム加入依頼メッセージを送り応答を待つ．その後，チーム参加を表明したエージェントとリーダの集合が G_T^0 となる．リーダは G_T^0 のメンバに割当関数 σ_T を元にサブタスクを割当てる．これについては後述する．

6.2.3 割当関数 σ_T の決定方法

G_T^0 の各メンバに対して，リーダ i は以下のようにサブタスクを割当てる．i は一つのサブタスクにつき L 体のエージェントに対してチーム加入依頼メッセージを送信するため，重複して受託される場合がある．この場合は，ε-$greedy$ 選択を用いて協調期待度の大きい方のエージェントにサブタスクを割当てる．また，全てのサブタスクを割当てた後，処理されないサブタスクが残っていた場合，G_T^0 内の実行可能かつ協調期待度の高いメンバに対して順に割当てる．このとき (G_T, σ_T, T) が式 (3.3) をみたせばチーム編成は成功となる．このようにして，実際にサブタスクを割当てたエージェントと，リーダの集合が G_T となる．

6.2.4 役割の学習と役割選択

エージェント i はチーム編成の成功率（したがって報酬獲得の可能性が）の高い役割を選択するために，役割適正値を学習をする．その学習のために，ある t tick 時のリーダあるいはメンバとしての報酬獲得可能性を表すリーダ適正値 $E_{i,t}^{ld}$ とメンバ適正値 $E_{i,t}^{mb}$ を導入する．このパラメータは i がリーダならば $E_{i,t}^{ld}$ を，メンバならば $E_{i,t}^{mb}$ を，それぞれ以下の式で更新する．

$$E_{i,t+1}^{ld} = \alpha \times \delta_{i,t}^{lsuccess} + (1-\alpha) \times E_{i,t}^{ld} \tag{6.3}$$

$$E_{i,t+1}^{mb} = \alpha \times \delta_{i,t}^{msuccess} + (1-\alpha) \times E_{i,t}^{mb} \tag{6.4}$$

ここで$0 < \alpha < 1$は役割適正値の学習率である．

$\delta_{i,t}^{lsuccess}$は，iがリーダのときにチーム編成が成功した場合$\delta_{i,t}^{lsuccess} = 1$，その他は0とする．$\delta_{i,t}^{msuccess}$は，$i$がメンバのときにチーム編成が成功し，そのとき割当てられたサブタスクが処理可能であったとき$\delta_{i,t}^{msuccess} = 1$，その他は0とする．

また，役割選択は以下のように行う．エージェントiは役割選択時に，$E_{i,t}^{ld}$と$E_{i,t}^{mb}$を比較し，その値の大きい方を自分の役割と判断して選択する．ただし，ε-$greedy$選択を用いて，εの確率で役割をランダムに選択する．もしリーダを選択したがタスクがタスクキューQにない場合，あるいはメンバを選択したが依頼のメッセージが無い場合は，処理すべきタスクは無いと判断し，次の時刻に再度役割選択に戻る．

6.3 戦略選択エージェントの提案

ここでは，エージェントが互恵戦略をとるべき対象 (これを信頼エージェントと呼ぶ) の定義とその決定方法について述べる．次に互恵戦略と合理戦略のそれぞれの行動を示し，最後にその戦略の選択方法を述べる

6.3.1 信頼エージェントと選択方法

信頼エージェントとは，エージェントが自分にとって協調的と判断し，互恵行動すべき相手を表すものとする．この判断のために本手法では，信頼エージェントの集合と信頼エージェント判定基準を導入する．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．本手法では信頼エージェント数の上限は設けない．また，信頼エージェント判定基準は，対象のエージェントが信頼エージェントかを判定する場合に用いる協調期待度の閾値であり，E^Jと表す．iがリーダjに対する協調期待度を式 (6.1) で更新したとき，信頼エージェント判定基準に基づき，$e_{i\to j,t} > E^J$ならばjを信頼エージェントと判断し，F^iに追加する．

リーダのチーム加入依頼メッセージが受託されない，あるいは式 (6.2) による協調期待度の減少により，F^iのエージェントでもE^J未満になるものが現れる．そのため，新たな信頼エージェントの判定時にF^iのエージェントの協調期待度を調べ，$e_{i\to k,t} < E^J$となるエージェント$k \in F^i$があれば，それをF^iから除く．

6.3.2 互恵戦略と合理戦略

合理戦略と互恵戦略はリーダとメンバの役割によってそれぞれ異なる行動を取る．特に戦略の違いが現れるのは，リーダであればメンバにサブタスクの依頼をするとき，メンバであればリーダから依頼を受けたときである．以下に合理戦略と互恵戦略の違いを述べる．

リーダにおける合理戦略

合理戦略をとるリーダは，サブタスクの依頼をする際，協調期待度とエージェントのリソースを勘案し，各サブタスクにつき L 体のメンバを選択する．

リーダにおける互恵戦略

互恵戦略をとるリーダは，信頼エージェントに対してサブタスクの依頼をする際，そのエージェントにのみ依頼し，他のエージェントに対しては，合理戦略と同様，各サブタスクにつき L 体のメンバを選択する．つまり，合理戦略で依頼するリーダに比べ，サブタスクごとのチーム参加依頼提案数の総数が小さくなるためサブタスクの受託率は下がる．しかし，この互恵戦略によって以下に述べる本モデルの課題を解決し，エージェントの学習を安定させることができる．本モデルはサブタスクの依頼をしたメンバと，実際にサブタスクを割当てるメンバが必ずしも一致するとは限らないため，学習が不安定になることがある．例えば本モデルではチーム参加依頼提案数 L を 2 とした場合，あるサブタスク一つにつきリーダが依頼すメンバ候補は 2 体となるが，その 2 体のメンバからの返答メッセージがどちらも受託であることもある．しかし実際にサブタスクを割当てるのはどちらか一方であるため，エージェントの信頼関係がここで相互に信頼とならない．これはリーダは受託したメンバ候補の両方の協調期待度を更新するが，メンバが実際にタスクを割当てられたリーダの協調期待度を更新するためである．この互恵戦略はリーダは相互に信頼関係を構築する可能性を高め，安定してタスク処理ができる．

メンバの合理戦略

合理戦略をとるメンバは，利得の取得 (チームに参加しサブタスクを処理すること) を優先し，受け取ったチーム加入依頼メッセージから自分が処理可能で，かつ協調期待度の

高いリーダからの依頼を選択する．

メンバの互恵戦略

互恵戦略をとるメンバは，信頼エージェントからの提案のみを選択対象とし，それ以外からの依頼は断る（拒否する）．このとき，受けた依頼に信頼エージェントからのものが無ければ，全ての依頼を断る．つまり，合理戦略をとるメンバに比べて，依頼を拒否するため「何もしない」状態である可能性が高くなるが，この行動がエージェントの協調関係を創発させ，リーダの互恵戦略とともに分散環境での安定したタスク処理を実現する．特に，この行動はシステムの負荷が大きくなったときに発生する競合に対して有効に働くと考える．たとえば，合理戦略をとるメンバ m が協調期待度の低いリーダ l からの依頼しか受けていない場合，(a) 報酬を得る可能性があること，(b) その後に到着する依頼を予測できないことの理由から，受託することになる．しかし，この直後に協調期待度の高いリーダ l から依頼を受けても m はサブタスクを処理しているため断らざるを得えない．そのためこのように合理戦略をとると l の m に対する協調期待度が低下し，学習が不安定になると考えられる．しかし，この互恵戦略はあえて拒否することでその可能性を減らし，チーム編成の成功率を高められる．

6.3.3 戦略決定方法

エージェント i は役割決定時に，信頼エージェントの有無によって合理戦略と互恵戦略のいずれかを選択する．具体的には，信頼エージェントが 1 体以上存在すれば互恵戦略を，その他は合理戦略を選ぶ．初期段階では $F^i = \emptyset$ のため，合理戦略をとる．なお信頼エージェントは協調期待度によって決まるが，この値は成功率，つまり報酬が期待できるものに対して大きくなる．そのため協調期待度は合理性の指標と言える．本研究では，適切なエージェントを決定する指標は協調が成立し報酬を得たことであるが，その値がある一定値を超えたときに，信頼できるエージェントとして認識し，その行動を変える．つまり基本的には報酬の追求がベースとなるが，それを高い確率でもたらす信頼できる相手を得たときに，その相手を優先するだけでなく，その相手の期待を裏切る機会を減らしている．

6.4 評価実験:戦略選択の有効性調査と可視化

6.4.1 評価実験の概要と比較手法

ここでは提案手法のタスク処理の効率を評価するために，多数エージェントが存在する環境で様々なシステム負荷を想定して実験した．実験は，一定期間内にエージェントが処理に成功したタスクの数で評価する．また，第 4 章や第 5 章とは異なり，信頼度に基づくエージェント間の構造がポイントの一つであるため，Cytoscape [77] を用いてエージェントの相互タスク依頼・受託の関係を可視化した．

比較手法

本実験で用いる比較手法は以下の 3 種類であるとした．

合理手法

エージェントは常に合理戦略に基づき行動する．そのため，リーダの場合は協調期待度を考慮し，各サブタスクにつきチーム参加依頼提案数依頼する．メンバは，チーム編成に成功する可能性 (協調期待度) の高いリーダからのチーム加入依頼を必ず選択する．これは提案手法において，常に信頼エージェント $F^i = \emptyset$ とした場合と同様であり，互恵戦略はとらない．

チーム固定手法 (SGS)

この手法では予め組織構造を与え，その組織内でのみチームを組むものである．これは，合理手法の対局にあるものであり，エージェントは常に決まった相手と協調する．この手法では，役割も予め決めておき，実験の開始時にリーダ 1 体，メンバ 5 体の，合計 6 体をランダムに決め，それを 1 チームとした SGS 6 と，リーダ 1 体，メンバ 6 体の，合計 7 体をランダムに決め，それを 1 チームとした SGS 7 の 2 つの手法を用意した．ここでチーム数を 6 および 7 とした理由は，タスクを構成するサブタスク数の最大値に合わせた (後述の実験におけるタスク参照)．また本手法に限り，エージェント数を 6 と 7 の倍数となる $|A| = 504$ とした．このためチーム数はそれぞれ 84 と 72 となる．

提案手法 2 (複数メッセージ)

この手法では，提案手法同様，合理戦略と互恵戦略を選択する戦略選択エージェントであるが，リーダが依頼をする場合，常に冗長的に依頼をするとする．具体的にはサブタスクの依頼をする際，各サブタスクにつき L 体のメンバを選択するとし，リーダは常に合

表 6.1: 実験おけるエージェント

パラメータ	値
エージェントの数 $\|A\|$	500
エージェントのリソースの種類数 p	6
エージェントの各リソース量 h_j^k	1 or 0
チーム参加依頼提案数 L	2
信頼エージェント判定基準 E^J	0.5

表 6.2: 実験におけるタスク

パラメータ	値
1 tick のシステム負荷 λ	5 から 30
$s \in S_T$ の各リソース量 r_j^k	1 or 0
一つのタスクに含まれるサブタスクの数 $\|S_T\|$	3 から 6 のランダム

理戦略を取る．他方，メンバは提案手法と同じとする．そのためメンバは信頼エージェントの有無によって合理戦略か互恵戦略を選択する．この提案手法 2 (複数メッセージ) と本提案手法の比較によって，提案手法の互恵性に依存したチーム加入依頼メッセージの違い (リーダの互恵戦略の有効性) を確認できる．また，この提案手法 2 (複数メッセージ) と合理手法の比較によってメンバの互恵戦略の有効性を確認できる．

6.4.2 実験 1:システム負荷毎のタスク処理効率調査

提案手法とその他の比較手法において，タスク処理効率を調査する．ここではシステム負荷 λ は $5, 10, 15, 20, 25, 30$ とし各手法のタスク処理成功数の推移を比較した．また，提案手法による効率化の理由を探るため，最終的に得られたエージェント間の構造を可視化した．実験設定の一覧を表 6.1〜表 6.3 に示す．

各システム負荷のタスク処理成功数の推移

ここで示す結果は 50 tick 毎の各手法のタスク処理数を測定し，50000 tick までプロットした．また，実験結果は 30 回試行の平均値である．λ の値を変化させた実験の結果を

表 6.3: 学習パラメータの設定

パラメータ	値
協調期待度の初期値	0.1
役割適正値の初期値	0.5
学習率 α	0.05
ε-greesy 選択の ε	0.01
減少数 α_F	0.00005

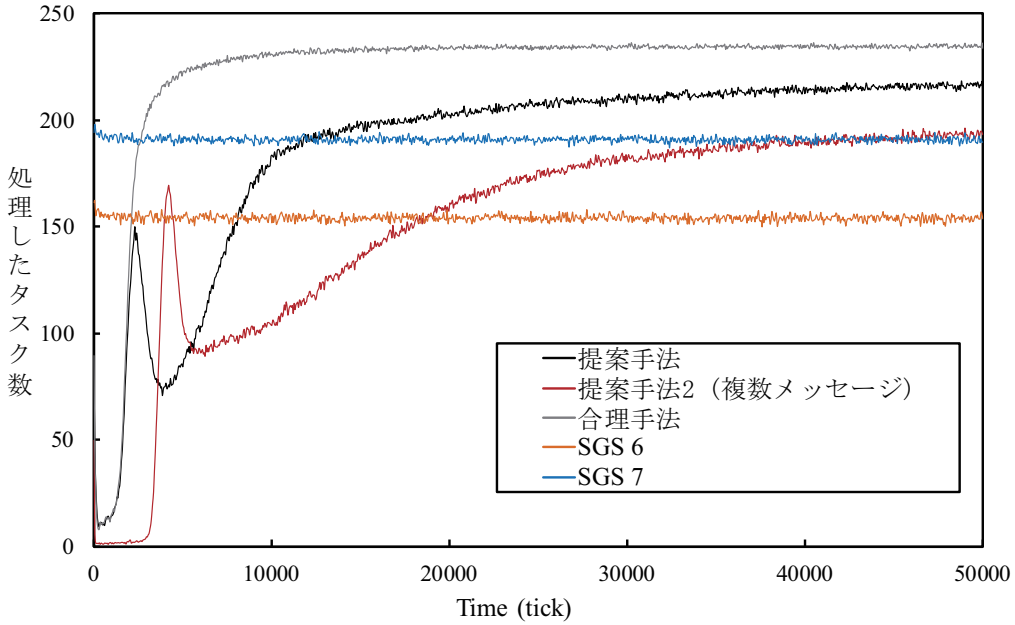

図 6.2: $\lambda = 5$ における各手法のタスク処理数の推移

図 6.2〜図 6.7 に示す．

ここで，$\lambda = 5$ は，システム負荷が小さく競合が発生しにくい環境であり，$\lambda = 25$ は，システムの処理限界の手前の環境である．また，$\lambda = 30$ は，キュー溢れ（ドロップ）が発生しており，非常に負荷が高い環境を想定している．

$\lambda = 5$ (図 6.2) は，負荷が小さい状況であるため，チーム編成時にメンバが複数のリーダから同時に依頼を受ける，または実行中に別のリーダから依頼を受ける，といった競合の発生頻度が低い環境である．このため，エージェントは互恵戦略による拒否をせず，可能性のある依頼を常に受託する合理戦略をとる方が多くのタスクを処理でき，すべてのエージェントが合理戦略をとる合理手法が最も効率が良くなる．

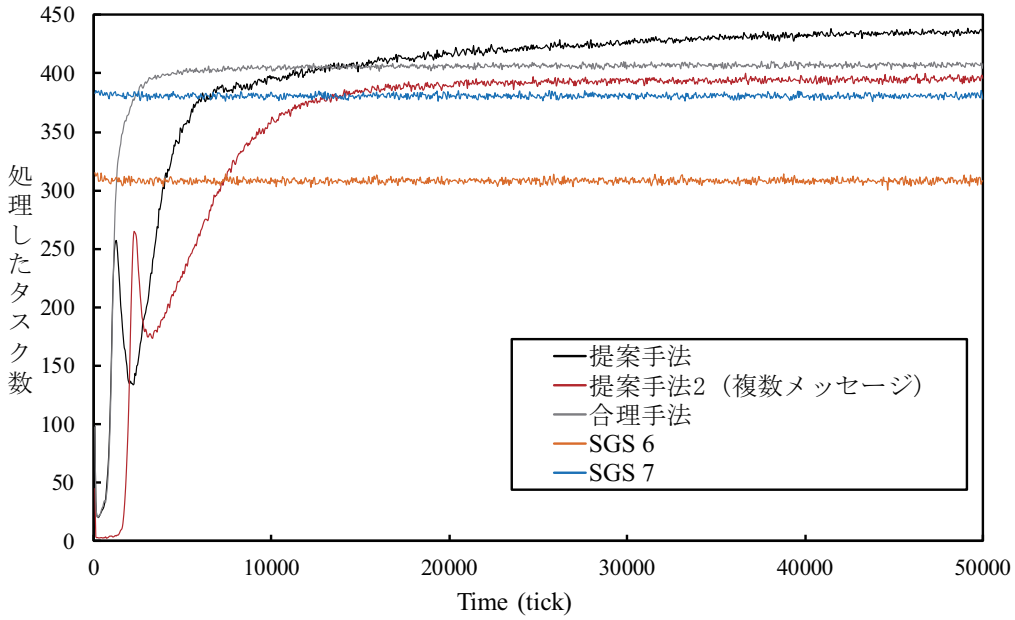

図 6.3: $\lambda = 10$ における各手法のタスク処理数の推移

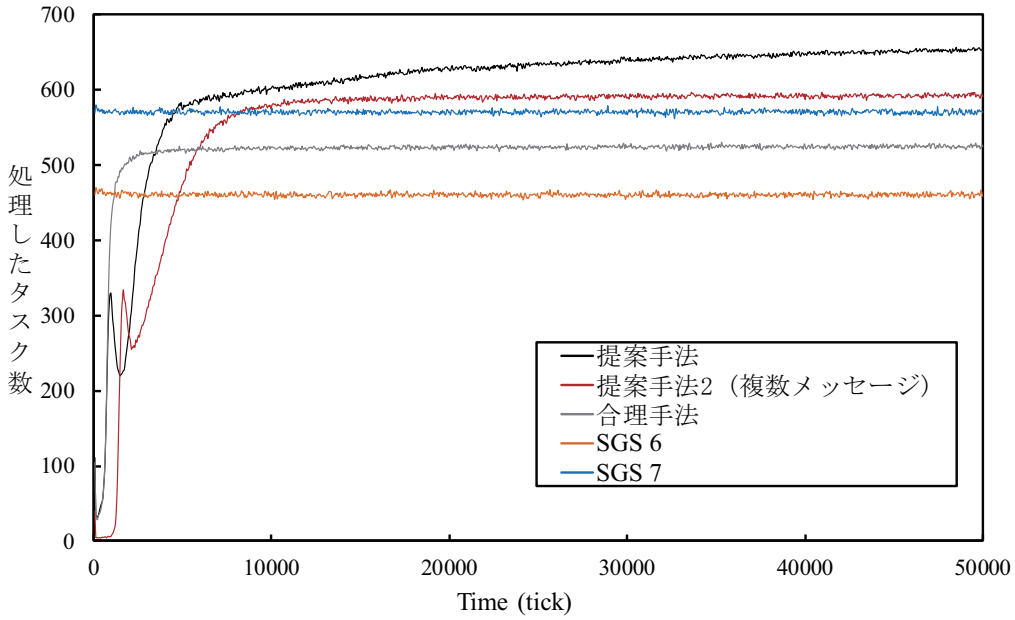

図 6.4: $\lambda = 15$ における各手法のタスク処理数の推移

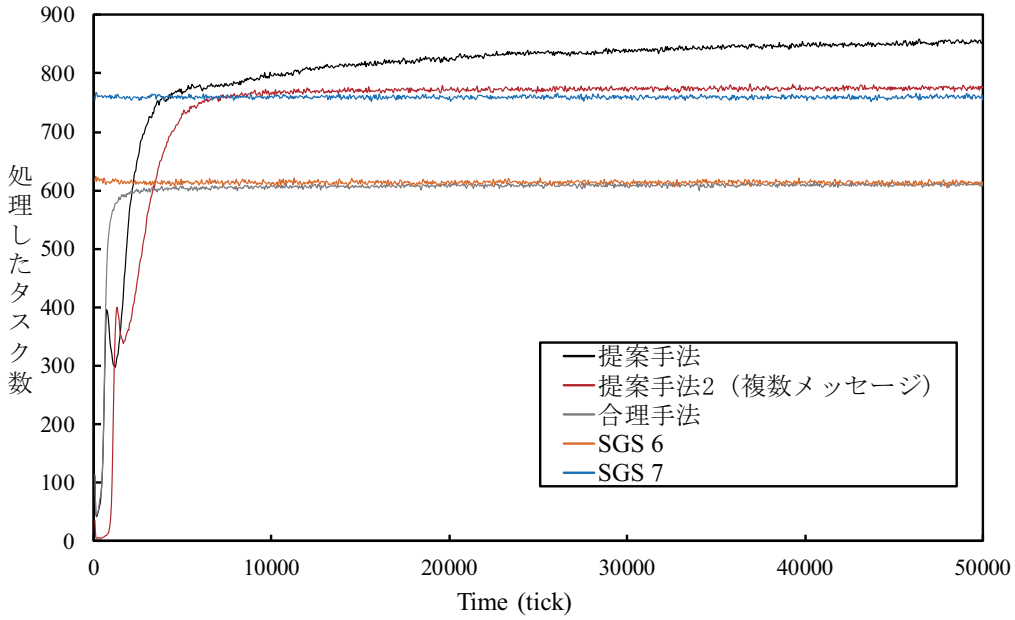

図 6.5: $\lambda = 20$ における各手法のタスク処理数の推移

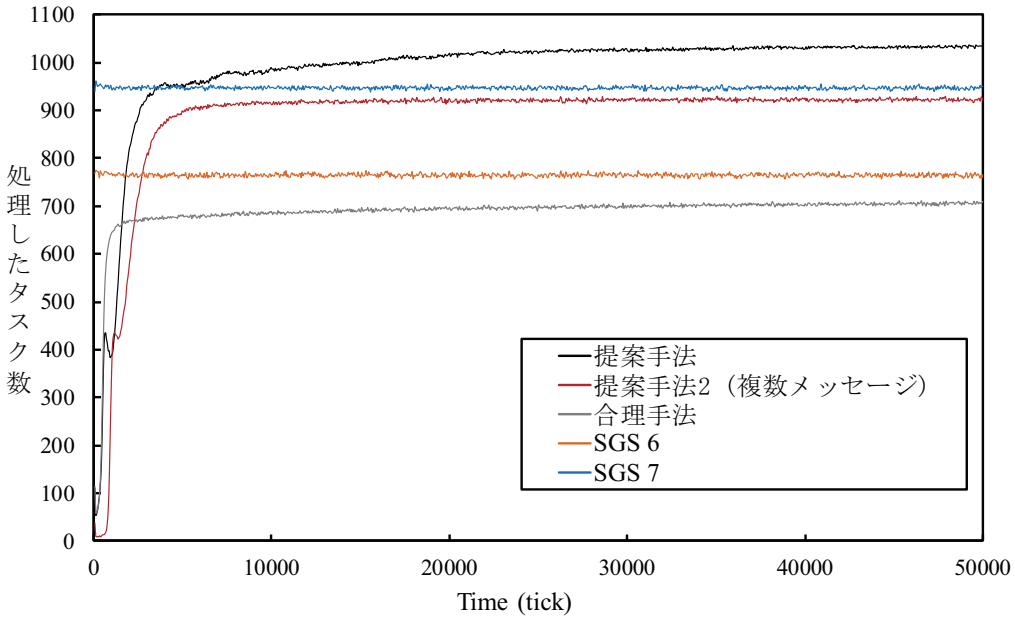

図 6.6: $\lambda = 25$ における各手法のタスク処理数の推移

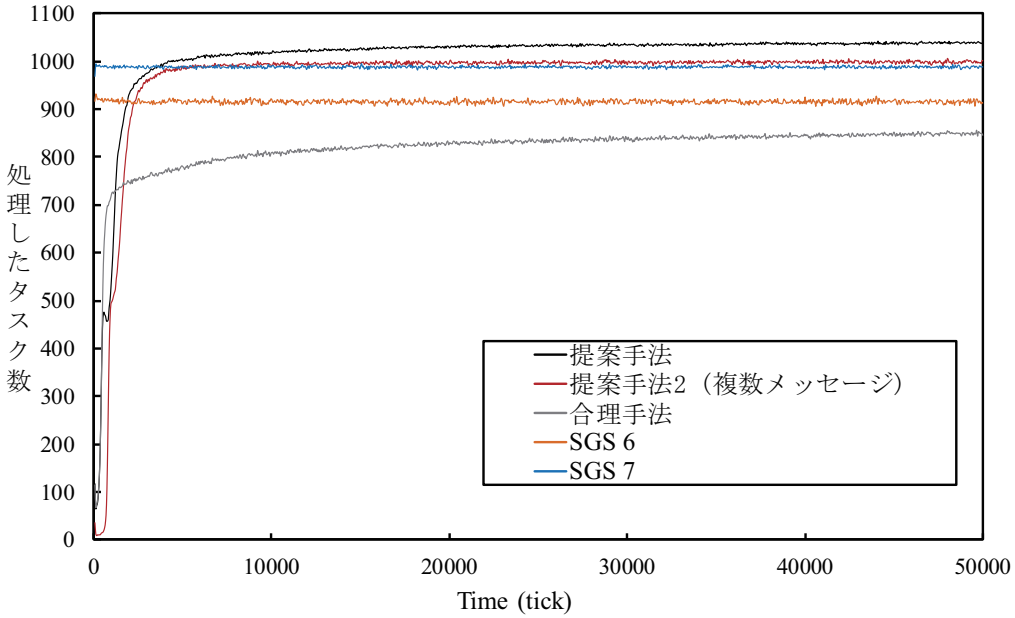

図 6.7: $\lambda = 30$ における各手法のタスク処理数の推移

また，$\lambda = 10$〜25 (図 6.3〜図 6.6) では，本提案手法が最もタスク処理の効率化を実現している．一方で，$\lambda = 30$ (図 6.7) では提案手法が最も効率は良いが，その他の手法とのタスク処理成功数の差は小さくなっている．特に $\lambda = 25$ (図 6.6) のときは，合理手法の効率が極めて低いが，これは個別の学習と自律的な合理的判断により多くの競合を招き，学習が不安定になったためと考えられる．SGS 手法はどちらも，提案手法に比べて効率は良くない．これはチームのメンバおよび役割を固定しており，タスクの処理はできるが，タスクのサイズによっては働いていないエージェント (サブタスクの割当がないエージェント) も存在し，それが効率の限界になる要因と思われる．

効率化割合の変化

ここではシステム負荷 λ を 5 から 30 まで 5 ずつ変化させた上記の実験結果 (図 6.2〜図 6.7) において，各手法の 40000 から 50000 tick 間で処理したタスク数を集計し，比較手法に対する提案手法の効率化割合を以下の式で求めた．

$$\frac{T(\text{提案手法}) - T(\text{比較手法})}{T(\text{比較手法})} * 100\ (\%) \tag{6.5}$$

この値が大きいほど，提案手法は比較手法に比べてタスク処理の効率が良いことを表す．ここで，T(比較手法) は，比較手法ごとのタスク処理数を表し，提案手法 2 (複数メッセージ)，合理手法，SGS6，SGS7 のいずれかである．

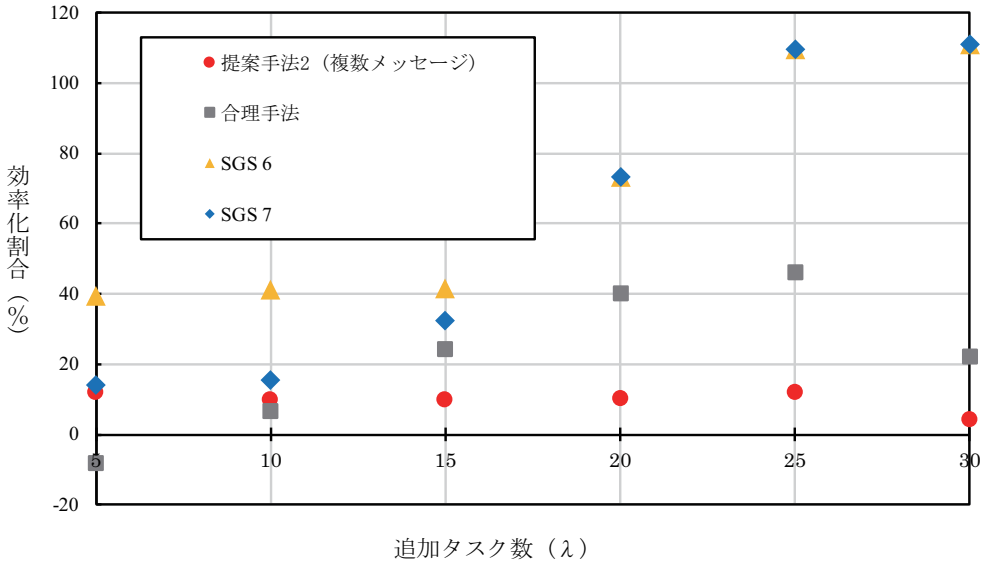

図 6.8: 提案手法の効率化割合

この実験結果を図 6.8 に示す．図 6.8 から，効率化割合は λ の値に大きく依存することがわかる．まず合理手法の効率化割合は，$\lambda = 5$ のとき提案手法を上回っており，効率化割合は負の値をとる．しかし，その後 λ が 25 前後で提案手法との差がピークとなり，λ が 30 の場合はその差が縮まる．また，SGS6 および SGS7 との効率化割合を比較すると，提案手法が常に効率を上回っているが，λ が 15 を超えると急激に差が大きくなった．この実験の結果から，負荷の低いときには，信頼エージェントに基づく互恵戦略はやや効率を落とす可能性がある．しかし重要なのはシステム負荷が大きくなったときの効率であり，提案手法は負荷が大きい場合にも網羅的に対応できている．

エージェント構造

各手法で得られた最終的なエージェント間のメンバの依頼・受託関係の構造を調査するために Cytoscape [77] を用いて可視化した．各ノードはエージェントであり，役割と 50000 tick 時で選択していた行動戦略で分類し表示させる．その一覧を表 6.4 に示す．ここでリーダエージェントとは，役割選択でリーダを選択するエージェントである．このため，リーダ適正値がメンバ適正値より大きいエージェントとなる．そのほかのエージェントは，メンバを役割を選択するエージェントである．このため，メンバ適正値がリーダ適正値より大きいエージェントとなる．また，メンバのみその行動戦略ごとに分類した．ここで生成した構造は $\lambda = 20$ のときに実験の最後の 5000 tick (45000 から 50000 tick まで)

表 6.4: ノードの設定

パラメータ	値
リーダエージェント	円形（赤）
合理戦略をとるエージェント	三角形（緑色）
互恵戦略をとるエージェント	四角形（青色）
チーム固定手法のメンバエージェント	四角形 (黄色)

において，チーム編成に成功した回数が 10 回以上のときのみに実質的な共同関係があると考え，そのチームのリーダとメンバの間にリンクを生成した．なおここで 10 回としたのは，その値の前後でギャップが存在すると思われたからである．ただし，SGS6 および SGS7 は役割とチームのメンバが固定であるため生成したエージェント構造を可視化した．提案手法の構造を図 6.9 に示す．また，提案手法 (複数メッセージ) のエージェント構造は全体図 6.10 と合理戦略をとるエージェントを削除しリーダと互恵戦略をとるエージェントのみを出力した図 6.11 の 2 種類とした．合理手法は図 6.12，SGS6 を図 6.13，SGS7 を図 6.14 に示す．

SGS のエージェント構造はあらかじめ全チームが 6 体あるいは 7 体で固定したものが出力されている．また，図 6.10 と図 6.12 から，合理手法および提案手法 2 (複数メッセージ) は，複雑な構造になっているが，これは合理戦略をとるエージェントがチーム編成の度に，リーダは冗長的にメンバへ依頼し，またメンバは異なるリーダからの依頼を受託することで，結果的に多様なリーダとチームを組んだためである．合理戦略をとるエージェントは，協調期待度があまり高くなくても，その他のリーダからの依頼がなければ，その時点での協調期待度最大のリーダの依頼を受託するため，常に同じエージェントでチームが編成されるとは限らないため，このような複雑な構造となる．ここで，提案手法 2 (複数メッセージ) の互恵戦略をとるエージェントの共同関係（信頼関係）の確認のため，合理戦略をとるメンバエージェントを除いた図 6.11 を参照すると，リーダを含め 4 体前後のチームが構成されており，共同関係によるチームの組織化が見られる．なお，この図 6.11 の上部分では，複数のリーダとつながる互恵戦略をとるエージェントが数体存在するが，これは合理戦略と互恵戦略を選択するタイミングが原因である．可視化結果の集計期間は，最後の 5000 tick 間であり，その間に選択する行動戦略を変える可能性がある．そのため集計開始段階では合理戦略をとるエージェントとして，複数のリーダとチームを編成し，途中で互恵戦略に変化したと考える．そのため互恵戦略をとり，信頼エージェント

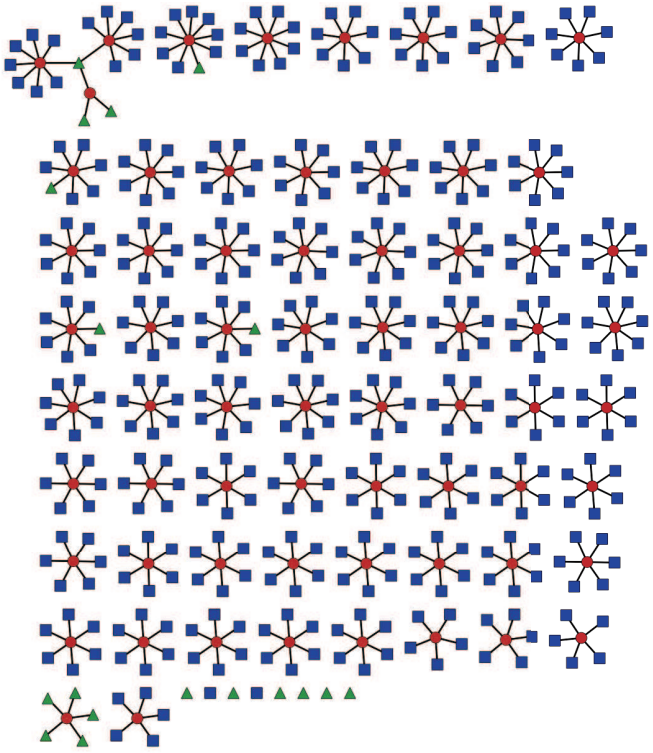

図 6.9: 提案手法のエージェント構造

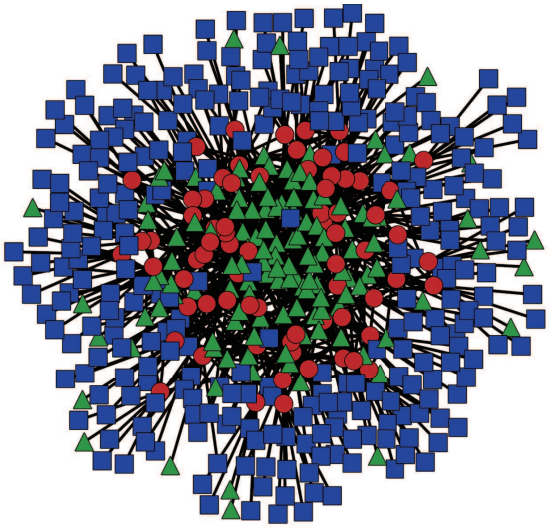

図 6.10: 提案手法 2 (複数メッセージ) のエージェント構造

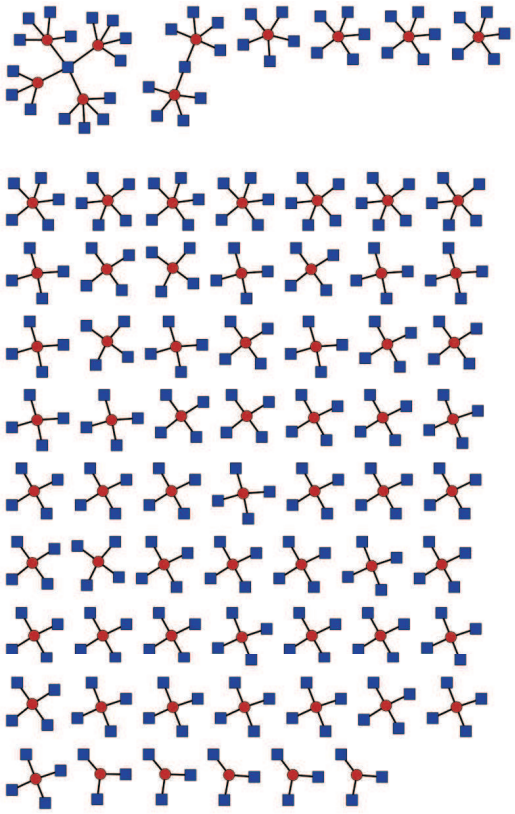

図 6.11: 提案手法 2 (複数メッセージ) のエージェント構造 (合理戦略のエージェント削除)

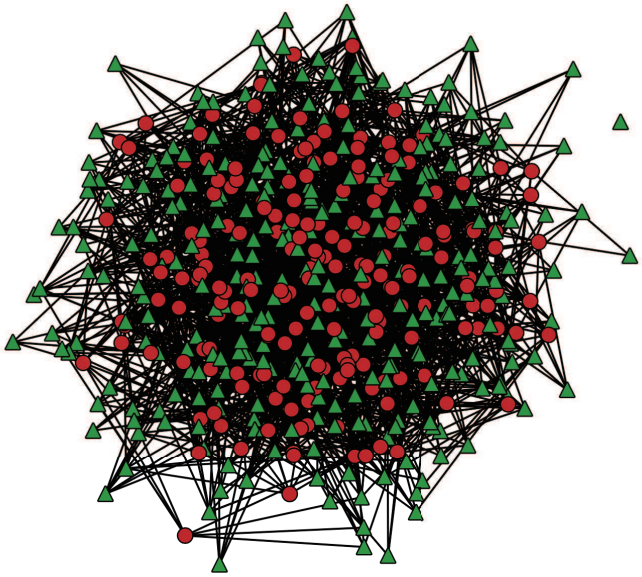

図 6.12: 合理手法のエージェント構造

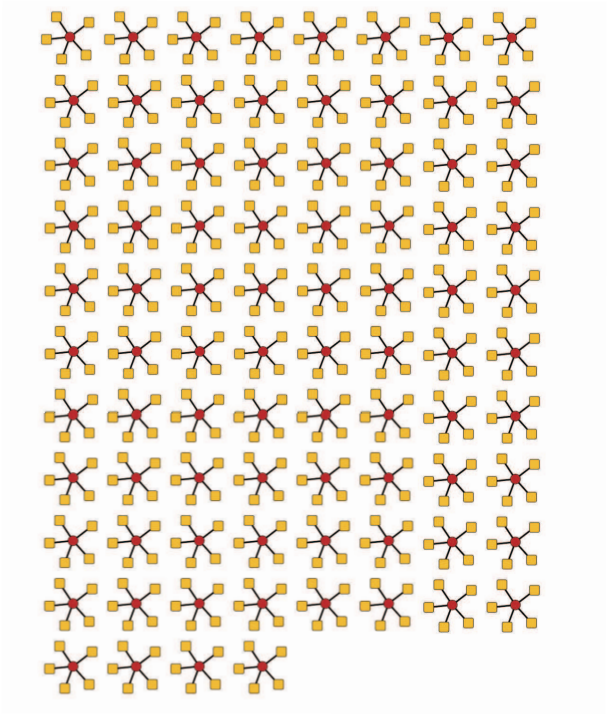

図 6.13: SGS 6 のエージェント構造

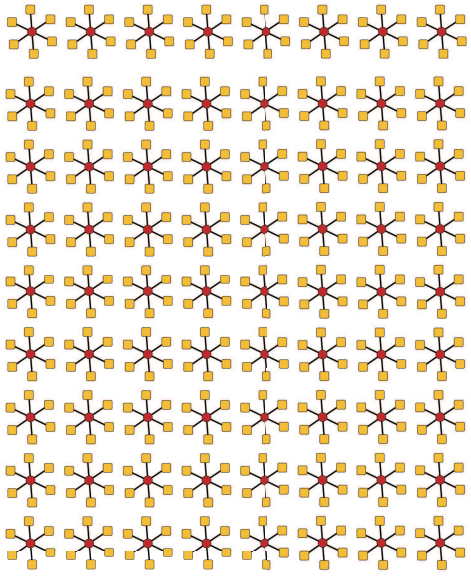

図 6.14: SGS 7 のエージェント構造

のみの依頼しか受託していないにも関わらず複数のリーダとエッジが残っている．また，提案手法は，リーダおよびメンバが互恵戦略をとることで相互に信頼性を構築しており，提案手法 2 (複数メッセージ) のように合理戦略をとるメンバエージェントを削除しなくともその提携が確認でき，すでに安定していることがわかる．この提携が様々なシステム負荷に対応できたと考える．

実験 1 のまとめ

ここでは，実験 1 の結果をふまえ，それらをまとめて考察する

まず，図 6.8 および図 6.2〜図 6.7 の結果では比較手法に比べ，提案手法によるタスク処理数が多いことを示している．これは，互恵戦略をとるエージェントが，信頼エージェント以外のチーム加入依頼を断ることで，徐々にメンバの組織化・構造化を進めたためと考えられる．特に提案手法は，リーダも互恵戦略をとるため，提案手法 2 (複数メッセージ) と比較しても効率を示している．一方，メンバが合理戦略をとる場合，第 6.4.2 節でも述べたように学習結果が不安定になる．その結果，チーム編成も安定しなくなるため，提案手法とその他の手法とで効率に差が現れた．ただし λ が小さいときには，そもそも競合が発生にしにくく，合理的な行動をとる方が優位になるため，λ が 5 の場合は合理手法が最も効率が良い．

効率化割合を示す図 6.8 は，システム負荷 λ が小さいときと，システム負荷が高くなりキューからタスクがドロップし始める少し手前あたり以降に効率化の割合が差が小さくなり，λ が 25 程度の負荷は高いが限界よりはやや手前の状態で効率化のピークを迎える．これは [78] が指摘した組織化によるスイートスポット (sweet spot, SS) の存在にも関係する．この論文では，マルチエージェントシステムにおける組織化の意義を調査するために，ロボカップレスキューを題材に組織構造の有無による効率の違いを調べた．その結果，系の負荷が小さいときには効率上の差は小さいが，負荷が上がるとともに組織構造を持つ系の相対的効率が上昇する．この上昇は，系の限界よりやや手前でピーク (SS) となり，さらに負荷が大きくなり限界を超えるとその差が再び小さくなる．本研究で，図 6.8 の合理手法との効率化割合も同様なカーブを描いており，同じ現象があると考えられる．他方，組織化が適切でない場合には，その効率が下がる．

提案手法と SGS6 および SGS7 手法では，ともにエージェント構造が導入されているが，特に後者 2 手法は有効性を示せていない．これはやはり Daniel D. Corkill, Daniel Garant, and Victor R. Lesser[78] が指摘したように，構造を初めから適切に与えることは難しく，

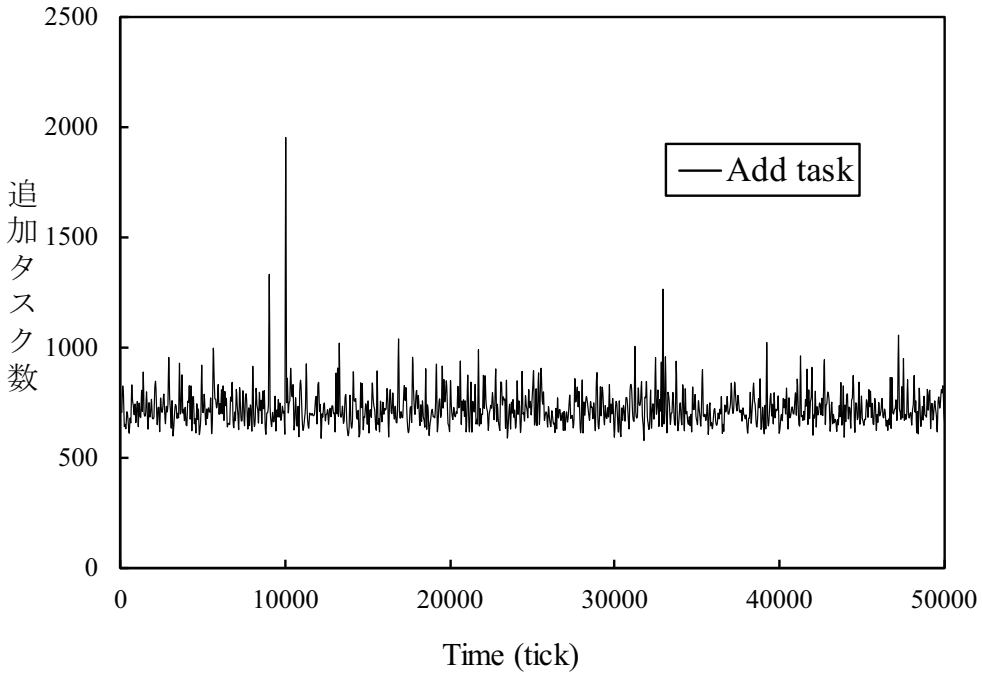

図 6.15: システム負荷の推移

それが適合していないとかえって効率を落とすことを示唆している．特に SGS では，リーダとメンバを予めランダムに決め一チームとしたため，チーム内のエージェントのリソースが偏ることがある．そのため，例えば処理能力の低いエージェントのチームや特定の能力のみを持つエージェントのチームが作られ，それが適合していなかった (処理できないタスクがあった) と考える．提案手法ではチーム編成成功率に基づいた共同関係をボトムアップに構築するため，チームのエージェントのリソースを環境に合わせて学習し動的に調節できる．

6.4.3 実験 2:システム負荷が動的に変化する環境での実験

実験 2 では，システム負荷 λ が実験途中で変化する環境を想定して実験した．実世界では，一定の数や種類のタスクが毎回追加されることは稀であり，それらは変化する．そこで，本実験では，パレート分布によりシステム負荷を変化させ，急激な環境変化を再現した．図 6.15 に 50 tick 毎のシステム負荷 (追加されるタスクの数) の推移を示す．この環境においてエージェントはチームを編成しタスクを処理する．実験条件はシステム負荷 λ 以外は実験 1 と同様である．

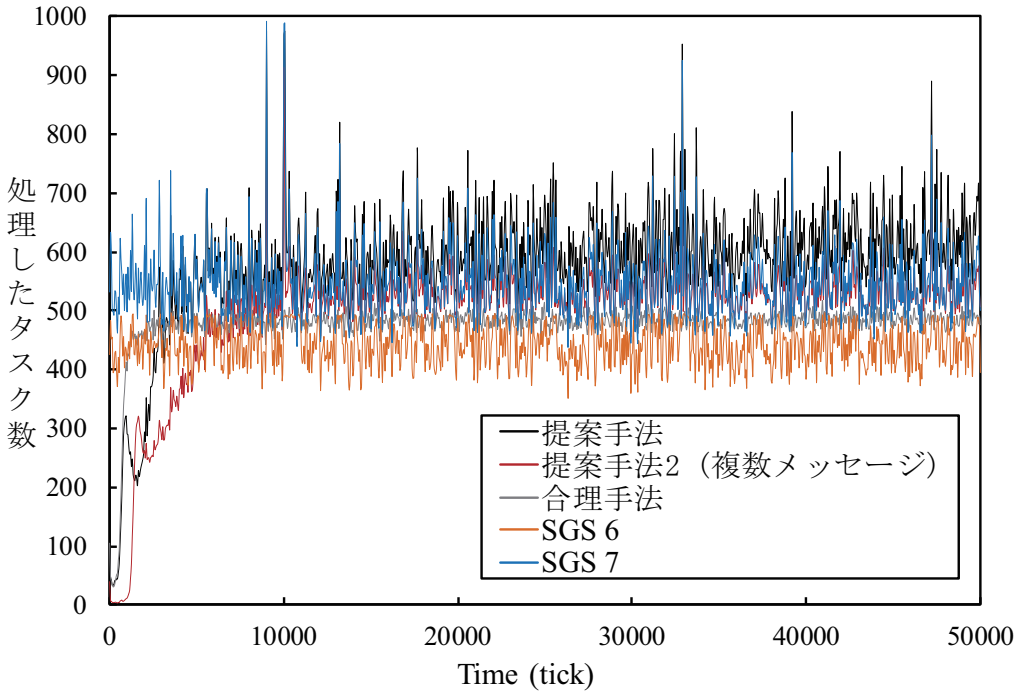

図 6.16: パレート分布における各手法のタスク処理効率

実験 2 の結果

実験 2 の実験結果は以下のように示す．本実験では，環境の急激な変化によってタスク処理数も変化するため，各手法の結果が重なり結果が見づらくなる．そのため，全手法のタスク処理数の推移をまとめた結果を図 6.16 に示し，各手法毎のタスク処理数の結果を図 6.17 から図 6.21 に示した．結果は 50 tick 毎の各手法のタスク処理数を測定し，50000 tick までプロットした．また，実験結果は 30 回試行の平均値である．

実験結果から，提案手法，提案手法 2 (複数メッセージ)，SGS7 手法は急激な環境変化に対応しているが，その他の手法は対応できなかった．

実験 2 の考察

本実験では，システム負荷，つまりタスクの要求数が動的に変化するため，その数に応じた「チーム数」が必要となる．これは本モデルはタスク一つにつきチームが一つ形成されるためである．このことから実験 2 では，要求の増減に対応できるチームを安定して編成できる点が重要となる．

結果から，提案手法 (図 6.17) および提案手法 2 (複数メッセージ) (図 6.18)，SGS7 (図

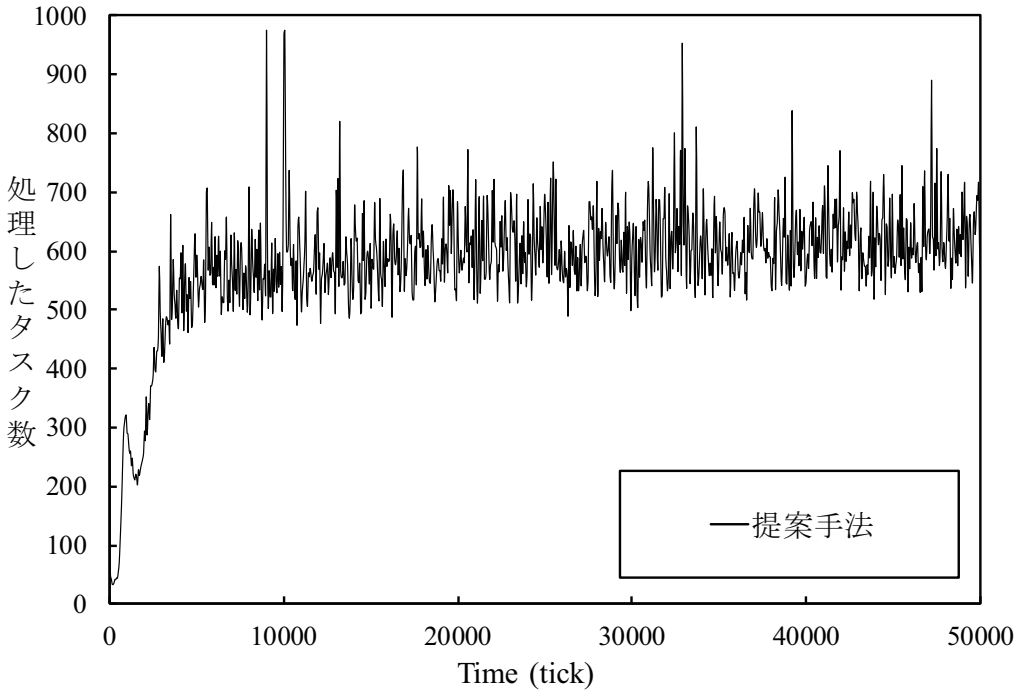

図 6.17: パレート分布における提案手法のタスク処理効率

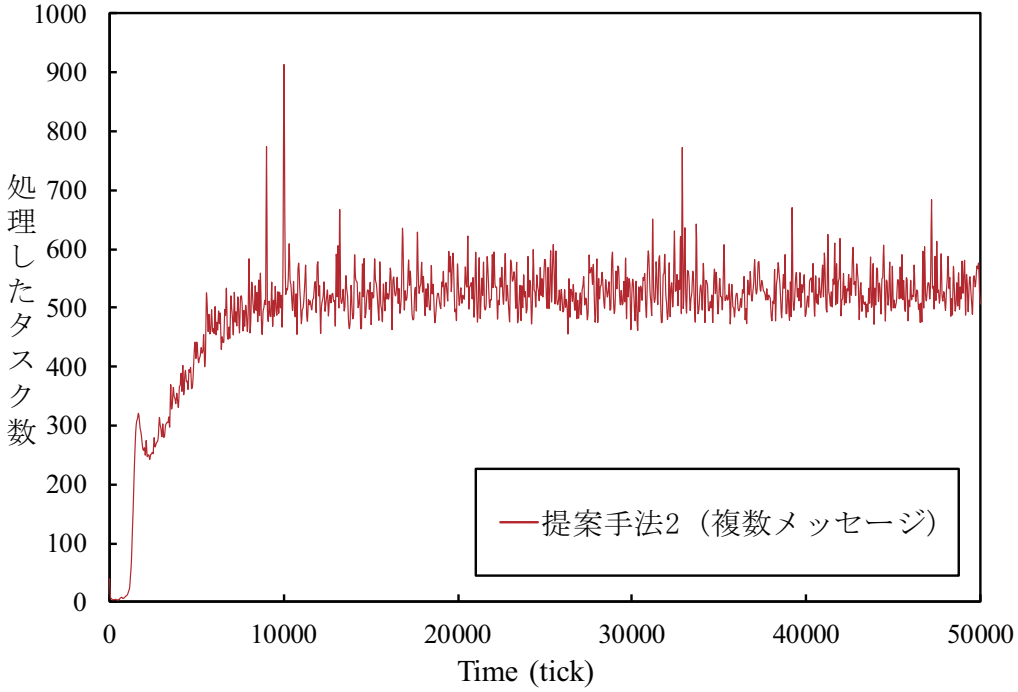

図 6.18: パレート分布における提案手法 2 (複数メッセージ) のタスク処理効率

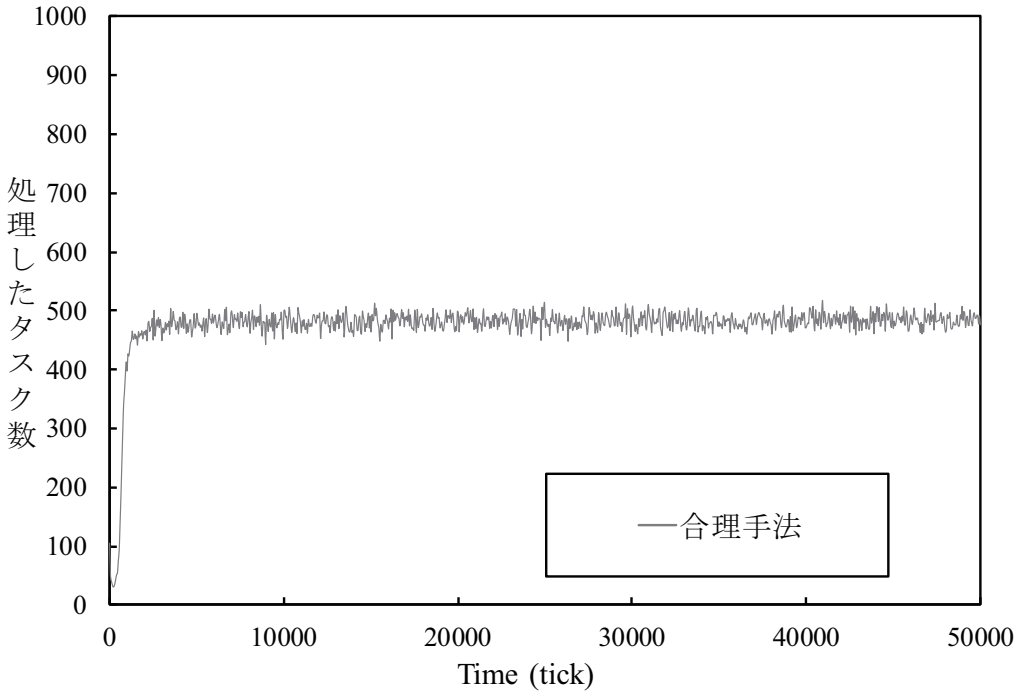

図 6.19: パレート分布における合理手法のタスク処理効率

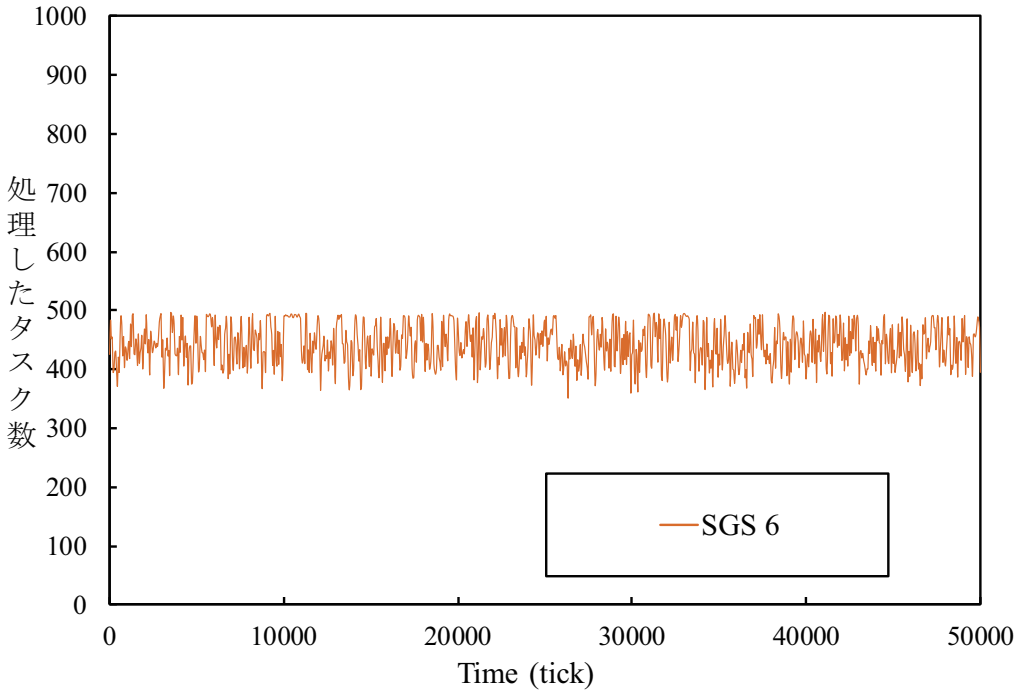

図 6.20: パレート分布における SGS 6 のタスク処理効率

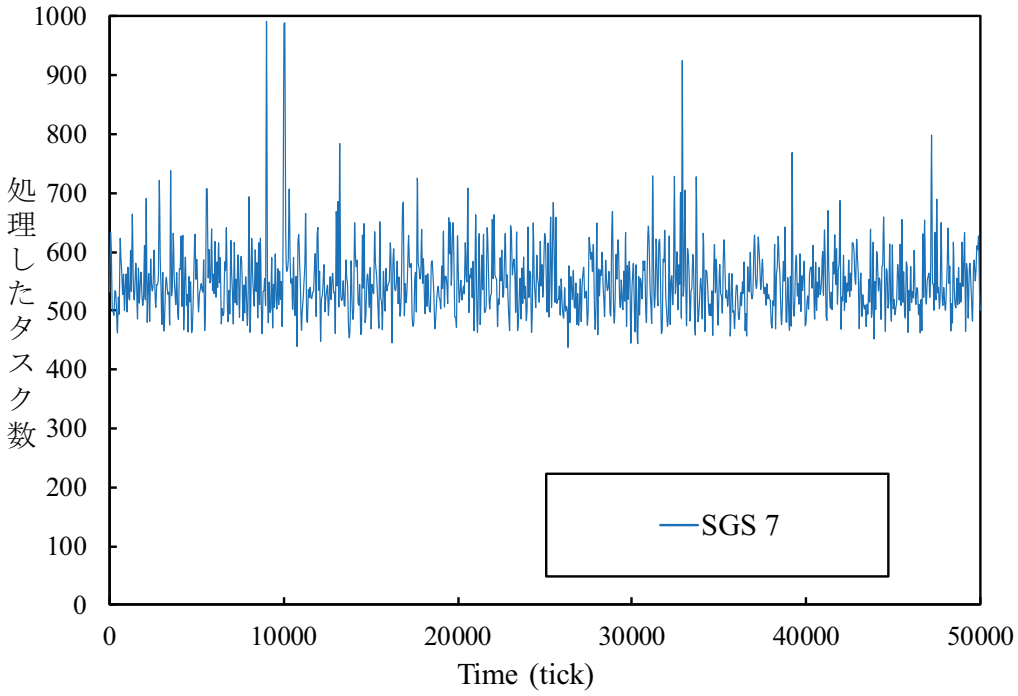

図 6.21: パレート分布における SGS 7 のタスク処理効率

6.21) は変化に対応できることを示した．この要因として提案手法及び提案手法 2 (複数メッセージ) はエージェント学習と戦略選択によって安定的なチームが編成できるためである．特に提案手法は，リーダが信頼できるメンバに対してチーム参加依頼提案数を 1 とし，メンバも信頼できるリーダからのみの依頼を受けるため，これらの互恵戦略が有効に働いており，提案手法 2 (複数メッセージ) に比べて相互に信頼した頑健な提携が形成されたため効率がよい．これは実験 1 の可視化結果からも明らかである．その頑健性によって，要求が増減して数が変化してもその変化に対応できる．また，SGS7 は，あらかじめ役割やチームのメンバ数を設定しており，システム負荷が変化しても用意したチームで対応できる．ただし，これは実験のエージェント数，タスクに含まれるサブタスクの数 (メンバ数) を元に計算してチームを設定しており，それらが予測できない分散環境では応用できないと考える．本手法は動的にチームを編成し，さらにシステム負荷の変化に対応可能な提携構造をボトムアップに構築できた．また，SGS6 (図 6.21) は SGS7 に比べて環境変化に対応できていないが，これはチームとして処理できるタスクが少ないためである．これは実験 1 の考察でも議論したが，チームのメンバがもつリソースに偏りが生じ，処理できないタスクが発生するためである．このことからここでもシステム開始時にチームを予め構築する手法は環境変化に対応できず．ボトムアップにチームを編成する重要性を示すことができた．また，合理手法 (図 6.19) はタスク処理効率が一定であり，そもそもシ

表 6.5: 実験におけるタスクサイズ $|S_T|$

時間 (tick)	一つのタスクに含まれるサブタスクの数 $|S_T|$
0-99999	3-6 のランダム
100000-149999	4-8 のランダム
150000-200000	3-6 のランダム

ステム負荷の変化に対応できていない．これは全エージェントが合理的な行動をとることで競合が発生するため，安定してチーム編成ができず，一定数のチームしかタスク処理に成功しないためである．

以上のことから，システム負荷が変化する環境では，タスクの増減に合わせたチーム数が必要であるが，結果から本手法は，環境変化に対応でき，更にタスク処理も効率化できた．

6.4.4 実験 3:タスク構造が変化する環境での実験

実験 3 では，タスク構造，つまりタスクに含まれるサブタスクの数が実験途中で変化する場合を想定して実験した．これまでの実験ではタスクに含まれるサブタスクサイズ $|S_T|$ を 3 – 6 のランダムで変化させていたが，ここでの実験はより大きな変化で実験した．また，システム負荷 λ を 15 と固定し，時間経過におけるタスクサイズ $|S_T|$ を以下の図 6.5 の設定で変化させて実験した．その他の実験条件は実験 1 と同様である．

実験 3 の結果

実験 3 の実験結果を図 6.22 に示す．結果は 50 tick 毎の各手法のタスク処理数を測定し，200000 tick までプロットした．また，実験結果は 30 回試行の平均値である．

実験結果から，100000 tick までは，$\lambda = 15$ の場合と同様な結果が得られている．その後，100000 tick から 150000 tick までは構造が変化し，サブタスクの数が増加する．本モデルではタスクに含まれるサブタスク一つにつき 1 体のメンバが必要であるが，このときタスクを処理をするために必要なメンバ数が増加する．一方でシステム負荷は変わらないため，エージェントが不足し，タスク処理効率は全手法で低下する．しかし，提案手法と提案手法 2 (複数メッセージ) は一時的に効率が落ちるものの，再度タスク処理効率が

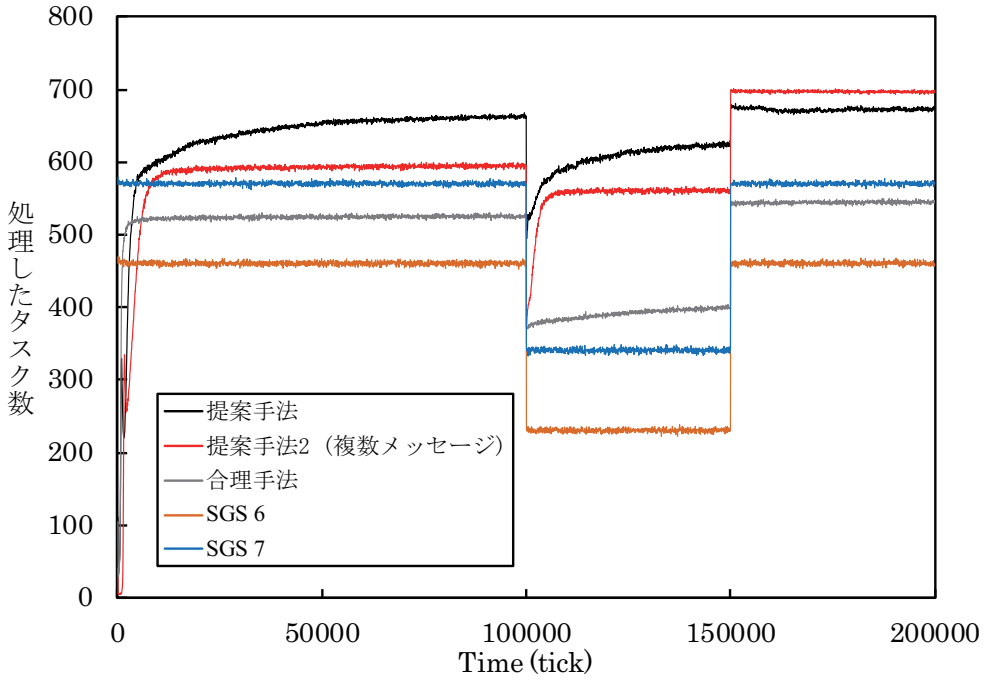

図 6.22: タスク構造変化環境における各手法のタスク処理効率

向上しておりその変化に対応している．SGS は効率が落ちた状態を維持したまま変化しなかった．また，150000 tick 以降になるとシステム開始時と同様のタスク構造に戻るが，提案手法もその他の手法もその変化に対応できている．

実験 3 の考察

ここではタスクサイズの変化に応じた各 tick 毎に分類して考察する．

初期：0 tick から 100000 tick まで

この期間におけるタスク処理は実験 1 での $\lambda = 15$ までと同様の結果が得られている．そのため，実験 1 でも考察したが提案手法はエージェントの学習によってシステム負荷に対応している．

中期：100000 tick から 150000 tick

この期間は初期に比べてタスクサイズが変化し，チーム編成に必要なメンバ数が増加する．そのため，どの手法も初期に比べてタスク処理数が低下する．しかし，提案手法は，

一時的に効率が低下するが，学習によってその変化に対応している．また，提案手法 2 (複数メッセージ) も提案手法と同様，学習を進めているが収束が速い．この 2 手法の収束速度の違いはメンバへ送るチーム加入依頼メッセージ数の違いに起因する．提案手法 2 (複数メッセージ) はサブタスク一つにつき常に 2 通依頼を行っているため必要なメンバを同定し易いこと，さらに実験 1 の可視化結果 (図 6.10) からも合理戦略をとるエージェントが一部存在しているため，タスクサイズの変化に柔軟に対応できるためである．これは後述するがエージェントが会社の正社員とアルバイトのような形態を取っていることが考えられる．SGS6 と SGS7 は，予めチームのメンバ数を固定にしているため，タスクサイズが変化した環境には対応できない．このことからやはり，予めエージェント構造を決めることは分散環境では適さないことがわかる．

後期：150000 tick 以降

この期間は，システム開始時と同様の負荷がかかる環境であり，チームを編成するために必要なメンバ数が初期の段階と同じサイズに戻る．結果から各手法はそれぞれ初期と同様の結果が得られている．特に，SGS6，および 7 はチーム数，メンバ数は変化していないため，環境の変化によって効率が変わることなく，変化前と全く同じ効率を得ている．ここで注目すべき点は，合理手法，提案手法，提案手法 2 (複数メッセージ) は，初期と比べて効率が向上していることである．この要因として，タスクサイズが変化したことで学習が精査されたことが要因である．中期でタスクサイズが大きくなるとともにチームのエージェント数 (メンバ) が増えるが，そのとき協調期待度も合わせて学習する．このとき初期で学習したエージェントに加えて新しく有用なエージェント (例えばリソースの種類が多い) を取り入れてチームを編成できる．その後，後期で元のタスクサイズに戻ったときに，増えたメンバから一番適合しなかったエージェントが除かれ，前のチームより精査されたと考えられる．特に，提案手法 2 (複数メッセージ) は，中期での考察でも述べたが提案手法に比べて合理戦略をとるエージェントが多く，今回のようにタスクサイズの変化によってチーム構造を変化させる必要がある場合，より多くのエージェントから選抜できるため柔軟に対応できる．そのため，後期では提案手法よりも効率を上げることができた．

以上のことから，このようなタスクサイズの動的な変化の対応には，エージェントが会社のアルバイトと正社員のような関係を構築する必要があると考える．具体的には，まずは通常タスクの負荷 (今回の場合サブタスクのサイズ) が一定の場合は正社員 (互恵戦略をとるエージェント) で確実にタスクを処理する．次にもし人手が必要なタスクが発生した場合 (今回の場合サブタスクのサイズが増えたとき)，暫時的な労力をアルバイト (合理戦

略をとるエージェント) に依頼して補い，その後学習をすすめる．その結果，今回のような変化に対応でき効率的にタスクを処理できる．特に，この様子は提案手法 2 (複数メッセージ) で顕著に現れたと考える．

これらの結果から，タスク構造が変化する場合，その変化に柔軟に対応するために，ある程度合理戦略をとるエージェントの存在が必要であることがわかった．提携を密にしすぎると，余剰にエージェントが必要になったときにその確保が遅れることも明らかになった．

6.5 本章のまとめと課題

エージェント数が増加した大規模環境へのアプローチとしてエージェントの提携を協調関係に基づきボトムアップに創発することで効率的なタスク処理を実現した．特に本手法は互恵性に基づく新たな戦略として互恵戦略を導入し，信頼エージェントの有無によって合理戦略と戦略を選択する戦略選択エージェントを提案し，大規模環境かつ変化する環境でも対応できることを示した．一方で，本手法はエージェント関係を密にすることで様々な負荷に網羅的に対応することを示したが，密にしすぎた結果，その提携の解除や再学習が遅れることも課題として明らかになった．

第7章　総括

7.1　まとめ

本研究は分散環境での効率的なタスク割当の実現を目指した．そのために，エージェントのタスク割当チーム編成モデルを導入した．これは，エージェントの役割や協調すべき相手をエージェントが自律的に判断して選択するモデルである．また，従来のマルチエージェントシステムにおける研究や実世界の課題を以下の 2 つに分類しそれらを解決する手法を提案した．

(A) エージェントの能力が収集できないリソースが未知な環境

(B) エージェント数および要求が増加した環境

まず，(A) の解決のために第 4 章ではリソース推定手法とエージェントの学習を提案した．リソース推定手法では，エージェントが自分以外のエージェントの能力を推定・学習するためのリソース推定パラメータを導入し，リソースが未知な環境でのタスク処理を実現した．また，合わせて提案したエージェントの学習によってエージェントが自分の役割と協調相手を学習し，分散環境でも自律的かつ効率的なチーム編成ができることを示した．

次に，(B) の解決のために，第 5 章では保持する情報をスコープして削減する手法，第 6 章では互恵性に基づいた共同関係を構築する手法の 2 つを提案した．前者では，個々のエージェントが持つ情報量 (今回は自分以外に対するエージェントの能力や学習パラメータ) をスコープで削減しても，情報量を制限しない手法 (つまり全情報を保持する手法) より効果的にタスク処理ができることを示した．後者では，互恵性に基づいたエージェントの共同関係を創発させることで，様々なシステム負荷に対応できることを示した．

以上のことから本研究は，(大規模) 分散環境において効率的かつ効果的なチーム編成・タスク割当手法を実現した．次項では本研究の課題を述べる．

7.2 今後の課題

以下に本研究の課題について述べる．

組織の評価と再構築

本論文では，エージェントチームをボトムアップに構築する観点で議論を進めてきた．一方で編成したチームがシステムにとって真に必要であるかを評価する必要がある．なぜならチームがシステムに実際には貢献していないことも考えられる (チームのエージェント全員が結託して仕事を放棄している可能性もある)．そのため，必要に応じてチームを解散し，組み替えることが必要であると考える．

個々のエージェント評価と交換

チームの評価だけはなく，個々のエージェントの貢献度合いも合わせて評価する必要があると考える．本研究では，エージェントの数，能力をあらかじめ設定しており，環境の変化をシステム負荷の変化として実験したが，エージェントの環境も故障などを理由に変化する．そのため，システムとして不要なエージェント (例えば故障したエージェント) を迅速に発見し，新しい高性能なエージェントに入れ替えることで，効率的なタスク処理ができると考える．また誰からも相手にされない「孤立したエージェント」が発生する可能性もある．このエージェントは，リーダとして自分はタスクを取りに行くが，誰に依頼してもメッセージが受託されずチームを編成することができない，あるいは，メンバとして自分はタスクの依頼を待っているが，誰からも依頼が来ない，といったエージェントが当てはまる．これらのエージェントの存在は有効に働く場合も考えられる (例えば第 6 章の実験 3 のように，人手が必要になったときのバックアップとして動く場合がある)．しかし殆どの場合，不要なエージェントと判断でき，より性能の高いエージェントにアップデートしたほうが効率よくタスク処理ができると可能性がある．ただし，上記の理由により不要なエージェントが環境変化によって必要なエージェントになる可能性もあるため，環境変化とシステム状況から貢献度を評価する仕組みが必要であると考える．

エージェント間距離の実装とその評価

モデルの拡張として，エージェント間の距離が考えられる．今回，エージェントは空間に配置されているが，そのエージェントの間に距離は考慮していない．そのためメッセージの送受信にかかる通信時間や遅延は考慮していない．実際の環境では，エージェントは様々な場所に配置されており，エージェント間には距離が存在する．また，それに比例し，エージェントのやり取りするために通信時間が発生する．これらをモデルに追加し，その評価をする必要があると考える．

タスクスケジューリング

最後にタスクスケジューリングも考慮すべきだと考える．今回はエージェントにサブタスクを割当てる際，そのエージェントに対する信頼性と処理の可否だけで判断してタスクを割当ていた．しかし，実際はタスクに優先順位 (例えば緊急を要するタスク，デッドライン) が存在する場合も考えられ，どのような順番でエージェントに割当てるかは重要である．優先度を考慮することでより効率的なシステムモデルの提案ができると考える．

謝辞

本博士論文を作成するにあたり，多くの方々のご支援，ご協力を賜りました．この場を借りて謹んで御礼申し上げます．誠にありがとうございました．

菅原俊治教授には，言葉に尽くしきれない感謝の意を表したいと思います．研究室に配属された学部 4 年生のときからこれまで，研究だけでなく就職関係，その他多方面に関して多くの助言をいただきました．菅原俊治教授のサポートのおかげで，現在の自分があると言っても過言ではありません．本当に，本当にありがとうございました．

また，お忙しい中審査していただいた大須賀昭彦教授，酒井哲也教授，菱山玲子教授にも感謝しております．諸事情により審査が延期になった際も再度引き受けていただき，ありがとうございました．

菅原研究室の皆様にも本当にお世話になりました．研究指導をして頂いた先輩方，相談に乗ってくれた同期の皆，議論をした後輩，深く感謝しています．ありがとうございました．特に，杉山歩未君には，研究議論から，助手の仕事，私生活にわたり様々な面で相談にのっていただき，本当に感謝しています．ありがとう．

これまで何があってもいつもあたたかい励ましを送ってサポートしてくれた家族にも本当に感謝しています．ありがとうございました．

最後にこれまで支えてくれたすべての方々に感謝の意を表します．

ありがとうございました．

参考文献

[1] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications. Vol. 17, p. Fourthquarter 2015, November 2015.

[2] J. A. Stankovic. Research Directions for the Internet of Things. *IEEE Internet of Things Journal*, Vol. 1, No. 1, pp. 3–9, February 2014.

[3] 日経 BP 社出版局. クラウド大全 第 2 版. 日経 BP 社, 第 2, April 2010.

[4] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D. Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patterson, Ariel Rabkin, Ion Stoica, and Matei Zaharia. A View of Cloud Computing. *Commun. ACM*, Vol. 53, No. 4, pp. 50–58, April 2010.

[5] Ragunathan (Raj) Rajkumar, Insup Lee, Lui Sha, and John Stankovic. Cyber-physical Systems: The Next Computing Revolution. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pp. 731–736, New York, NY, USA, 2010. ACM.

[6] 岩野和生, 高島洋典. サイバーフィジカルシステムと IoT(モノのインターネット実世界と情報を結びつける). 情報管理, Vol. 57, No. 11, pp. 826–834, 2015.

[7] H. Farhangi. The Path of the Smart Grid. *IEEE Power and Energy Magazine*, Vol. 8, No. 1, pp. 18–28, January 2010.

[8] X. Fang, S. Misra, G. Xue, and D. Yang. Smart Grid;The New and Improved Power Grid:A Survey. *IEEE Communications Surveys Tutorials*, Vol. 14, No. 4, pp. 944–980, Fourth 2012.

[9] M. N. Huhns, M. P. Singh, and Burstein. Research Directions for Service-Oriented Multiagent Systems. *Internet Computing, IEEE*, Vol. 9, No. 6, pp. 65–70, 2005.

[10] Sebastian Stein, Enrico Gerding, and Nicholas R. Jennings. Optimal Task Migration in Service-Oriented Systems: Algorithms and Mechanisms. In *Proceedings of the 19th European Conference on Artificial Intelligence*, pp. 73–78, Amsterdam, The Netherlands, The Netherlands, 2010. IOS Press.

[11] B. Dunin-Keplicz and R. Verbrugge. *Teamwork in Multi-Agent Systems: A Formal Approach*. Wiley Series in Agent Technology. Wiley, 2011.

[12] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, Vol. 101, No. 1, pp. 165 – 200, 1998.

[13] Somchaya Liemhetcharat and Manuela Veloso. Modeling and Learning Synergy for Team Formation with Heterogeneous Agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pp. 365–374, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[14] Marc Pujol-Gonzalez, Jesus Cerquides, Alessandro Farinelli, Pedro Meseguer, and Juan Antonio Rodriguez-Aguilar. Efficient Inter-Team Task Allocation in RoboCup Rescue. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pp. 413–421, Richland, SC, 2015. International Foundation for Autonomous Agents and Multiagent Systems.

[15] Sarvapali D. Ramchurn, Alessandro Farinelli, Kathryn S. Macarthur, and Nicholas R. Jennings. Decentralized Coordination in RoboCup Rescue. *Comput. J.*, Vol. 53, No. 9, pp. 1447–1461, November 2010.

[16] 山田誠二. 適応エージェント (認知科学モノグラフ 8). 共立出版, November 1997.

[17] 石田亨. エージェントを考える (特集「エージェントの基礎と応用」). 人工知能学会誌 (Journal of Japanese Society for Artificial Intelligence), Vol. 10, No. 5, pp. 663–667, September 1995.

[18] S.J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Series in Artificial Intelligence. Prentice Hall/Pearson Education, 2003.

[19] Shinichi Nakasuka, Takehisa Yairi, and Hiroyuki Wajima. Autonomous Generation of Reflexion-based Robot Controller Using Inductive Learning. *Robotics and Autonomous Systems*, Vol. 17, No. 4, pp. 287 – 305, 1996.

[20] 木村元, 宮崎和光, 小林重信. 強化学習システムの設計指針. 計測自動制御学会, 1999.

[21] Christopher J.C.H. Watkins and Peter Dayan. Technical Note: Q-Learning. *Machine Learning*, Vol. 8, No. 3, pp. 279–292, May 1992.

[22] 伊庭斉志. 遺伝的アルゴリズムの基礎 - GA の謎を解く. オーム社, September 1994.

[23] 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, March 2007.

[24] Sebastian Raschka. Python 機械学習プログラミング達人データサイエンティストによる理論と実践 (impress top gear). インプレス, June 2016.

[25] 熊沢逸夫. 学習とニューラルネットワーク (電子情報通信工学シリーズ). 森北出版, July 1998.

[26] マルチエージェント学習:相互作用の謎に迫る. コロナ社, 2003.

[27] Michael Woolridge and Michael J. Wooldridge. *Introduction to Multiagent Systems*. John Wiley Sons, Inc., New York, NY, USA, 2001.

[28] Alan H. Bond. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann Pub, August 1988.

[29] 電気学会 GA ニューロを用いた学習法とその応用調査専門委員会. 学習とそのアルゴリズム:ニューラルネットワーク・遺伝アルゴリズム・強化学習. 森北出版, 2002.

[30] 伊藤正美. 自律分散システム研究の課題と将来. 計測と制御, Vol. 32, No. 10, pp. 789–796, 1993.

[31] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson. M2M: From Mobile to Embedded Internet. *IEEE Communications Magazine*, Vol. 49, No. 4, pp. 36–43, April 2011.

[32] 桑原和宏, 石田, 亨. 分散人工知能 (2) : 交渉と均衡化. 人工知能学会誌 Journal of Japanese Society for Artificial Intelligence, Vol. 8, No. 1, pp. 17–25, January 1993.

[33] 大沢 英一. マルチエージェント環境における交渉のモデル (特集「エージェントの基礎と応用」). 人工知能学会誌 Journal of Japanese Society for Artificial Intelligence, Vol. 10, No. 5, pp. 690–696, September 1995.

[34] Reid Garfield Smith. *A Framework for Problem-solving in a Distributed Processing Environment.* PhD thesis, Stanford, CA, USA, 1979. AAI7912412.

[35] L. D. ERMAN. Hearsay-II 音声認識システム : 不確定性の解決のための知識の統合. コンピュータサイエンス, pp. 53–91, 1981.

[36] L.D. Erman, F. Hayes-Roth, V.R. Lesser, and D.R. Reddy. The HEARSAY-II Speech Understanding System: Integrating Knowledge to Resolve Uncertainty. *Computing Surveys*, Vol. 12, No. 2, pp. 213–253, June 1980.

[37] 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.

[38] Tuomas Sandholm. An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI'93, pp. 256–262. AAAI Press, 1993.

[39] Toshiharu Sugawara, Toshio Hirotsu, Satoshi Kurihara, and Kensuke Fukuda. Performance Variation due to Interference among a Large Number of Self-interested Sgents. In *IEEE Congress on Evolutionary Computation*, pp. 766–773. IEEE, 2007.

[40] Randall Davis and Reid G. Smith. *Negotiation as a Metaphor for Distributed Problem Solving*, pp. 51–97. Springer Berlin Heidelberg, Berlin, Heidelberg, 2003.

[41] Jonathan Billington, AmarKumar Gupta, and GuyEdward Gallasch. Modelling and Analysing the Contract Net Protocol - Extension Using Coloured Petri Nets. In *Formal Techniques for Networked and Distributed Systems–FORTE 2008*, Vol. 5048 of *Lecture Notes in Computer Science*, pp. 169–184. Springer Berlin Heidelberg, 2008.

[42] Gaojun Fan, Hongbing Huang, and Shiyao Jin. An Extended Contract Net Protocol Based on the Personal Assistant. *Computing, Communication, Control and Management, ISECS International Colloquium on*, Vol. 2, pp. 603–607, 2008.

[43] Cheng Gu and Toru Ishida. Analyzing the social behavior of Contract Net Protocol. In Walter Van de Velde and John W. Perram, editors, *Agents Breaking Away*, pp. 116–127, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[44] Michael Schillo, Christian Kray, and Klaus Fischer. The Eager Bidder Problem: A Fundamental Problem of DAI and Selected Solutions. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2*, AAMAS '02, pp. 599–606, New York, NY, USA, 2002. ACM.

[45] Toshihiro Matsui and Hiroshi Matsuo. A constraint based formalisation for distributed cooperative sensor resource allocation. *International Journal of Intelligent Information and Database Systems*, No. 4, pp. 307–321.

[46] 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. IEEE Computer Society.

[47] 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. IEEE Computer Society.

[48] Kazuki Urakawa and Toshiharu Sugawara. Reorganization of Agent Networks with Reinforcement Learning Based on Communication Delay. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference on*, Vol. 2, pp. 324–331, 2012.

[49] Liat Sless, Noam Hazon, Sarit Kraus, and Michael Wooldridge. Forming Coalitions and Facilitating Relationships for Completing Tasks in Social Networks. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent*

Systems, AAMAS '14, pp. 261–268, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[50] Onn Shehory and Sarit Kraus. Methods for Task Allocation via Agent Coalition Formation. *Artif. Intell.*, Vol. 101, No. 1-2, pp. 165–200, May 1998.

[51] Onn Shehory and Sarit Kraus. Feasible Formation of Coalitions Among Autonomous Agents in Non-Super-Additive Environments. *Computational Intelligence*, Vol. 15, No. 3, 1999.

[52] Carlos Merida-Campos and Steven Willmott. Modelling Coalition Formation over Time for Iterative Coalition Games. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems - Volume 2*, AAMAS '04, pp. 572–579, Washington, DC, USA, 2004. IEEE Computer Society.

[53] Thomas Genin and Samir Aknine. Coalition Formation Strategies for Self-Interested Agents in Task Oriented Domains. In *International Conference on Intelligent Agent Technology*, pp. 205–212, November 2010.

[54] Dai Hamada and Toshiharu Sugawara. Autonomous Decision on Team Roles for Efficient Team Formation by Parameter Learning and its Evaluation. *Intelligent Decision Technologies*, Vol. 7, No. 3, pp. 163–174, 2013.

[55] Somchaya Liemhetcharat and Manuela Veloso. Modeling and Learning Synergy for Team Formation with Heterogeneous Agents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 1*, AAMAS '12, pp. 365–374, Richland, SC, 2012. International Foundation for Autonomous Agents and Multiagent Systems.

[56] Christian Guttmann. Making Allocations Collectively: Iterative Group Decision Making under Uncertainty. In Ralph Bergmann, Gabriela Lindemann, Stefan Kirn, and Michal Pechoucek, editors, *MATES*, Vol. 5244 of *Lecture Notes in Computer Science*, pp. 73–85. Springer, 2008.

[57] Makoto Yokoo and Katsutoshi Hirayama. Algorithms for Distributed Constraint Satisfaction: A Review. *Autonomous Agents and Multi-Agent Systems*, Vol. 3, No. 2, pp. 185–207, June 2000.

[58] Alan K. Mackworth. The Logic of Constraint Satisfaction. *Artificial Intelligence*, Vol. 58, No. 1, pp. 3 – 20, 1992.

[59] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce, and P. Varakantham. Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-event Scheduling. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004.*, pp. 310–317, July 2004.

[60] J. Von Neumann and O. Morgenstern. *Theory of Games and Economic Behavior*. Princeton University Press, 1947.

[61] S. Hargreaves-Heap and Y. Varoufakis. *Game Theory: A Critical Introduction*. Taylor & Francis, 2004.

[62] R Axelrod and WD Hamilton. The Evolution of Cooperation. *Science*, Vol. 211, No. 4489, pp. 1390–1396, 1981.

[63] Werner Gth, Rolf Schmittberger, and Bernd Schwarze. An Experimental Analysis of Ultimatum Bargaining. *Journal of Economic Behavior & Organization*, Vol. 3, No. 4, pp. 367 – 388, 1982.

[64] Yuki Miyashita, Masashi Hayano, and Toshiharu Sugawara. *Formation of Association Structures Based on Reciprocity and Their Performance in Allocation Problems*, pp. 262–281. Springer International Publishing, Cham, 2016.

[65] I. Stojmenovic and S. Wen. The Fog Computing Paradigm: Scenarios and Security Issues. In *2014 Federated Conference on Computer Science and Information Systems*, pp. 1–8, September 2014.

[66] Toshiharu Sugawara Masashi Hayano, Dai Hamano. Role and Member Selection in Team Formation Using Resource Estimation. In *Proceedings of the 7th International KES Conference on Agents and Multi-agent Systems - Technologies and Applications (KES-AMSTA 2013) (published as Frontiers in Artificial Intelligence and Applications*, Vol. 252, pp. 125 – 136, 2013.

[67] Masashi Hayano, Dai Hamada, and Toshiharu Sugawara. Role and Member Selection in Team Formation Using Resource Estimation for Large-scale Multi-agent Systems.

Neurocomputing, Vol. 146, No. Supplement C, pp. 164 – 172, 2014. Bridging Machine learning and Evolutionary Computation (BMLEC) Computational Collective Intelligence.

[68] Masashi Hayano, Yuki Miyashita, and Toshiharu Sugawara. Switching Behavioral Strategies for Effective Team Formation by Autonomous Agent Organization. In *ICAART 2016 - Proceedings of the 8th International Conference on Agents and Artificial Intelligence*, Vol. 1, pp. 56–65. SciTePress, 2016.

[69] Masashi Hayano, Yuki Miyashita, and Toshiharu Sugawara. Adaptive Switching Behavioral Strategies for Effective Team Formation in Changing Environments. pp. 37–55, Cham, 2017. Springer International Publishing.

[70] 早野真史, 宮下裕貴, 菅原俊治. エージェントの行動戦略選択による組織化とタスク処理効率化の実現. 人工知能学会論文誌, Vol. 31, No. 6, pp. AG F 1–11, 2016.

[71] Naoki Iijima Masashi Hayano and Toshiharu Sugawara. Asynchronous Agent Teams for Collaborative Tasks Based on Bottom-Up Alliance Formation and Adaptive Behavioral Strategies. In *Proceedings of The 15th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2017)*, Vol. 252, pp. 589–596, 2013.

[72] Robert L. Trivers. The Evolution of Reciprocal Altruism. *The Quarterly Review of Biology*, Vol. 46, No. 1, p. 35, 1971.

[73] 川越敏司. 行動ゲーム理論入門. エヌティティ出版, March 2010.

[74] Hisashi Ohtsuki and Yoh Iwasa. The Leading Eight: Social Norms that can Maintain Cooperation by Indirect Reciprocity. *Journal of Theoretical Biology*, Vol. 239, No. 4, pp. 435 – 444, 2006.

[75] Ernst Fehr and Urs Fischbacher. Why Social Preferences Matter – The Impact of Non-Selfish Motives on Competition, Cooperation and Incentives. *Economic Journal*, Vol. 112, No. 478, pp. C1–C33, 2002.

[76] Marco Dorigo and Gianni Di Caro. New Ideas in Optimization. chapter The Ant Colony Optimization Meta-heuristic, pp. 11–32. McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999.

[77] Cytoscape. http://www.cytoscape.org.

[78] Daniel D. Corkill, Daniel Garant, and Victor R. Lesser. Exploring the Effectiveness of Agent Organizations. In Virginia Dignum, Pablo Noriega, Murat Sensoy, and Jaime Simão Sichman, editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems XI: COIN 2015 International Workshops, COIN@AAMAS, Istanbul, Turkey, May 4, 2015, COIN@IJCAI, Buenos Aires, Argentina, July 26, 2015, Revised Selected Papers*, pp. 78–97, Cham, 2016. Springer International Publishing.

業績リスト

1. 論文誌

   - ○早野真史, 宮下裕貴, 菅原俊治,“エージェントの行動戦略選択による組織化とタスク処理効率化の実現”, 人工知能学会誌 人工知能学会論文誌, Vol. 31, No. 6, p.AG-F 1-11, 2016. DOI: 10.1527/tjsai.AG-F
   - ○ Masashi Hayano, Dai Hamada and Toshiharu Sugawara,“Role and Member Selection in Team Formation Using Resource Estimation for Large-Scale Multi-Agent Systems” Neurocomputing, Volume 146, Pages 164-172, Elsevier, ISSN: 0925-2312, Dec. 25 2014. DOI: 10.1016/j.neucom.2014.04.059

2. 国際会議 (査読付き)

   - ○ Masashi Hayano, Naoki Iijima and Toshiharu Sugawara “Asynchronous Agent Teams for Collaborative Tasks Based on Bottom-Up Alliance Formation and Adaptive Behavioral Strategies,” Proccedings of The 15th IEEE International Conference on Dependable, Autonomic and Secure Computing (DASC 2017), Orlando, Florida, USA, November 6-10, 2017.
   - ○ Masashi Hayano, Yuki Miyashita and Toshiharu Sugawara, “Adaptive Switching Behavioral Strategies for Effective Team Formation in Changing Environments,” Agents and Artificial Intelligence (Revised and selected papers from ICAART 2016), LNAI (LNCS) 10162, Springer, pp. 37-55, 2017. DOI: 10.1007/978-3-319-53354-4 3
   - ○ Masashi Hayano, Yuki Miyashita and Toshiharu Sugawara, “Switching Behavioral Strategies for Effective Team Formation by Autonomous Agent Organization,” Proceedings of the 8th International Conference on Agents and Artificial Intelligence, pp. 56-65, Feb. 24-26, 2016. DOI: 10.5220/0005748200560065
   - ○ Masashi Hayano, Dai Hamada and Toshiharu Sugawara, “Role and Member Selection in Team Formation Using Resource Estimation,” Proceedings of the 7th

International KES Conference on Agents and Multi-agent Systems - Technologies and Applications (KES-AMSTA 2013) (published as Frontiers in Artificial Intelligence and Applications, Vol. 252), pp. 125-136, IOS Press, May 27-29, 2013. DOI: 10.3233/978-1-61499-254-7-125

- Naoki Iijima, Ayumi Sugiyama, Masashi Hayano and Toshiharu Sugawara, "Adaptive Task Allocation Based on Social Utility and Individual Preference in Distributed Environments," Proceedings of 21st International Conference on Knowledge-Based and Intelligent Information Engineering Systems (KES 2017), Procedia Computer Science, Vol. 112, pp. 91-98, Elsevier, Sep. 6-8, 2017.DOI: 10.1016/j.procs.2017.08.177
- Naoki Iijima, Masashi Hayano, Ayumi Sugiyama and Toshiharu Sugawara, "Analysis of Task Allocation Based on Social Utility and Incompatible Individual Preference," Proceedings of the 21st Conference on Technologies and Applications of Artificial Intelligence (TAAI 2016), pp. 24-31, IEEE Xplore, Nov. 2016. DOI: 10.1109/TAAI.2016.7880161
- Yuki Miyashita, Masashi Hayano and Toshiharu Sugawara, "Formation of Association Structures Based on Reciprocity and Their Performance in Allocation Problems," Coordination, Organizations, Institutions, and Norms in Agent Systems XI (COIN book 2015), LNAI Vol. 9628, pp. 262-281, Springer, 2016. DOI: 10.1007/978-3-319-42691-4 15
- Ryutaro Kawaguchi, Masashi Hayano, and Toshiharu Sugawara, "Balanced Team Formation for Tasks with Deadlines," Proceedings of 2015 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT 15), pp. 234-241, IEEE Computer Society Press, Dec. 6-9, 2015. DOI: 10.1109/WI-IAT.2015.57
- Yuki Miyashita, Masashi Hayano and Toshiharu Sugawara, "Self-Organizational Reciprocal Agents for Conflict Avoidance in Allocation Problems," Proceedings of 9th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2015), pp 151-155, IEEE Xplore, September 21-15, 2015. DOI: 10.1109/SASO.2015.24
- Yuki Miyashita, Masashi Hayano and Toshiharu Sugawara, "Association Formation Based on Reciprocity for Conflict Avoidance in Allocation Problems," Pro-

ceedings of Workshop on Coordination, Organizations, Institutions and Norms in Agent Systems (COIN 2015), (held in conjunction with the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS2015)), pp. 143-157, May 4-8, 2015. IFAAMAS

3. 国内会議 (査読付き)

- 早野真史, 宮下裕貴, 菅原俊治, “行動戦略選択エージェントによる協同関係強化手法の提案,” エージェント合同シンポジウム (JAWS2015) 予稿集, 日本ソフトウエア科 学会，電子情報通信学会，人工知能学会， 情報処理学会共催, Sep. 30 - Oct. 2, 2015.
- 杉山歩未, Vourchteang Sea, 早野真史, 菅原俊治, “継続協調巡回問題における分業創発と環境変化への追従性,” エージェント合同シンポジウム 予稿集 (JAWS 2017), 日本ソフトウエア科学会，電子情報通信学会，人工知能学会， 情報処理学会共催, Sep. 15-17, 2017.
- 飯嶋直輝, 杉山歩未, 早野真史, 菅原俊治, “分散環境における希望順位戦略の学習機能を備えたタスク割り当て手法の提案と評価,” エージェント合同シンポジウム 予稿集 (JAWS 2017), 日本ソフトウエア科学会，電子情報通信学会，人工知能学会，情報処理学会共催, Sep. 15-17, 2017.
- 宮下裕貴, 早野真史, 菅原俊治, “チーム編成ゲームと互恵エージェントを用いた自律的組織化について,” 第 11 回ネットワークが創発する知能研究会ワークショップ (JWEIN2015) 論文集, 日本ソフトウエア科学会, 日本大学, August 19-21, 2015.

4. 国内会議 (査読なし)

- 早野真史, 菅原俊治, “非同期チーム編成における互恵編成戦略の提案と評価,” 第 79 回情報処理全国大会予稿集, 7F-05, Mar. 16-18, 2017.
- 早野真史, 宮下裕貴, 菅原俊治, “共同関係の強化による効率的なチーム編成手法の実現,” 情報処理学会全国大会, 4T-04, Mar. 17-19, 2015.
- 早野 真史, 菅原 俊治, “大規模な環境におけるリソースの同定学習とチーム編成の効率化について,” 人工知能と知識処理研究会技術研究報告, 信学技報, AI2013-20, pp. 7-12, 電子情報通信学会, Nov. 28-29, 2013
- 早野 真史, 浜田 大, 菅原俊治, “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,” 知能システム研究会予稿集 ICS, Vol.2013-ICS-170, No. 11, 情報処理学会 社会システムと情報技術研究ウィーク, Mar. 10-13, 2013.

- 尾形直哉, 早野真史, 菅原俊治, “値引きを考慮した小売店の発注戦略の分析,” 知能システム研究会予稿集 ICS, Vol. 2017-ICS-186, No. 5, pp. 1-7, 情報処理学会, 社会システムと情報技術研究ウィーク, Mar. 2-5, 2017.
- 飯嶋 直輝, 齋藤 健吾, 早野 真史, 菅原 俊治, “継続的に発生する優先度つきタスクの効率的割り当て手法の一解法について,” 知能システム研究会予稿集 ICS, Vol. 2016-ICS-182, No. 9, pp. 1-7, 情報処理学会 社会システムと情報技術研究ウィー ク, Mar. 1-3, 2016.
- 川口 竜太郎, 早野 真史, 菅原 俊治, “デッドライン付きタスクを効果的に割り当てるチーム編成手法の提案,” 人工知能と知識処理研究会技術研究報告, Vol. 115, No. 381, AI2015-47, pp. 125 - 130, 電子情報通信学会, Dec. 18-19, 2015.
- 川口 竜太郎, 早野 真史, 菅原 俊治, “デッドライン付きタスクを対象とした効率的チーム編成手法の提案,” 知能システム研究会予稿集 ICS, Vol. 2015-ICS-178, No. 4, 情報処理学会 社会システムと情報技術研究ウィーク, Mar. 1-4, 2015.