マルチエージェント学習システムにおける動的な役割分担†

*井上寛康(1)(2) 高玉圭樹(1)(3) 下原勝憲(1)(2) 片井修(2)
(1) ATR 人間情報科学研究所 京都府「けいはんな学研都市」光台 2-2-2
(2) 京都大学大学院情報学研究科 京都府左京区吉田本町
(3) 東京工業大学大学院総合理工学研究科 神奈川県横浜市緑区長津田町 4259

Abstract: エージェント個々が異なる役割を担っているマルチエージェントシステムにおいて，あるエージェントをそのシステムから除去する必要がある場合，単に除去すると一般的に役割の協調のバランスが崩れ全体のパフォーマンス低下を招く．これを防ぐために，本研究では安全なエージェント除去アルゴリズムを次の 3 つの方法で実現する．それらは，(1) エージェント間の役割の違いをお互いの持つルールセットからエージェントが自律的に計算する方法，(2) いつ，どのエージェント間で役割の違いを認識するかの方法，および，(3) 除去されるエージェントの知識を除去されないいずれかのエージェントに上書きするか判断する方法である．上記 3 つの方法を試作した結果，あるエージェントを除去するリスクが他のエージェントよりも確実に高い場合，パフォーマンスの低下が抑えられることがわかった．

1 序論

シングルエージェントが達成できない問題へのアプローチとしてマルチエージェントによる解決が模索されている [3]．このようなマルチエージェントが環境内で個々に異なる役割を担う場合，その協調のバランスが問題解決のパフォーマンスに大きく影響する．故障や点検，あるいは別のシステムへ移すなどの理由で，あるエージェントをそのシステムから除去する際，役割協調のバランスが崩れて全体のパフォーマンス低下につながる．このようなパフォーマンス低下を最小限にとどめるにはどうすればよいのであろうか．

ここで前提としているエージェントはその行動決定にルールベースのシステムを用いるとし，備えている能力の仕様はエージェント間で同じとする．すなわち，このようなエージェントが異なる役割を持つということは，ルールベースシステムの持つルールが異なることを意味する．このようなマルチエージェントの故障を扱った従来研究として，故障したエージェントを発見し取り除くことによりパフォーマンスの低下を防ぐというものがある [1]．この研究においては各エージェントのパフォーマンスを観察し，故障を判断して除去している．これに対し本研究では，故障等のエージェント除去のきっかけは与えられたと仮定する．加えて，エージェント除去のきっかけは突如与えられるとし，除去に対する計画をエージェントは持てないとする．このような場合において，役割協調のバランスを崩すことなくエージェントを除去する方法を提案する．

以上を踏まえ本研究では次の 3 つの方法が必要と考える．1 つ目はエージェント間の役割の違いをルールセットから判定する方法である．役割は明示的に与えられているわけではなくルールを調べなければならないためである．逆に担当する役割が適応的に変化するシステムにおいては明示的な役割を持たないはずである．2 つ目はいつ，どのエージェント間で役割の違いを認識するかの方法である．これが必要であるのは，すべてのエージェント間の役割の違いを常時認識するのは現実的ではないためである．3 つ目は除去されるエージェントの知識をいずれかの残されるエージェントに引き継がせるか判断する方法である．すなわち，あるエージェントが取り除かれるとき，エージェント間の役割の違いの認識をもとにそのエージェントをその知識とともに除去するか，そのエージェントの知識をいずれかの除去されないエージェントの知識に

†本研究は通信・放送機構の研究委託「人間情報コミュニケーションの研究開発」により実施したものである．

上書きするか判断する方法ということである．本稿では，上記 3 つの方法を用いてエージェント除去アルゴリズムを試作した結果と検証について述べる．

2 エージェント除去アルゴリズム

1 章で述べたアルゴリズム実現の 3 つの方法を整理すると，(1) エージェント間の役割の違い認識方法，(2) いつ，どのエージェント間で役割の違いを認識するかの方法，(3) 除去（あるいは上書き）される知識選択方法の 3 つである．本稿でのこれらの実現方法についてそれぞれ述べる．

1. **エージェント間の役割の違い認識方法**：ルールは条件部と行動部よりなる．(条件は互いに排他的とする．すなわち，条件間には包含や共通などは存在しない.) 加えて各ルールはそのエージェントが使用した頻度を持っているとする．(ただしエージェントの行動すべてについて記録するのは現実的ではない．したがって過去何ステップかの記録から頻度を求めるとする.) そのような 2 つのルールセット A と B がある際，以下のような方法でルールセットの違い (D) を求め，役割の違いとする．要約すると同じ状況において違う行動をとるかどうかを求めていることになる．
   - D に 0 を初期設定.
   - A の各ルールについて，そのルールの条件部が同じ B のルールを探す.
     - ルールが見つかって，行動部が同じ ： 何もしない.
     - ルールが見つかって，行動部が違う ： 該当する A と B のルールの頻度の積を D にたす.
     - ルールが見つからなかった ： A のルールの頻度を D にたす.

   相手のエージェントとの役割の違いはお互いのエージェントが記録して持っておく．
2. **いつ，どのエージェント間で役割の違いを認識するかの方法**：各エージェントはエージェント除去イベントの直前にエージェント間の役割の違いを認識しているとする．また，すべてのエージェントは他のすべてのエージェントとの役割の違いがわかっているとする．すなわち，この方法において本稿では特別な方法を提案しない．(ただし，ここには様々な方法が存在し，調査の余地がある.)
3. **除去される知識選択方法**：除去されると決まったエージェント（除去エージェント）の実体（知識以外）は除去されるが，そのエージェントが持っている知識を一緒に除去するときのリスクがわかれば，これを残すかどうか判断できる．知識を残すということは他の残されたエージェントに知識を上書きすることを示す．知識を上書きされうるこのエージェントをここでは候補エージェントと呼ぶ．(「候補エージェント」をどのように選ぶかも方法が必要であるが，ここではすべての他のエージェントとする.) 候補エージェントとの役割の違いを除去エージェントは持っているのでこれよりリスクを求める．すなわち候補エージェントとの役割の違いのうち，最小のものをリスクの基本値とする．これは，自分と最も近い役割のエージェントの D の値がいくらか，ということであり，この値が高いということは候補エージェントと異なる役割を担っている可能性が高いといえる．言い換えれば，除去エージェントの知識が除去される際の危険度といえる．除去される知識選択のためにまず，候補エージェントの知識が除去されるとした場合のリスクの基本値も，除去エージェントと同じように計算する．すなわち 1 つの候補エージェントが，残りの候補エージェントおよび除去エージェントとの間の役割の違いを使ってリスクの基本値を求める．次にそれらリスクの基本値を比較して，最も小さいリスクの基本値を持つエージェントを選び，その知識を取り除く．具体的には，もしも除去エージェントの知識が選ばれた場合は，除去エージェントをその知識とともに除去する．候補エージェントの知識が選ばれた場合は，除去エージェントの知識を選ばれた候補エージェントの知識に上書きする．すなわち，その候補エージェントの知識は除去される．今後，リスクというときは除去エージェントと候補エージェントのリスクの基本値を足して，その値で各基本値を割ったものを指すとする．(比例配分されている.)

3　実験と結果

実験は 1 次元の離散的環境で行った．3 体のエージェントは最低 1 体ずつ線分の両端に到達するのを目的とする．このとき，この線分の端のどちらに行くかは明示的に与えられていない．したがってどちらか一方に行くことを専門に行うエージェント，状況を見てどちらに行くか決めるエージェントなどさまざまな役割分担の可能性がある．もちろん，この実験環境は実問題のメタファである．

役割分担のさまざまな組み合わせをテストするため，エージェントは分類子学習システムの 1 つである XCS[2] により，ルールを持たないところから学習する．また GA メカニズムとルールに含まれる # は除いてある．環境において，3 体のエージェントのうち最低 1 体のエージェントがそれぞれの端に到達した際に報酬が得られ，スタート地点である線分の中心に戻される．(スタートしてからこの状態になるまでを 1 試行と呼ぶ，エージェントが 1 回離散的に知覚・移動するのを 1 ステップと呼ぶ.）いったん端に到達したエージェントは移動しない．また，すべてのエージェントが片方の端に到達した場合は報酬はなく線分の中心に戻される．

実験では 3000 試行の間学習したのち，ランダムに除去エージェントが選ばれるとした．除去エージェントのリスクと実際に除去エージェントの知識が除去された場合のパフォーマンスの低下が図 1 である．横軸があるエージェントが除去される際のリスク，縦軸が実際にそのエージェントがそのまま除去されたときのパフォーマンスの低下度合いである．具体的には残った 2 体のエージェントで線分の両端に到達するまでにかかったステップ数である．各プロットはエージェントが除去された後,（残り 2 体のエージェントの）300 試行の間のステップ数の平均である．また，この間学習は止められており，1000 ステップを越えるときはそこで打ち切る．図 1 を見ると，リスクが 1/3 を越えたあたりから低下することが多いことがわかる．また 1/2 を越えたあたりからほぼ確実に低下している．これはエージェントが 3 体いるためにリスクの等分である 1/3 でどのエージェントを除去するのが危険かはあいまいになるためであり，また 1/2 を越えるということは他のエージェントより確実にリスクがあるためである．

次にパフォーマンスの低下を防ぐため本稿で提案したエージェント除去アルゴリズムを適用した．その結果が図 2 である．リスクが 1/2 を越えた場合にはアルゴリズムが（リスクが最小の）候補エージェントの知識の上書きを選択するためパフォーマンスが低下しない．これは 1/2 を越えるリスクの場合にほぼパフォーマンスの低下がないことから見てとれる．ただし，1/3 から 1/2 の間はリスクが最小のエージェントの知識を除去したとしても，エージェント間の役割分担が曖昧であることから，アルゴリズムの効果は出ていない．

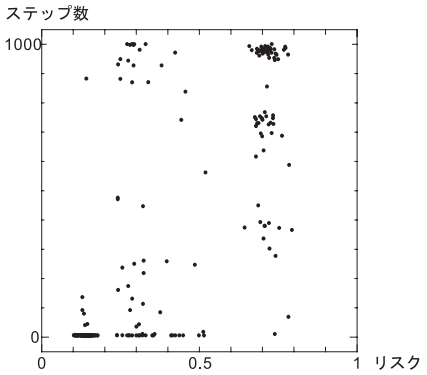

図 1: リスクとパフォーマンス低下の関係

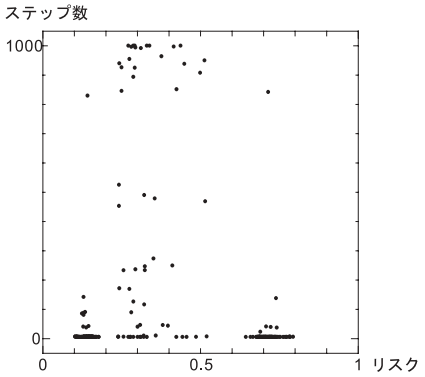

図 2: アルゴリズム適用後の関係

今後の課題として，(1) 実験環境を現実の問題から作成，(2) いつ，どのエージェント間で役割を認識するかの方法の提案，(3) 候補エージェントの選択方法の提案，(4) 図 2 でリスクが 1/3 から 1/2 の場合のパフォーマンス低下が防げていないのでこれを改善，などがある．

参考文献

[1] H. Kasahara, K. Takadama, S. Nakasuka and K. Shimohara. A troubleshooting mechanism based on an organizational learning model for multiple robots. *The 1998 Internatinal Technical Conference on Circuits/Systems, Computer and Communications (ITC-CSCC'98)*, 1023-1026, 1998.

[2] M. V. Butz and S. W. Wilson. An Algorithmic Description of XCS. *Soft Computing*, 6(3-4):144–153, 2002.

[3] G. Weiss. ed. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence* MIT Press, 1999.