マルチメディア通信と分散処理 50－18
（１９９１．５．１７）

分散協調型問題解決における知識表現モデル

柴田知博　長木正宏　村田努　井内稔　山崎晴明
山梨大学

知識の表現モデルの異なる複数の問題解決エージェントにより構成される分散協調型問題解決システムの問題解決方式について報告する。分散型問題解決にみられる、矛盾を含む情報の取扱いおよび知識の重複の度合いについて考察し、矛盾を含んだ状態での推論の継続を試みた。さらにここでは、「ネゴシエーション」「Functionally-accurate Cooperation」と呼ばれる協調方法を組み合わせた協調方法を提案する。また、この方法を実現するために必要な概念として、中間解の信頼度を示す尺度およびエージェント間での情報交換の形式について述べる。さらに、中間解から解を導き出す方法について報告する。

KNOWLEDGE REPRESENTATION MODELS ON COOPERATIVE DISTRIBUTED PROBLEM SOLVING

Tomohiro Shibata　Masahiro Choki　Tsutomu Murata　Minoru Iuchi　Haruaki Yamazaki

Yamanashi University

4-3-11 Takeda, Kofu, Yamanashi 400, Japan

In this paper we describe problem solving system that consists of several problem solvers each of which have its own knowledge representation model. We also considered a treatment of incomplete and contradictory infomation as well as the redundant infomation.
Further, some cooperation schemes like "negotiation" or "Functionally-accurate Cooperation" are discussed.

1．はじめに

知識表現モデルの異なる複数の問題解決エージェントから構成される分散協調型問題解決システムにおける協調の方法について述べる。

従来の問題解決システムはおもに単一の知識表現を持ち、単一の問題解決機構上で動作するシステムが一般的であった。このため、解決できる問題には得意とするものと不得意なものがはっきりと分かれ、解決可能な問題の領域は狭くなり、解の信頼性の面において、必ずしも望ましいとはいえなかった。ところが、現在の解決すべき問題には、問題解決の過程において、多種多様な側面を持つものが多い。従って、このような性質を持つ問題を解決するためには、異なる推論の手段や知識表現を持つ複数の問題解決エージェントが協調して問題を解決するシステムが有効であると考えられる。また、複数の計算機に処理させることによる信頼性の向上、処理の分散による負荷の軽減も期待される。

そこで我々は、協調の方法について考察し、知識表現モデルの異なる複数の問題解決機構を用いて、従来の問題解決システムには見られなかった解の導出法について考察しようと試みた。

2．さまざまな協調の方法

2.1. 矛盾の取扱い

データや知識が空間的、時間的、機能的に分散されているために一つの問題がいくつかの部分問題に分かれるような問題を、単一の問題解決システムによって解決することは、かなり難しいことである。このような問題を分散協調型問題解決システムは得意とする。

しかし、分散協調型問題解決における基本的な問題点は、エージェントが導き出した局所解およびエージェントが取り込む情報に矛盾した情報が含まれている可能性がある、ことである。この矛盾をどう取り扱って効果的に協調を行うことができるかということが、システムの性能を左右する。この矛盾の取扱いに対して、次のような考え方ができる。[1]

第一に、はじめから矛盾の存在を認めないか、矛盾を認識しない。

第二に、エージェント間の話し合いにより、矛盾を解消する。

第三に、矛盾が存在しても、推論を継続して行う。

これらの考えに基づいて現在までに研究された分散型問題解決システムにおける協調の手法にはそれぞれ、「マルチエージェントプランニング」「ネゴシエーション」「Functionally-accurate Cooperation」と呼ばれる方法が挙げられる。

マルチエージェントプランニングとは、問題解決を行う前に、エージェントが矛盾を含まない問題解決の実行計画（マルチエージェントプラン）を作成し、この計画に従って問題解決を行う方法である。

ネゴシエーションとは、エージェント相互が適切な局所解の情報を交換し合い、局所解の共通点を認識した上でエージェント間の話し合いにおいてよりよい解を導き出す方法である。この方法は、人間社会における協調の基本となる方法である。

Functionally-accurate Cooperationは、エージェントが導き出した矛盾した局所解を交換し推論を継続しながら最終的には矛盾を克服する方向に解を収束させていく方法である。

2.2. 知識の重複度

分散協調型問題解決システムを構成するエージェントの持つ知識や資源によって協調の方法も異なる。エージェントの知識表現による分類では、ホモジニアス（知識表現モデルが同じ）なものとヘテロジニアス（知識表現モデルが異なる）なものとに分けられる。これに知識の重複の度合いを考え、分類すると次のようになる。

(1) ホモジニアス・知識の重複あり：

完全に重複した処理が行われる可能性があり、それを避けるための管理機構をおく必要がある。

知識の重複度が少ない場合は、処理が機能分散的になり、並列性は減少するが、並列性を挙げようとすると、知識の重複度を上げなければならない。

しかし、知識の重複度を上げれば処理の重複が増すことになる。さらに、処理の重複を避けるためには管理機構が必要になり、処理の重複度を下げるためにはより強いプロトコルになるため、通信量は増す。

(2) ホモジニアス・知識の重複なし：

各エージェントが持つ知識、視点などが異なるため、自然と処理が分散される。従って、管理機構の必要はない。しかし、全体の解をまとめる処理は、エージェント間で中間結果を交換する必要がある。この場合には、通信量と計算量がトレードオフ要因となる。

(3) ヘテロジニアス・知識の重複あり：

解決する問題の性質や問題解決のフェーズによって、最適なエージェントに問題を割り当てることができる。従って、知識の重複度をどの程度にするかを設計時に考慮しておくことが重要である。

知識表現が異なるため、エージェント間の中間解の交換にはすべてのエージェントが理解できる表現を用意する必要がある。

また、知識の重複度に応じて、処理の重複を避ける管理機構が必要であるが、(1)と比べてあまり強いプロトコルの必要はない。

(4) ヘテロジニアス・知識の重複なし：

各エージェントの持つ知識や価値観の相違から、それぞれ異なる視点で推論を行うため、重複を避ける管理機構をもうける必要はない。

全体の解をまとめる処理では、中間解の交換を行う必要がある。また、(3)と同様に、エージェントが共通に理解することができる表現を用意する必要がある。

3．実現する協調方法

既存の協調方法を考慮にいれ、我々は次のような協調方法を考案した。

まず、矛盾の取扱いについては、さきに挙げたネゴシエーションとFunctionally-accurate Cooperation の両方を取り入れている。また、解に多様性を持たせるため、エージェント間の推論の重複をできる限り避ける。最終的な解を求める段階ではなくても、ネゴシエーションを行うために、中間解の交換を行っている。最終的な解を求める段階では、Functionally-accurate Cooperation に類似した手法により部分解の交換を行い、解に収束させる。

この方法の具体的な内容は５．で述べる。

また、システムを構成するエージェントは、ヘテロジニアスで知識の重複があるもの（2.2.の(3)の構成）とした。

4．分散協調型問題解決システム

今回我々が試作した、分散協調型問題解決システムの構成を、図1に示す。

図のように、複数のエージェントとエージェント相互の通信をサポートするプロトコル・マネー

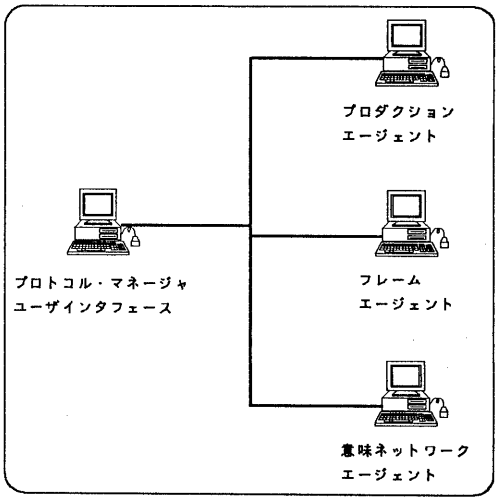

図1 分散協調型問題解決システム構成図

ジャから構成される。システム全体で4台のパーソナル・コンピュータを使用し、内3台をエージェントとして、残りの1台をプロトコル・マネージャとして使用する。これらのパーソナル・コンピュータは、Ｅｔｈｅｒｎｅｔで結ばれており、エージェント間でやりとりする局所解および推論に必要な情報は、Ｅｔｈｅｒｎｅｔを通じて行われる。

エージェントは、「プロダクション・システム」「フレーム・システム」「意味ネットワーク・システム」であり、それぞれ異なる知識表現モデルを用いており、異なる推論機構を持つ。また、協調動作を行うために、通信機能を持つ。なお、各エージェントは完全に重複した知識をそれぞれの知識表現モデルを用いて保持している。

プロダクション・システムは、「もし～ならば…」という「プロダクション・ルール」と呼ばれる知識表現モデルで知識を保持し、プロダクション・ルールを状態に適用して推論を進める。前向き推論方式のプロダクション・システムである。

フレーム・システムは、「スロット」「付加手続き」などから構成される「フレーム」と呼ばれる知識表現モデルで知識を保持し、スロットの参照や、付加手続きによってスロットの値を決定することにより推論を進める。手続き指向型のフレーム・システムである。

意味ネットワーク・システムは、「ノード」「アーク」からなる「意味ネットワーク」と呼ばれる知識表現モデルで知識を保持し、「活性化伝播」を用いてアークをたどることにより推論を進める。

分散協調型問題解決と、エージェント間の通信を集中管理するプロトコル・マネージャが存在することとは、矛盾することである。しかし、本システムはパーソナル・コンピュータ上で実現しているため、ハードウェアの制限があり、完全に分散させることが困難である。従って、エージェント間で相互にやりとりする情報にはいっさい関知しないという条件で、プロトコル・マネージャをもうけることにした。

このプロトコル・マネージャは、あるエージェントから受け取った情報を他のエージェントに渡す機能、およびエージェントから送られてきた情報を保持する機能を持つ（情報の保持の方法については後章で詳しく述べる）。

５．協調動作

5.1. 送信情報の形態

今回我々が試作したシステムにおいて、エージェントが協調のために送信する情報として、４種類の形態がある。

(1) 提案

エージェントが推論を行う際に得た中間結果および結果をプロトコル・マネージャに送信することをいう。このとき、送信する情報にこの情報の信頼度（以下評価値と呼ぶ）を付加する。

(2) 質問

エージェント内で推論を継続することが不可能になり、他のエージェントに対して推論の継続を依頼する際に行う。このとき送信する情報には、評価値は付加せず、推論をできなかった部分を明らかにして送信する。

(3) 応答

「質問」に対する答である。「質問」を受け取ったエージェントが推論結果および評価値を送信する。

(4) 再評価

推論結果が問題に対する答であると判断した場合に、他のエージェントに情報の評価を依頼する場合に送信する。

これらの情報を状況に応じて使い分けることで、情報の意味がより明確になり、エージェントの推論およびエージェント間の協調が、より容易に実現できる。

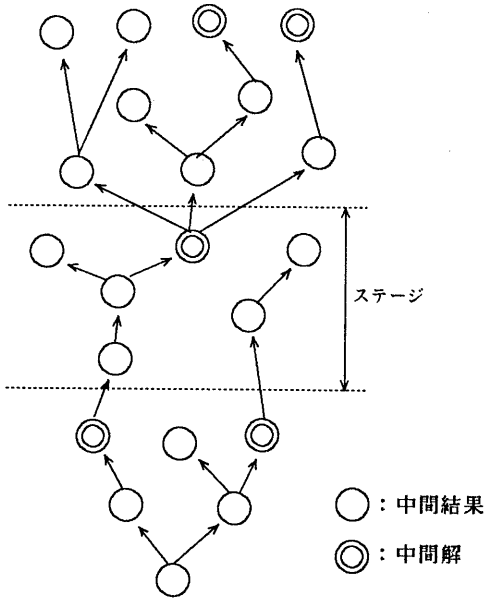

図2 ステージの概念

5.2. マルチ・ステージ

現実に存在する問題を考えると、一回の導出過程で解が求められることは希である。一般に、何段階もの過程を経て解が求められることが多い。この段階のことを「ステージ」と呼ぶことにする。

本システムでは、ステージを何回も繰り返して問題に対する解を求める方法をとる。これは、システムの問題に対する柔軟性を持たせること、および協調動作の効果を高めるためである。

従って、協調して問題を解決するために与えられた問題の分析を行い、問題を部分問題に分割する必要がある。この分割する作業はシステム上で動作するアプリケーション・プログラムに依存するため、あらかじめアプリケーションによって実現されていなければならない。

分割された問題の最初のステージが初期問題となり、推論を開始する。各ステージごとに、次のステージに引き継ぐべき中間解を一つ以上選択する。選択された中間解を次のステージの問題として引き継ぎ、推論を継続して行う。これを繰り返し行った結果、最終ステージで選択された解が与えられた問題の解となる。中間解と解の選択方法については7.2.で詳しく述べる。

5.3. システムの協調動作

本システムは協調動作として以下の動作を行う。

(1) 推論の開始

エージェントはプロトコル・マネージャから問題を受け取り、推論を開始する。

(2) 推論結果の送信

エージェントは推論の中間結果、および結果を定期的にプロトコル・マネージャに送信する。このとき、「提案」「応答」「再評価」のいずれかの形式で情報を送信する。

プロトコル・マネージャに送られた情報は、エージェントが受け取り、情報の種類に応じて処理を行う。

(3) 「質問」の送信

エージェント内で推論の継続が不可能になると、他のエージェントに対して推論の継続を依頼する。このときの情報の形式は、「質問」である。

(4) 「質問」を受け取り「応答」を送信する

受け取った情報が「質問」であった場合、エージェントはそれを認識し、他のエージェントが推論できなかった部分についての推論を行う。推論終了後、「応答」の形式で結果を送信する。

(5) (2)～(4)を現在のステージの解であると思われる情報ができるまで繰り返す。

(6) エージェント内で、現在のステージの解であると判断すると、「再評価」の形式で中間解を送信する。

(7) 「再評価」

受け取った情報が「再評価」の形式である場合は、エージェントは受け取った情報に対する評価値を求め、「応答」の形式で送信する。

この段階で、どのエージェントからも高い評価値をつけられた情報は、システムが求めた解として信頼度が高いと判断する。エージェントによって評価値にばらつきがみられる場合は、各エージェ

ントが求めた評価値の平均をとる。これは、エージェント間の妥協によって求められた解であると判断し、評価値が高ければ、信頼性があると判断する。

以上の動作を行い、最終段階の評価値の高い順にいくつかを次のステージへの問題として推論を行う。

最終ステージの推論が終了し、評価値のもっとも高い情報が、システムが求めた解となる。

6．共通表現形式

エージェント間で情報の交換を行う際、他のエージェントがその情報を読み取れる形式で情報を送信しなければならないのは明らかである。

エージェントが1対1で通信を行うのであれば、互いに相手のエージェントの表現を理解できれば良い。しかし今回のシステムでは、1対1の通信ではなく、通信の相手は複数あるため1対1の通信方法をそのまま行うと、複数ある通信相手全ての表現を各エージェントが持っていなければならない。この方法の問題点は、

(1) システムに新しいエージェントを組み込むときに、全てのエージェントに新しく組み込まれたエージェントの表現をもたせなければならない。
(2) 通信を行うときには、必ず通信相手を指定して行わなければならない。
(3) エージェントの数だけ表現の変換系を持たなければならないので、エージェントの数が増えるとそれぞれのエージェント内に含めることが困難になる。

などがあげられる。

そこで、エージェント間の共通の表現（以下「共通表現形式」と呼ぶ）を用いることを仮定する。共通表現形式をR_cと表すことにする。共通表現形式を用いると、エージェントX，Yの知識表現モデルをそれぞれR_X，R_Yとすると、エージェントXからエージェントYへの情報の送受信は次のような手順になる。

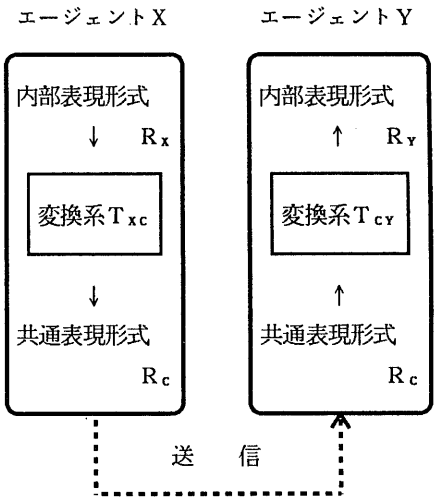

図3 共通表現形式の必要性

(1) エージェントX内の変換系T_{XC}において知識表現モデルR_Xから共通表現形式R_Cに変換する。
(2) エージェントXからエージェントYに送信する。
(3) エージェントY内の変換系T_{CY}において共通表現形式R_Cから知識表現モデルR_Yに変換する。

この方法を用いると、先に示した1対1通信の方法を用いたときの問題点はすべて解消される。

また、通信の度に通信相手を指定しなくても良い。これは、エージェントの導いた中間解を、よりたくさんのエージェントに評価してもらえるため、解の信頼性において良い結果をもたらすと考える。

しかし、共通表現形式を用いる場合の問題点として、次のようなことが考えられる。

(1) 共通表現形式の変更を行った場合、すべてのエージェントに対して変更を加えなければならない。
(2) 表現の変換を2回行わなければならない。
(3) 汎用的な共通表現形式というものが存在する

のかどうか。

(1)については、共通表現形式の設計を十分検討して行えば避けられると考える。(2)については、共通表現形式に変換するためには避けられないが、利点に挙げたことが重要であるので、あまり問題にはならないと考える。

従って、共通表現形式を用いた情報の交換を行うことにする。(3)の問題が残るが、エージェントが問題を引き継ぐことができるだけの十分な情報を交換することにして、その書式のみを決めておくことにする。

7．協調による推論結果

7.1. 推論状態の保持

5．で述べたように推論は進められ、解を求めることができる。このとき、推論の中間状態としてエージェントがプロトコル・マネージャに送信する情報は、プロトコル・マネージャに保持する。

今回本システム上にインプリメントするアプリケーションは、その問題の性質上探索に問題がおかれるため、エージェントから送られてくる情報（共通表現形式）を木構造の形で保持する。木のノードは、推論の中間状態を表す。あるノードはその親ノードから派生した中間状態であることを示す。つまり、推論が進むにつれて木は深くなる。

プロトコル・マネージャはエージェントからの要求により、この木構造を操作する。情報の内容には関知しないという条件があるため、プロトコル・マネージャが意図的に木を操作することはない。

エージェントから情報が送られてきた場合、情報が「提案」であることは、木に新しくノードを追加することになる。「質問」である場合には、木にノードを加えることは変わらないが、そのノードには評価値はない。このノードの評価値は、他のエージェントの「応答」によりつけられる。

エージェントが情報を受け取る場合は、木構造のあるノードの情報をもとに、その状態をエージェント内に復元することになる。これにより、推論の継続および「再評価」が可能になる。

7.2. 解をまとめる方法

各ステージの推論が終了すると、中間解および最終的な解を選択することは先に述べたが、その方法として、次の二つが考えられる。一つは、複数の中間解から一つの新しい解を生成する方法であり、もう一つは複数の中間解から一つの中間解を選択する方法が考えられる。前者は、エージェントが中間解それぞれの内容を検討しなければならず、実現することはかなり困難であると思われる。後者は、解の信頼性を示す尺度をもうける必要がある。

前者を実現することが理想的ではあるが、今回は後者の方法をとることとした。このため、選択の基準が必要になり、先に述べた「評価値」を導入することにした。この評価値は、各エージェントが別々の観点で算出する、その情報の信頼度を示す値である。しかし、評価値を算出する基準はアプリケーション・プログラムによるところが大きいため、アプリケーションによって定められることとし、システムがはじめから評価値基準を持っているわけではない。ただし、値の範囲がまちまちであると、比較することの意味がなくなるので０～１００の値をとることとした。

本システムでは、各ステージの推論が終了したときエージェントは、送信する情報を「再評価」の形式で送信する。この情報を受け取ったエージェントは情報に対する評価値を求め、送信する。

ここで、すべてのエージェントから高い評価値をつけられた情報は、システムの解として見たときに信頼性があると判断することができる。しかし、エージェントによって判断がまちまちになった場合には、平均値をとり、この値が高い場合には信頼性があると判断することにする。

一定数の情報に対して再評価を終えたとき、評

価値の高い順にいくつか（最終ステージでは一つのみ）を選択する。

8．まとめ

現在のシステムでは、協調のための通信を集中管理するプロトコル・マネージャが存在し、本来の分散協調型問題解決システムではない。この集中管理型のプロトコル・マネージャに障害が発生した場合に、すべてのエージェントの動作が保証できなくなる。これは、分散することの利点である「障害に対する強さ」が失われてしまう。

この欠点を解消するため、各エージェントに分散したプロトコル・マネージャを実現することが必要である。

どのような方法で情報に対する評価値を決定するのか、どのような協調動作を行えば良いか、ということは、難しい問題である。本稿では、協調動作について一つの考え方を示した。今後、解の質と応答時間とのバランスなどの問題について考察を続けていく予定である。

参考文献

[1] E.H.Durfee, V.R.Lesser, D.D.Corkill: "Trends in Cooperative Distributed Problem Solving", IEEE transaction on knowledge and data engineering, Vol.1, No.1. (1989)

[2] Susan Leigh Star: "The Structure of Ill-Structued Solutions: Boundary Objects and Heterogeneous Distributed Problem Solving", Distributed Artificial Intelligence.

[3] Gregor von Bochmann: "Concepts for Distributed System Design". (1983)
邦訳: 水野忠則, 井手口哲夫: 「分散処理システムデザイン」(1987)

[4] 小林重信: 「知識工学」(1986)

[5] 上野晴樹: 「知識工学入門」(1985)

[6] 上野晴樹, 石塚満: 「知識の表現と利用」(1987)