ソフトウェア基礎論　37－17
プログラミング言語　27－17
（1990. 12. 14）

分散協調マルチエージェントモデルによる通信網の複数回線設定制御

湊　賢治　　奥村　康行　　岸本　了造
ＮＴＴ伝送システム研究所

あらまし　本論文は、複数回線設定問題における分散協調マルチエージェントモデルの適用法を提案し、そのモデルを分散環境上に構築し動作を確認した。このマルチエージェントモデルは、複数のエージェントが互いに対等の権利を有し、非同期かつ並行に動作し、すべてのエージェントを統合するようなシステムの存在しない形態である。複数回線設定問題にこのモデルを適用するため、この問題を非同期に解くことが可能なような部分問題への分割法、複数のエージェントがこの部分問題を解決した後の解の統合方法ならびにエージェント間通信プロトコルを明らかにした。さらに、エージェント数が１つの場合と４つの場合について比較し、マルチエージェントモデルにおいても実用的に計算処理可能である事を示した。

Multipath Allocation Control in Communication Network using Cooperative Distributed Multi-agent Model.

Kenji Minato　Yasuyuki Okumura　Ryozo Kishimoto
NTT Transmission Systems Laboratories
1-2356 Take, Yokosuka-shi, Kanagawa-ken, 238-03 Japan

Abstract

This paper proposes an application method of Cooperative Distributed Multi-agent Model in Multipath Allocation Algorithm, and confirms its valid calculation results by realizing this model in distributed computer environment. The multi-agent model, in this paper, is an environment where each agent has the same function of assynchronus and parallel calculation, and no grobal controlling agents exist. To realize this model, this paper proposes an appropriate dividing method of multipath allocation problem for assynchronus and parallel calculation, unifying method of each agents' answers, and inter-agent communication protocols. It also compares the required calculation time of single agent with that of 4 agents, and shows the multi-agent model's practical calculation ability.

1．まえがき

今後の通信網においては、回線の再構成、カスタマコントロール、パーソナル通信等を実現可能とするために、より柔軟で知的な制御が望まれる。さらに、通信網の大容量化や多品種化の傾向はますます進行していくと思われる。そのような複雑化する通信網の制御を行なおうとした場合、全国１ケ所の制御システムで処理する集中制御方式では、負荷の過度な集中や、データベースの肥大化等の問題が生じる。さらに、集中制御は集中システムの障害により全国のシステムがダウンしてしまう問題もある。これらの問題を解決する方法として、システムの分散化は有効な手段であると思われる。

また、情報化社会の進展に伴い、通信網は産業基盤としてその重要性は増すばかりであり、その障害が社会に及ぼす影響は計り知れないものがある。そのため、システムの２重化等による高信頼化と共に、回線切断時の迅速な迂回救済が必要となる。この迂回救済の技術は回線の再構成等に応用する事も可能である。

本論文では分散制御システムによるネットワーク制御を考慮し、特に、分散環境における回線の迂回救済法について述べる。迂回救済は、複数の回線を限られたネットワーク資源中に設定する問題であり、グラフ理論における多種フロー問題である。

ここでは、システムの分散化の形態として地域分散を取り上げ、その分散環境上で地域にまたがる回線を分散システム間で交渉を行いながら設定する方法について述べる。

まず、従来までの複数回線設定法について説明し、従来方式を分散環境に適用する際の根本的な問題について述べる。そして、問題点を解決した新たなアルゴリズムの提案を行なう。さらに、提案アルゴリズムをワークステーション上にシミュレートし、その動作を確認した。

2．複数回線設定問題について

回線設定とは、ネットワーク上の指定された２ノード間の経路を探索する問題である。ここで、ネットワークはノードとリンクで構成されているとする。リンクはノード間を接続するものであり、所定の容量を有す。回線の設定要求は始点ノード、終点ノードと、回線に必要な容量を示す。そして、あるリンクにはその許容量以上の回線の経路は設定できない。

１回線だけの経路探索では、ダイキストラ法と呼ばれるラベリングにより最短経路を探索する方法が最も高速とされている。このダイキストラ法は、ネットワーク上の各ノードに始点ノードからの距離をラベルとして付与していきながら探索を進めていく方法である。図１に２５個のノードが格子状に配置された例を示す。黒色のノードが既に探索されてラベル値が付与されているノードで、ラベル値は（）の中の数字で示してある。白色のノードはまだ探索されていないノードである。探索の進め方は、ラベルが付与されているノードの中で最もラベル値の小さいノードを選択して、そのノードに隣接するノードのラベルに値を付与する。選択されたノードをノードＡとし、ノードＡに隣接するノードをノードＢとすると、ノードＢのラベル値は、ノードＡのラベル値にノードＡとノードＢの間の距離を足した値を付与する。この処理はノードＡに隣接するすべてのノードに対して行なう。ただし、隣接するノードに既にラベル値が付与されており、そのラベル値が付与しようとしているラベル値よりも小さければラベルの付与は行なわない。図１の例ではノード間の距離はすべて１としてある。そうして、終点ノードに探索が至ると、終点のノードのラベル値は始点からの最短距離を示す事になる。そして、終点から始点に向かってラベルが書き換えられた順にたどる、最短経路が得られる。

次に、複数回線設定問題を考える。複数回線設定とは、限られたネットワーク資源の中にできるだけ効率良く多数の回線を収容する問題である。この問題は、ネットワークにおいてリンクが切断等によって障害が発生した場合、切断されたリンクに収容されている回線を他の経路に迂回させる時や、１度設定した回線を再構成する事によりネットワークをダイナミックに運用する時等に必要となる。しかし、この計算は最適な解を得ようとすれば莫大な計算時間を必要とする。

一般に、ネットワーク上で始点ノードと終点ノードを結ぶ経路の候補は複数考えられ、複数回線の設定のためには、それら複数の回線の複数の経路候補の組み合わせを計算しなければならない。図１の例で見ると始点から終点にいたる経路候補はおよそ８００個存在する。別の終始点間にも同様の数の経路候補が存在するため、それらの組み合わせをすべて計算するには莫大な計算時間が必要である事がわかる。この計算手間は、ノード数と設定要求数が増えるにしたがって指数

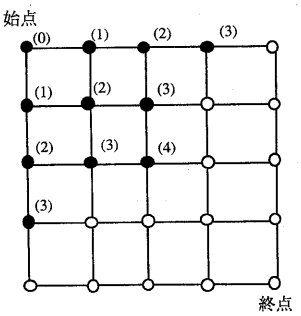

図１　　ダイキストラの探索方法

関数的に増大する。このような膨大な計算を実用的な時間で解を得るのはとても不可能であるため、近似的に解を求める必要が生じる。

複数回線設定問題に対する近似的な解法の基本的なフローを図２に示す。この方法は、経路を探索する順番を予め決めて、その順番に従って最短経路を探索する方法である。ｉ番目に探索する経路は、ｉ-１番目までの経路が使用している資源は利用できないという条件の元で探索を行なう。この方法は、互いに輻輳のない経路を高速に求めることが出来る。しかし、組み合わせの検討がなされていないので、探索する順番が大きく影響する。そこで、なるべく経路設定数を上げるため、従来ではこの基本フローを元に、経路探索の順番をヒューリスティックな方法で決めたり[1]、最小カットを用いる方法[2]等が提案されている。

３．分散協調による複数回線設定問題

３.１　分散による制御

ネットワークの制御を地域的に分散されたシステムで行なう事は、データベース負荷の軽減及び、耐災害性等において有用である。この問題は、制御システムをエージェントとして、エージェント間のネゴシエーションによる分散協調問題解決と言える。（図３）

本論文で対象とした分散環境における回線設定問題は、個々のエージェントは自分の管轄地域の情報だけをデータとして持っており、必要な情報を互いに交換しながら、エージェントの管轄間にまたがる回線の設定を行なっていく。この問題は分散システムの考察に適した問題であり、分散協調の研究を行なう例題として、Conry[3]らも取り上げている。しかし、そこで扱われている問題はノード数が１０程度で、回線数はせいぜい２本程度の小規模なものを対象としている。そこで述べられている内容は、エージェント間にまたがる経路はプランとしてすでに幾つか求められているとして、その幾つかのプランの中から１つをエージェント間の交渉によりに選択する事である。しかし、分散環境で複数回線設定を行なう場合、そのプランを求める事も大きな問題となる。

本論文では、分散環境上で経路探索を行ない、かつ、複数の回線の経路が互いに輻輳する事なく設定する方法について考察する。また、本論文ではノード数が１００程度の規模のノットワークに１０００個程度の回線を設定する事を対象とする。

３.２　分散環境上の探索の問題

本論文で構築を目指すネットワーク制御の形態を以下に整理する。

１．各エージェントは地域的に分散されており、各エージェントは管轄地域の情報のみを有している。

２．各エージェントは非同期で動作し、すべてのエージェントに共通なメモリー空間や、各エージェントの動作を調整するような中央処理システムは存在しない。

３．各エージェントは他のエージェントからの返答が返ってくる間、自分の計算処理を止めて待つ事はせず、各エージェントは並行かつ同時に処理を進める。

次に、このような形態を念頭において分散環境で複数回線を設定する場合、前章で示した従来の複数回線設定の基本フローではどのような問題があるか以下に示す。以下、問題点とその例を示し、本論文で提案する手法ではどのように問題点を解決したかを述べる。

「問題１」

エージェントは、どのエージェントに継続探索の依頼を出せば良いかわからない。

「例１」

図４においてエージェントＡの点ｓからエージェントＤの点ｔまでの経路を探索する場合、エージェントＡは継続探索の依頼をエージェントＢとエージェントＤのどちらに送ればより良い結果が得られるか解からない。

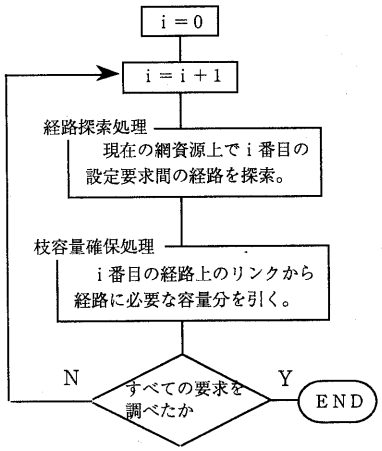

図２　複数回線設定方法の基本フロー

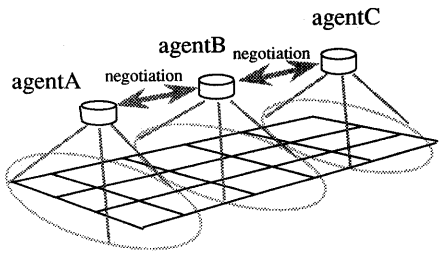

図３　　地域分散による制御

「解決法」

特定のエージェントに探索依頼を送ると経路を発見出来ない恐れがあるため、隣接するすべてのエージェントに依頼を送る事にする。

「問題２」

エージェントは隣接エージェントにどのような継続探索を依頼すれば良いかわからない。

「例２」

図５においてエージェントＡは点ｓから隣接エージェントまでの経路を図のように探索できたとする。ここで、エージェントＡはどの経路の継続探索を隣接エージェントに依頼すれば良いか解からない。同様の事が終点ノードに至るまでの中継エージェントにも言える。

「解決法」

従来の分散協調による経路探索の手法では、エージェントのローカルな情報だけから最も良いと思われるプランを選択し、その選択したプランの継続探索だけを依頼するという方法がよく取られている。しかし、この方法では探索の範囲を限定してしまい、始終点間に存在する経路を見つけだす事ができなくなる恐れがある。そのため、すべての経路の探索を継続する必要がある。しかし、１つ１つの経路毎に探索の継続を依頼していたのでは多くの手間を必要とするため、ここでは、ダイキストラ法の特徴を利用して、エージェント間の境界に存在するノードのラベル値を送る事にした。そうすると、継続探索依頼を受けたエージェントは、ラベル値を管轄内の境界に位置するノードに付与して、ダイキストラ法による経路探索を継続することにより、依頼元のエージェントが探索したすべての経路の探索を継続する事が可能となる。

「問題３」

エージェントは、継続探索の依頼を出した経路プランはすべて確保しておかなければならない。確保というのは、そのプランの経路が利用するリンクの容量は他の設定要求の回線に使用させないで空けたままにしておくという事である。そうしないと、依頼の返答が返ってきた時、経路を設定する事が出来なくなる。しかし、この経路確保を行なおうとすると大きな問題に直面する。この問題は、複数回線設定を分散環境で行なおうとした際の一番重要な問題である。その問題を２つの仮定のもとで説明する。

「仮定１」

探索依頼の返答が返ってくるまでの間、エージェントは何もしないで待つのでなく、別の要求の経路を探索するとした場合。つまり、エージェント間で並行して計算を進めると仮定した場合である。

この仮定のもとでは、ネットワーク上のリンクで確保されている容量分は使えないという条件の元で経路探索をしなければならない。これは、探索範囲を著しく狭くし、リンクに空きがあるにもかかわらず経路を探索できなかったり、必要以上に遠回りな経路を設定してしまう事になる。これでは、設定率を著しく落としてしまう。本当に使用するかどうか解からないプランのためにリンク容量を確保しておくのは大きな無駄と言えよう。

「仮定２」

探索依頼の返答が返ってくるまでの間、処理を停止して待つとした場合。

この仮定のもとでは、プランを確保しておいても結果が来るまで待つのであるから、他の設定要求に影響を与える事はない。よって、逐次処理と同様な計算が行なえ、仮定１で示した問題はない。しかし、この場合では当然の事ながら何もしない空白の時間が存在し、計算時間を多く必要としてしまう。そのため、分散にしたことによるメリットの１つと考えられる、処理の分散化による高速化が全く期待できないことになる。

このような、トレードオフが存在してしまう理由は、経路探索は１回線の探索だけでは成り立たず、複数の回線が相互に影響しあう点にある。このような相互依存の問題は回線設定の問題に限らず多くの問題に見られる。特に、非同期分散処理のように、早い時間に行なった処理が優先される事がなく、逐次処理の概念が全く通用しない処理形態では、このような相互依存の問題は非常に厄介な問題となる。つまり、ある時間においてあるエージェントが知りうる情報を元に得た解答が、他のエージェントへ送られる間に、その答えと前後して他のエージェントによって出された解答と矛盾してしまう事態に陥るからである。

分散処理におけるこのような問題を解決する方法としては、黒板システムのようにすべてのエージェントからアクセス可能な空間を設けて処理の同期化を計る方法があるが、黒板システムがボトルネックとなる可能性があり、好ましくない。

本論文では、エージェントに共通な空間の存在しない条件で、以上のトレードオフの問題を克服した複数回線設定法について述べる。

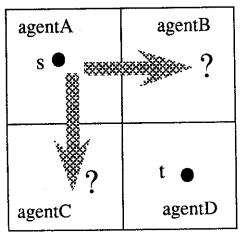

図４　問題１の例

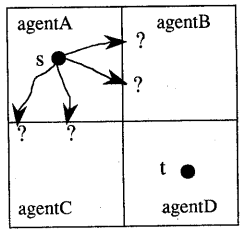

図５　問題２の例

4．提案手法について

4.1　割付法の提案

前章で述べた問題を解決するために、従来とは全く異なる新たなアルゴリズムを提案する。それは、経路探索と輻輳確認を別々のフェーズに分ける方法であり、割付法と呼ぶ。

割付法の基本フローを図6に示す。まず、経路探索フェーズにおいてすべての設定要求の最短経路を探索する。この探索は、リンク使用状態が同一であるという条件のもとで行なう。つまり、このフェーズでは他の設定要求にどのような経路を選んだかは全く関係なしに経路を探索する。これにより、複数の設定要求に対する経路を同時に探索する事が可能となる。このようにして設定していくと、リンクの許容容量以上に経路が存在する可能性が出てくるため、次のフェーズで確認を行なう。

輻輳確認フェーズでは、ネットワーク上のすべてのリンクの輻輳をチェックする。輻輳とは、回線がリンクの許容容量を超えて設定されている事を言う。リンクのチェックで輻輳が存在したら、そのリンクに設定されている回線を優先順序に従って許容容量以下になるまで削除する。優先順位は、経路長で決めたり、回線の容量で決めたり、もしくは、予め設定要求に優先順位を設けておく等の方法が考えられる。そして、削除された回線は、次の経路探索フェーズで再び経路を探索する。

2回目以降の経路探索は、先程の輻輳確認フェーズで削除されなかった回線とは輻輳しない経路を探す。そうして、経路探索フェーズにおいて、未だ経路が設定されていない設定要求に対して新たな経路が見つからなくなったら計算を終了する。また、この計算フローの1回目のサイクルですべての設定要求の経路を探索する事が不都合であれば、何回かのサイクルに分けて行なっても問題はない。後に述べる分散環境でのシミュレーションでは分けて行なっている。要点は、ある設定要求にとって、1回目の経路探索はまだどの設定要求もリンクを利用していないという仮定のもとに計算を行ない。2回目からは輻輳確認で削除されなかった回線が使用しているリンクの容量は使用しないようにすれば良い。これによって、なるべく最短な経路を無限ループに陥ることなく求めることが可能となる。

この割付法の特徴は、設定要求間の相互依存を気にしないで経路探索ができる点である。ある設定要求はいつ経路探索を行なってもよく、経路探索フェーズと輻輳確認フェーズを並行に処理しても全く問題がない。この事は非同期分散には非常に好都合である。あるエージェントから送られてきた継続探索の処理はいつでもできるし、他のエージェントに送った継続探索の返答はいつ戻ってきても構わない。また、他のエージェントに継続探索を依頼した経路を他の設定要求の回線に利用されないように確保しておく必要はなく、経路プランは記憶しておいて返答が帰ってきた時に該当するプランを経路として設定すればよい。

よって、この割付法の採用により、前章で示したトレードオフを解決する事ができる。

4.2　割付法の分散環境への適用

前節で説明した割付法を実際の分散環境に適用する。表1に適用した分散環境の形態を、図7にそのモデル図をそれぞれ示す。各エージェントはネットワークの1部分を管轄し、管轄地域内の知識だけを元に、エージェント間でメッセージを交換しながら処理を進める。

具体的な処理のフローは図8に示す。これは、図6のフローに中継フェーズを加えている。経路探索フェーズは、始点から経路の探索を開始すると共に、終了の判断を行なう。中継フェーズでは、他のエージェントから送られてきたメッセージに対して、各処理を行なう。このフェーズは、メッセージに対して迅速な応答を行なうため1計算サイクルに2回行なう。そして、輻輳確認フェーズでは、設定した経路で管轄内において輻輳のチェックを行なう。

各エージェントは、これらのフェーズを図9に示す様に非同期かつ並行に処理を行なう。図は2つのエージェントの動作関係を示しており、矢印はメッセージを表す。メッセージは必要に応じて相手先のバッファに送られる。以上の動作を終了まで繰り返す。各フェーズの処理は図10に、各メッセージの内容については表3にそれぞれ示す。

このアルゴリズムで行なう処理は大きく分けて、経路探索処理、回線削除処理、計算終了処理の3つである。これらの処理がどのフェーズで行なわれるかは表2に示す。以下に、各処理に関して説明を行なう。

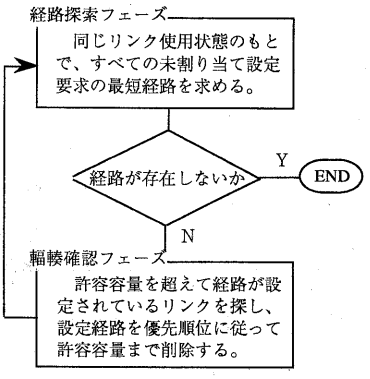

図6　割付法の基本フロー

（1）経路探索処理

1つの設定要求に着目した経路探索処理の流れを図11に示す。図はネットワーク上における探索の様子と、各エージェントにおける処理の流れを示している。ネットワークは4分割されており、各分割されたエリアをエージェントAからDがそれぞれ受け持っている。例では、エージェントA上の始点sからエージェントD上の終点tまでの回線設定要求があるとする。以下、各段階毎に説明をする。

ステップ1：

始点を有するエージェントが始点から境界までの経路を探索する。この探索は、ダイキストラ法を用いて1回の計算サイクルで、境界までの可能なすべての経路を求める事が出来る（図中の矢印付き実線）。管轄内の経路を求めた後は、隣接するエージェントに継続探索メッセージを送出し、探索の継続を依頼する。その時、求めた経路はプランとして記憶しておく（ステップ2図中の破線）。

継続探索メッセージは、表3にあるようにメッセージ名、設定要求情報、境界情報で構成されている。設定要求情報における識別番号は、この探索がこの設定要求にとって何回目の探索であるかを表す。この識別番号がないと、非同期環境では処理の矛盾をきたしてしまう。図11では、境界までの経路長を（）、識別番号を「」で表している。

以上の処理は、エージェントが経路探索フェーズの時に行なわれる。このフェーズでは、始点が存在する設定要求の経路をすべて1度のフェーズで探索する事はしないで、30回線ずつ行なう事にした。この理由は1度にすべての経路を探索すると、大量の継続探索メッセージが発生し、メッセージを蓄えておくバッファが容量オーバーするためである。1度に探索する回線数は計算システムによって適する値を選べば良いであろう。

ステップ2：

継続探索メッセージは、メッセージを受け取ったエージェントが中継フェーズになった時に読みだされ処

表1　分散環境への実装

分散協調 マルチエージェントモデル			実装システム (SUN4に実装)
エージェントの配置		・複数のエージェントを配置.	・部分網に各エージェントを配置.
知識の分割	知識の分割 (DB構造)	・通信網の知識を各エージェントが所有.	・部分網内の知識のみ所有.
	知識,解析結果の共有	・分散協調による知識、解析結果の共有.	・協調処理による解析結果の共有.
問題の分割		・問題を複数のサブ問題に分割し、その分散アルゴリズムを割り当てる.	・部分網内だけの探索結果を相互にやり取りして、全体の回線を設定する.
エージェントへの問題の割当		・同報通信プロトコルなどによってサブ問題を複数のエージェントに割り当てる.	・部分網内に始点を有する回線の経路探索問題を割り当てる.
問題の解決 解の統合		・分散アルゴリズムの実行. ・分散協調型問題解決プロトコルを用いて個々のエージェントで処理した解析結果をエージェント間で情報交換する.	・他のエージェントと継続探索依頼メッセージや経路決定メッセージをやり取りすることにより全体として回線設定を行なう.

知識：ノード、リンクなどの状態を表わす通信網の基本的な知識
解析結果：問題解決のための予備評価結果、計算結果などの知識

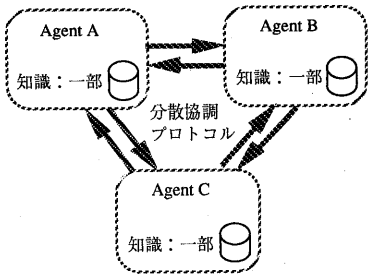

図7　エージェントモデル図

表2　各処理が行なわれるフェーズ

基本処理	処理を行なうフェーズ
経路探索処理	経路探索フェーズ　中継フェーズ
回線削除処理	輻輳確認フェーズ　中継フェーズ
計算終了処理	経路探索フェーズ　中継フェーズ

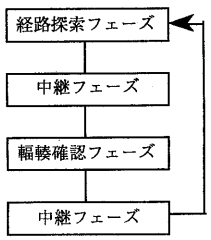

図8　分散環境における割付法のフロー

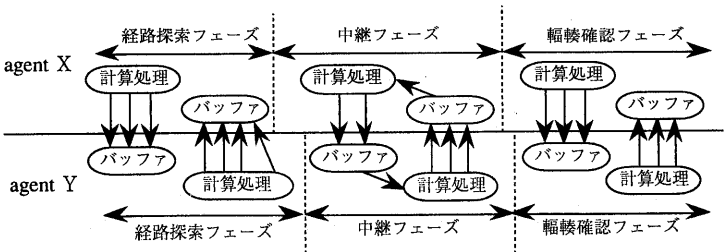

図9　エージェント間の相互動作図

理される。

図では、探索を行なった結果、エージェントBは境界までの経路が存在しない事が判明した。そうすると、エージェントBはその事をエージェントAに、プラン消去メッセージとして知らせる。

エージェントCは経路を見つけたので、継続探索メッセージをエージェントDに送出する。

ステップ3：

エージェントAが中継フェーズになった時、エージェントBからのプラン消去メッセージを読みだす。そして、エージェントBに依頼を出したプランを記憶から消去する。この記憶しているプランがすべて消去されたら、エージェントA（その設定要求の始点を有するエージェント）はその設定要求に対する経路が存在しない事を自覚する。

エージェントDが中継フェーズになった時、エージェントBからの継続探索メッセージを読みだし、探索の継続を行なう。そして、終点に到達したので経路を確定し（図中の実線）、経路決定メッセージを経路に沿って伝える。

ステップ4：

エージェントCが中継フェーズになった時、エージェントDからの経路決定メッセージを読みだす。そして、記憶しているプランの内で該当するプランを経路として決定させる。残りのプランは消去する。

エージェントCは始点を有しないので、経路決定メッセージを決定した経路に沿って伝搬させる。

ステップ5：

エージェントAが中継フェーズになった時、記憶しているプランの内で該当するプランを経路として決定させる。エージェントAはこの経路探索の発信元であるため、この設定要求の経路が求まったこと事になる。

（2）回線削除処理

回線削除処理の例を図12に示す。以下、ステップに沿って説明する。

ステップ1：

エージェントDが輻輳確認フェーズになった時、管轄内のあるリンクが許容容量オーバーに成っているのを確認し、そのリンクを利用している回線で優先順位の低い回線を削除する。そして、隣接エージェントの中でその経路が通過しているエージェントに向かって回線削除メッセージを送る。

ステップ2：

回線削除メッセージは、メッセージを受け取ったエージェントが中継フェーズになった時に読みだされ処理される。

エージェントCは、回線削除メッセージを受けると、該当する回線を削除する。エージェントCはその回線の端点（始点、終点）を含まないので、さらに回線削除メッセージを経路に沿って送る。図ではエージェントAに対して送出する。

ステップ3：

エージェントAが中継フェーズになるとメッセージを読みだし、処理を行なう。エージェントAは削除回線の始点を含むため、回線削除処理を終える。

この削除された回線は、エージェントAが経路探索フェーズになった時に、再び経路探索を行なう。

（3）計算終了処理

図13に終了のフローを示す。

経路探索フェーズにおいて、計算終了の条件が満たされたら、計算終了メッセージをブロードキャストする。計算終了メッセージを受け取ったエージェントは、自分の状態が終了状態であれば終了許可を、計算状態であれば終了不許可を、終了返答メッセージに乗せて返信する。終了返答メッセージには、計算終了メッセージに付与されていた識別番号を付与して送る。これにより、どの計算終了メッセージに対する返答であるかを明確にする。計算終了メッセージを送出して返答を待っているエージェントは、すべてのエージェントから終了許可の終了返答メッセージを受け取ったら、計算を終了する。

このような終了メッセージは、どれか1つのエージェントが非終了状態であると、終了ルーチンを繰り返すため、その繰り返し回数が大量に増える恐れがある。しかし、本シミュレーションでは数回程度で済んでいる。これは、計算終了メッセージを送出するための前提として、記憶域にはプランが残っていないという条件が付加されているためである。記憶域のプランを削除出来るのは、隣接エージェントからのメッセージのみである。そのため、記憶域にプランが存在しないという事は、全体としても処理がほぼ終了している事を示している。

5．シミュレーション

提案アルゴリズムを計算機上に実装し、シミュレーションを行なった。実装は、各エージェントを1つのプロセス上に構築し、エージェント間の通信はプロセス間通信を使用した。シミュレーションは、設定要求の違いによる計算時間の違いをエージェント数が1つの場合と4つの場合について行なった。

分散システムを評価する場合は、システム全体として評価する必要があり、本論文で取り上げた複数回線設定問題だけを対象にして評価すべきではない。しかし、今後分散システムを構築して行く上で、複数回線設定問題という視点から見た評価は重要である。

一般に、1つの回線の経路を求める場合、その計算

経路探索フェーズ

処理			送出メッセージ
始点から経路探索開始 →	終点が存在するか N →		継続探索メッセージ
終了状態の確認 →	終了状態か Y →		計算終了メッセージ

中継フェーズ

受理メッセージ	処理		送出メッセージ
継続探索メッセージ	探索の継続 →	終点が存在するか Y →	経路決定メッセージ
		N →	継続探索メッセージ
	Y ↑ 経路が存在する N →		プラン消去メッセージ
経路決定メッセージ	経路の決定 →	始点が存在するか N →	経路決定メッセージ
プラン消去メッセージ	該当するプランを消去 →	端点が存在するか N →	プラン消去メッセージ
回線削除メッセージ	決定経路の削除 →	端点が存在するか N →	回線削除メッセージ
計算終了メッセージ	終了状態の確認 →		終了返答メッセージ
終了返答メッセージ	計算終了可能か Y →	計算終了	

輻輳確認フェーズ

処理			送出メッセージ
輻輳の確認 →	回線を削除したか Y →	他管轄にまたがる回線か Y →	回線削除メッセージ

図１０　各フェーズの処理

表3　各メッセージの内容

メッセージ名	内容	送り先	メッセージの構成内容
継続探索メッセージ	探索の継続を依頼する。	隣接エージェント	メッセージ名 設定要求情報として 要求名（回線名） 識別番号 始点と終点 必要な容量 境界情報として 境界ノードのラベル値情報
経路決定メッセージ	経路が決定した事を伝える。	隣接エージェントで、決定経路のプランを有しているエージェント	メッセージ名 発エージェント名 設定要求情報として 回線名 識別番号 経路情報として 総経路長 境界リンク名
プラン消去メッセージ	プランに該当する経路が存在しないという事を伝える。	隣接するエージェントで、存在しなかった経路のプランを有しているエージェント	メッセージ名 発エージェント名 設定要求情報として 回線名 識別番号
回線削除メッセージ	輻輳のため回線が削除された事を伝える。	隣接エージェントで、削除される経路を有しているエージェント	メッセージ名 発エージェント名 設定要求情報として 回線名 識別番号
計算終了メッセージ	他のエージェントに終了の許可をもらう。	すべてのエージェントにブロードキャストする	メッセージ名 発エージェント名 識別番号
終了返答メッセージ	終了エージェントに終了の許可か不許可を返答する。	返答先のエージェント	メッセージ名 発エージェント名 識別番号 終了の許可か不許可

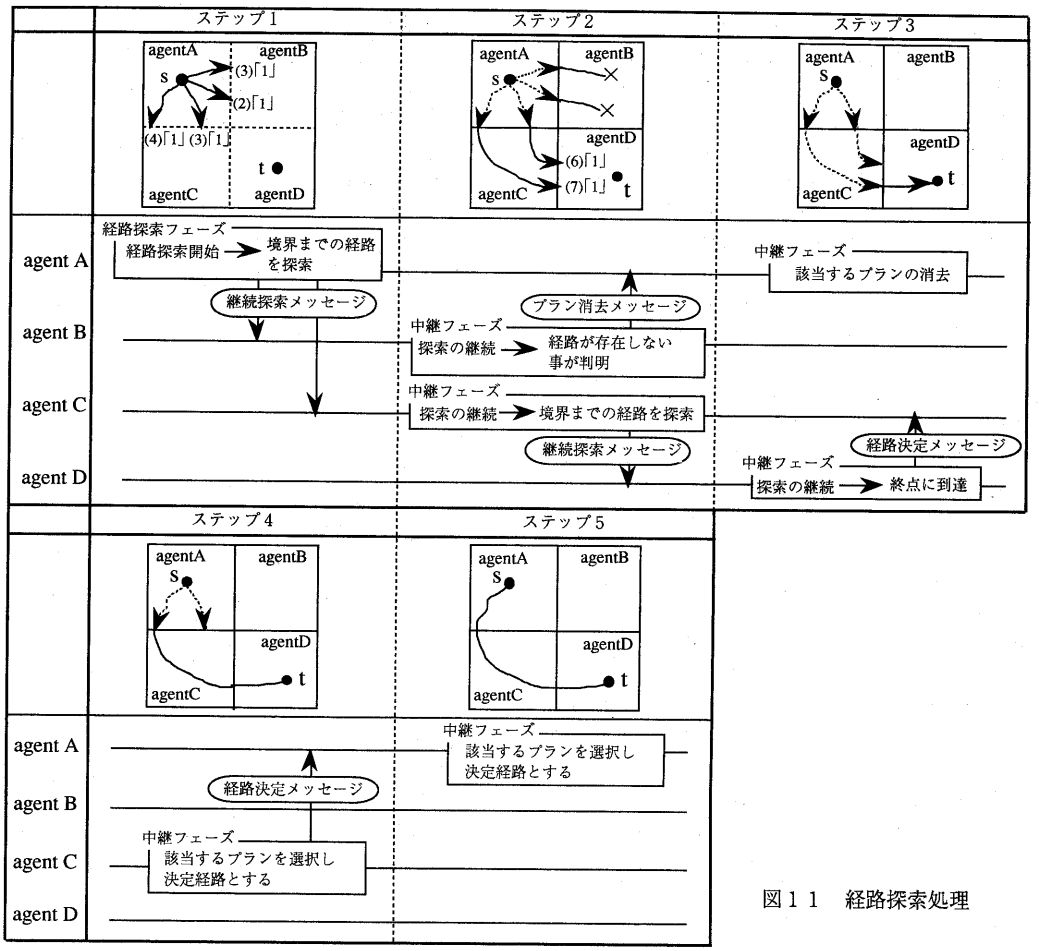

図１１　経路探索処理

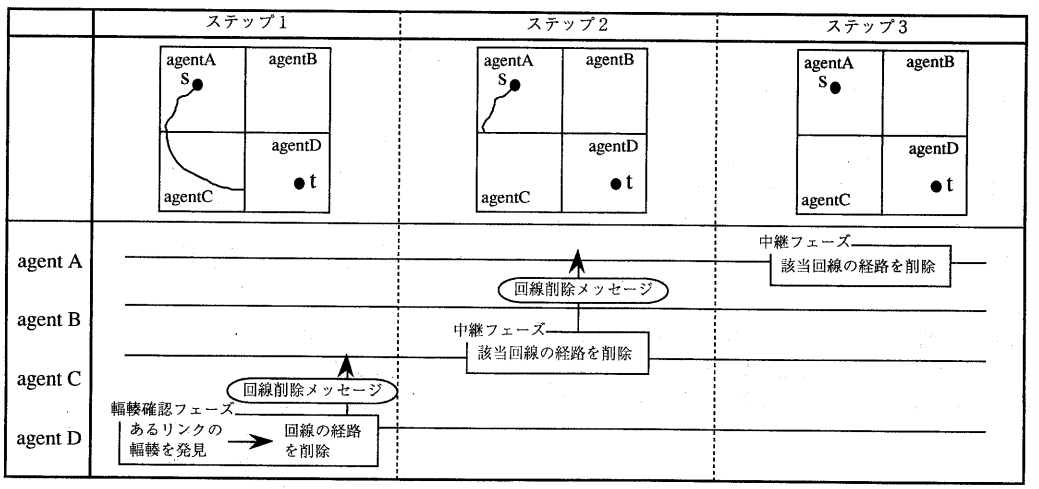

図１２　回線削除処理

の手間は、探索範囲が小さい程少なく、エージェント間にまたがると多くなる。つまり、分散環境で複数経路を求める場合、分散数を増やして１つのエージェントの探索地域を小さくすればその地域内の探索の手間は少なくなるが、エージェント間をまたがる経路が増すため全体としての計算手間は増えるかもしれない。そこで、どの程度の分割が良いかを考察する。

計算は、図１に示す様な格子状のネットワークを対象とし、その大きさはノード数が縦横それぞれ２０個ずつの４００個で、各リンクの容量は２０とする。そして、設定要求は１０００個で、各回線に必要な容量は１とした。設定要求の内容はランダムに始点と終点を選択するが、設定要求の内で何割かは始点と終点とが違うエージェントの管轄地域に属し、それ以外は同じ管轄地域にに属するとした。計算は、その割合を変化させた時のＣＰＵ時間と回線設定数を測定した。回線設定数は１０００個の要求に対して幾つ設定する事ができたかである。

図１４にＣＰＵ時間の結果を示す。横軸にある設定要求の管轄外率とは、すべての設定要求において、始点と終点が異なるエージェントに属する要求の割合を示している。エージェント数が４個のＣＰＵ時間は、すべてのエージェントなかで最も大きなＣＰＵ時間を採用した。回線設定数はエージェント数による違いはほとんどなかったため図示はしなかった。計算に使用した計算機は、Sun SPARC Station 1 である。

図１４を見ると、設定要求の管轄外率が５０％を切ると、エージェント数４個の計算時間が小さくなっている。そのため、設定要求の内の半分が管轄内で処理できるなら、マルチエージェント環境での実用の可能性がある事がわかる。

６．まとめ

分散環境に適した複数回線設定アルゴリズムである割付法を提案し、それを、分散協調マルチエージェントモデルに適用した。そして、そのモデルを実際の分散環境上に構築し動作を確認した。さらに、エージェント数が１つの場合と４つの場合について比較し、マルチエージェントモデルにおいても実用的に計算処理可能である事を示した。

謝辞

本研究を進めるに当たって、有益なアドバイスを頂いたＮＴＴ伝送システム研究所の三木　哲也伝送処理研究部部長、および槙　一光主幹研究員に深く感謝します。

［参考文献］

(1) 葉玉、太田、：”高速バースト多重伝送システムにおけるバーチャルパス経路制御の検討”、昭和６３年信学秋季全国大会、B-264.

(2) 翁長、有本、岸本、：”ネットワークにおける多種結線実現問題とその近似算法”、電子情報通信学会論文：昭５７－論２０２［A-52］.

(3) S.E.Conry, R.A.Meyer and V.R.Lesser, "Multistage Negotiation in Distributed Planning, " in *Readings in Distributed Artificial Inteligence,* A.H.Bond and L.Gasser,Eds. San Mateo, CA: Morgan Kaufman, 1988, pp. 367-384.

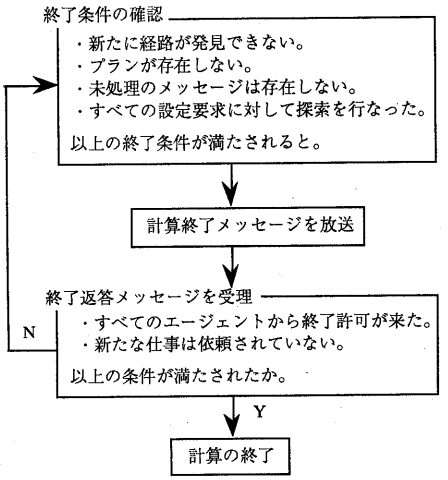

図１３　計算終了処理のフロー

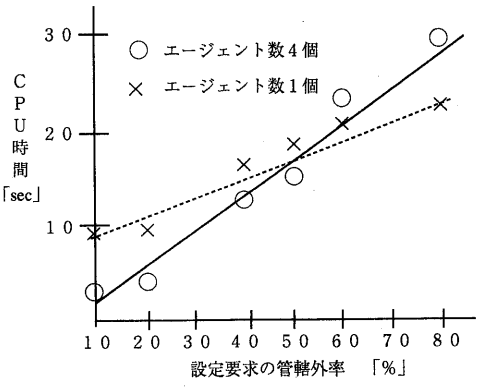

図１４　ＣＰＵ時間の比較