分散型問題解決における
推論方式と通信方式に関する研究

昭和 63 年 2 月

北 村 泰 彦

内容梗概

本論文は筆者が大阪大学大学院基礎工学研究科（物理系情報工学専攻）に在学中，情報基礎論講座ならびに産業科学研究所音響材料部門において行った分散型問題解決における推論方式と通信方式に関する研究をまとめたものである．

エキスパートシステムをはじめとする現在の人工知能システムは単一計算機でも実現可能な，比較的小規模な問題を対象としてきたが，記憶量や処理速度の制限のために将来予想される大規模な問題解決は難しくなってくる．そこで，複数の問題解決システムを結合し，互いの知識や解決能力を共有することによって，多量の知識処理と問題解決の分散並列化が可能な分散型問題解決が注目されている．分散型問題解決では，単独の解決器では解決できない問題を，複数の解決器が協力して解決するための推論方式と，解決器間で行われるメッセージのやり取りを規定する通信方式の研究が重要になる．

本論文の第１章では，分散型問題解決の工学的な意義と本研究の対象とする範囲を明らかにした後，その概要について述べる．

第２章では問題解決を，知識を表す状態空間グラフにおける初期状態から目標状態への経路探索と定式化している．さらに解決に必要な知識が各解決器にもれなく，重複なく分散しているという前提のもとで，複数の解決器が協力して経路探索を行う分散型探索アルゴリズムを提案している．

第３章では，分散型問題解決のための論理的な通信方式として回覧板方式を提案し，解決器への問題割り当てとその実行制御について述べる．回覧板方式では１対多，多対１の柔軟なグループ通信とともに，優先度付き通信により大局的な問題解決制御が可能になり，分散型探索のシミュレーションにより，その有効性を示している．

第４章では，分散型問題解決システム構築の試みとして，研究室内マイコンネットワークＡＩＤ－Ｎｅｔ上への実現について述べて

いる.

最後に第5章で全体のまとめと今後の課題, 展望について述べている.

関連発表論文

[1] 北村, 小川, 田村, 北橋: "優先度つきトークンパッシングプロトコルを用いた分散型探索機構", 人工知能学会誌, 1, 2, pp.254-258(1986-12).

[2] 北村, 小川, 北橋: "分散型問題解決における問題割り当てのための一通信方式", 電子情報通信学会論文誌(D), J71-D, 2(1988-2)(掲載予定).

[3] 北村, 横山, 小川, 田村: "マイコンネットワーク向き機械語変換方式について", 情報処理学会マイクロコンピュータ研究会資料, 32-3(1984-8).

[4] 北村, 小川, 田村: "マイコンネットワークを用いた分散型問題解決システム", 情報処理学会知識工学と人工知能研究会資料, 42-8(1985-9).

[5] 北村, 横山, 横山, 小川, 田村: "研究室内マイコンネットワークＡＩＤ－Ｎｅｔの作成", 情報処理学会関西ソフトウェア研究会資料(1986-1).

[6] 北村, 小川, 田村: "分散型問題解決システムのマイコンネットワーク上への試作", 電子通信学会技術研究報告, AI86-6(1986-4).

[7] 北村, 小川, 北橋: "分散型問題解決システムのための一通信方式", 情報処理学会知識工学と人工知能研究会資料, 52-9(1987-5).

分散型問題解決における
推論方式と通信方式に関する研究

目次

第 1 章　緒論

計算機によって人間の知能の代行をさせようという人工知能の研究が始まって３０年近くなる．その中で，専門知識を用いて助言するエキスパートシステムは扱う問題を限定すれば，すでに実用化の段階にまで至っている．しかし，より一般的な問題を扱うためには，解決するために必要な知識が非常に多くなり，多量の記憶容量と処理の高速化が必要になる．そこで，このような問題に対処するために，人工知能に分散処理の技法を応用した分散型問題解決の研究が進められている[1,2,3,4]．分散型問題解決によって，複数の問題解決システムの知識や解決能力を共有することが可能になり，多量の知識処理と解決の分散並列化が実現できる．また，解決すべき問題がもともと地理的に分散している場合にも有効な解決手法となる．このような分散型問題解決の応用例としては，地理的に分散したセンサからの情報をもとに総合的な状況判断を行うセンサネットワーク[6]や航空管制システム[7]，多量の知識を扱う分散型知識ベースシステム[8,9]，複数のロボットが相互干渉を回避しながら行動計画を立てる複数ロボット計画システム[10]，ＬＳＩ設計システム[11]などが報告されている．

分散型問題解決はその目的や応用により，用いる手法やアプローチが異なるが，本研究ではその範囲を「疎結合，複数のエージェント（解決器）による協調的な問題解決」[5]と限定している．疎結合とは解決器間が共有メモリや高速バスにより結合されているのではなく，コンピュータネットワークなど比較的低速の通信路により結合され，メッセージによって情報交換を行うことを意味している．また，協調的とは１台の問題解決器により解決できない問題が複数台の問題解決器の相互作用により解決できることを意味している．

従来の分散型問題解決に関する研究では，特定の問題に対する解決手法やアーキテクチャが中心に進められており，問題の持つ一般的な性質からその解法を求めるまでには至ってなかった．そこで第

2章においては，分散型問題解決の定式化を行い，協調的な推論方式である分散型探索アルゴリズムについて述べている．まず，問題解決の定式化として，知識を状態と，状態を変化させる規則からなる状態空間として表現し，問題として与えられた初期状態と目標状態に対して，初期状態を目標状態まで変化させる規則の系列（すなわち解）を求めることが問題解決であると定義している．すなわち問題解決は，状態空間を表すグラフにおける初期状態から目標状態に至る経路探索と等価になる．分散型問題解決では知識が各エージェントに分散して管理されるが，その編成形態に応じて以下の概念を導入する．分散型問題解決システムは，知識が各エージェントに重複なく，もれなく分散しているならば半編成システム，同等の知識を重複してエージェントに分散しているならば全編成システムと呼ぶ．このように知識を各エージェントに分散させる一般的な知識分割法は重要な問題であるが，本研究ではこの問題には触れず，分散されていることを前提として議論をすすめる．

次に，複数のエージェントが協調的に問題解決を行う半編成システムのための推論方式として，エージェントの局所的な探索とメッセージのやり取りを規定する二つの分散型探索アルゴリズムを提案する．単一解を求めるＤＰＳＡでは，各エージェントは探索可能な範囲で探索を進め，残りの部分を他のエージェントに依頼することにより，複数のエージェントの協力によって初期状態から目標状態に至る全体経路を求めている．しかし最適解を求める場合には，エージェントの情報，制御が独立しているので，解が得られた場合にそれが最適かどうかを確認する機能が必要である．そこでＤＯＰＳＡでは，DijkstraとScholtenの方法[13]により大局的終了性判定が可能なようにＤＰＳＡを拡張し，全エージェントの探索が終了した時点での最小コストの解を最適解としている．

また，それぞれのアルゴリズムに対して通信量の立場から考察もおこなった．通信量は分散システムのパフォーマンスのボトルネックになり，アルゴリズム評価の重要な要因となる．ここでの考察か

ら分散型探索においては問題の依頼の順序によって通信量が大きく変化し，大局的に重要な問題から先に解決するといった大局的問題解決制御の重要性が示される．

このような大局的問題解決制御は各エージェントの制御が独立している疎結合システムでは完全に実現することは難しい．しかし，優先度付き通信を用いて，重要な問題から先に依頼することができればそのパフォーマンスは大幅に改善されることが予想される．そこで第3章では，このような優先度付き通信を実現する論理的な通信方式として回覧板方式を提案し，エージェントへの問題割り当てと実行制御について述べる．回覧板方式は人間社会のグループ通信に用いられている回覧板をメタフォとしたもので，エージェントは論理的にリング状に結合され，送信エージェントから送出された主メッセージは一巡して自分自身に戻ってくるまで順次，エージェントに送られてゆく．主メッセージを受信した各エージェントは副メッセージを連結したり，追加するといった操作により送信エージェントとの1対多，多対1のグループ通信が実現できる．以上の回覧板方式により，知識の編成法や解の個数に応じた動的な問題割り当てと実行制御を実現している．優先度付き通信は主メッセージに優先度を示すタグを付け，それに応じてメッセージを送出することにより実現している．優先度付き通信の効用は分散型探索アルゴリズムDPSAのシミュレーションを行い，その全体的な探索時間が大幅に短縮されることにより確認できた．回覧板方式は従来の同報通信を基にした通信方式に比べ，通信中のメッセージに各エージェントが操作を加えることができ，より柔軟なグループ通信が実現できる．また，送出したメッセージが一巡して戻ってくるので，問題割り当てに関して受諾可能なエージェントが存在しない場合にもその判定が容易であるという特長がある．しかし，エージェントの数が多くなるとメッセージ一巡に時間がかかることが欠点としてあげられる．

第4章では分散型問題解決システムのマイコンネットワーク上へ

の実現について述べている．分散型問題解決に関する研究は歴史が浅く，理論的な考察と共に，実際にシステムを構築し，実用上の観点から考察を進めることも有益である．従来の分散型問題解決システムとして黒板システムをもとにした分散型解釈モデル[6]，Yangらのシステム[11]があげられるが，いずれもミニコンやワークステーションをエージェントとした大規模なシステムであった．そこで分散型問題解決に必要な機能を明確にし，さらに拡張が容易なように低コストの分散型問題解決システムをマイコンネットワーク上に試作した．本システムの特徴として以下のものがあげられる．

（１）分散型問題解決の基本的機能として，エージェントは問題の解決と共に，問題分割を行い，他のエージェントに依頼する協調的な問題解決を実現している．このような問題分割法として，ＡＮＤ／ＯＲ型，逐次／並列型の組合せの４種類が可能である．

（２）分散型問題解決システムに必要な機能が知識源，データ部，解決部，問題管理部，エージェント管理部，通信部の各部に明確にモジュール化されている．

（３）マイコンネットワーク上にシステムを構築することにより，低コストで分散型問題解決システムが実現できる．また，ネットワークのもつダイナミックダウンロード機能によりシステムの開発，保守が容易である．

最後に第５章で全体のまとめと今後の課題，展望について述べている．

第２章　分散型問題解決の定式化と探索アルゴリズム

第１節　序言

従来の分散型問題解決の研究においては，特定の問題に対する実現手法やアーキテクチャに関する研究が中心に進められており，問題の持つ一般的な性質からその解法や限界を調べるまでには至ってなかった．そこで本章ではまず，分散型問題解決の定式化をおこない，その推論方法となる二つの分散型探索アルゴリズムについて述べる．また，それぞれのアルゴリズムに対して通信量の観点からの考察も行っている．

第２節ではまず，問題解決の定式化について述べる．本研究における問題解決のモデルはBanerjiらのモデル(14)と同様に，知識を状態と，状態を変化させる規則からなる状態空間として表現し，問題として与えられた初期状態と目標状態に対して，初期状態を目標状態まで変化させる規則の系列（すなわち解）を求めることが問題解決であると定義している．すなわち問題解決は状態空間を表すグラフにおける初期状態から目標状態に至る経路探索と等価である．また，このような問題解決には単一解を求めればよい場合と，最適解を求める場合とがある．分散型問題解決では知識が各エージェントに分散して管理され，分散している知識はもとの知識の部分グラフとして表せるが，この形態として半編成システムと全編成システムの概念(8)を導入し，以下の議論を半編成システムに限って行っている．知識の分割に関する議論は重要ではあるが，知識はすでに分割されているとし，ここでは触れない．

分散型問題解決では，各エージェントの探索範囲は限られているので，それを越えるような問題に対しては，他のエージェントと協力して経路探索を行わなければならない．そこで，エージェントの局所的な探索と他のエージェントとのメッセージ交換を規定する分

散型探索アルゴリズムを提案した．第3節では半編成システムという前提のもとで，このような二つの分散型探索アルゴリズムを提案する．単一解を求めるＤＰＳＡでは，各エージェントは探索可能な範囲で探索を進め，残りの部分を他のエージェントに依頼することにより，複数のエージェントの協力によって初期状態から目標状態に至る全体経路を求めている．しかし最適解を求める場合には，エージェントの情報，制御が独立しているので，解が得られた場合にそれが最適かどうかを確認する機能が必要である．そこでＤＯＰＳＡでは，DijkstraとScholtenの方法[13]により大局的終了性判定が可能なようにＤＰＳＡを拡張し，全エージェントの探索が終了した時点での最小コストの解を最適解としている．

第4節ではそれぞれのアルゴリズムに対して通信量の立場から考察を行った．通信量は分散システムのパフォーマンスのボトルネックになり，アルゴリズム評価の重要な要因である．分散型探索においては問題依頼の順序によっては通信量が大きく変化し，より重要な問題から先に割り当てる大局的な問題解決制御が重要であることが示される．

第２節　定式化

２．１　準備

まず問題解決を定式化する前にグラフに関する用語を定義しておく．

グラフ(graph) $G = < N, E >$ は，空でない節点(node)の集合Nと，有向辺(edge)の集合 $E \subseteq N \times N$ で表される．$(n_i, n_j) \subseteq E$ の時，節点 n_i と節点 n_j は接続される(connect)といい，節点 n_i は節点 n_j の親(parent)，節点 n_j は節点 n_i の後継者(successor)という．ある節点の後継者の数を次数(degree)といい，全ての節点の次数は有限とする．N，Eの要素の数が有限のグラフを有限グラフ(finite graph)という．また，任意の相異なる二節点に対して，それらを接続する辺が存在するとき完全グラフ(complete graph)であるという．木(tree)は根節点(root node)と呼ばれる特定の節点を除いて，各節点が一つだけ親を持つグラフである．後継者のない節点のことを葉節点(leaf node)という．節点の深さ(depth)を根節点の深さを０，他の節点の深さを親節点の深さに１を加えたものとして定義する．

節点の列 $(n_0, \ldots, n_k)$ $(k \geq 1)$ は $\forall j (n_j, n_{j+1}) \in E$ の時に n_0 から n_k に至るの長さkの経路(path)と呼ぶ．節点 n_i から節点 n_j への経路が存在するとき，n_j は n_i から到達可能(accessible)であるという．このとき節点 n_i は節点 n_j の先祖(ancestor)，節点 n_j は節点 n_i の子孫(descendant)であるという．辺 (n_i, n_j) のそれぞれにコスト $c(n_i, n_j) > \delta$ (≥ 0) が割り当てられているとき，経路 $p = (n_0, \ldots, n_k)$ のコスト $c(p)$ は

$$c(p) = \sum_{j=0}^{k-1} c(n_j, n_{j+1})$$

で与えられる．

また，記法上の定義として，集合 $S_1, S_2, \ldots, S_n$ に対してn項関係 $R \subseteq S_1 \times S_2 \times \ldots \times S_n$ が定義されるときに，これを２項関係R

$\subseteq S_1 \times T$（ただし，$T = S_2 \times \ldots \times S_n$）と見なすことがある．また，$e_1 \in S_1$に対して，$R(e_1) = \{(e_2, \ldots, e_n) \mid (e_1, e_2, \ldots, e_n) \in R\}$と書くことがある．

2．2　問題解決と探索アルゴリズム

人工知能における従来の問題解決の枠組みとしてプロダクションシステム(production system)があり，データベース(database)，プロダクション規則(production rule)の集合，制御システム(control system)から構成されている．データベースには最初，初期状態を満たす事実の集合が記述され，目標状態を満たす事実の集合が得られるまで，制御システムが繰り返しプロダクション規則を適用し，データベースを変化させてゆく．このようにプロダクションシステムは問題解決に必要な静的な知識を事実の集合として，事実の関係として表される動的な知識を規則の集合として表している．プロダクションシステムが人工知能システムを構築する上で，基本的な枠組みとなりえたのは人間の問題解決において起こりがちな試行錯誤的な解決過程を計算機上にうまく実現できるからである．すなわち試行錯誤を制御システムにおける非決定的な規則の適用として表現できるという点である．ただしプロダクションシステムが正しい問題解決を行うためには規則の適用をでたらめに行うのではなく，順序だててておこなう必要がある．以上の議論を元に問題解決を定式化すると以下のようになる．

［定義2．1］　（知識）

知識(knowledge)はグラフ$K = < S, R >$により表される．ここでSは状態(state)の集合，$R \subseteq S \times S$は規則(rule)の集合である．規則(s_i, s_j)をr_{ij}と表すこともある．　□

［定義2．2］　（問題と解）

問題(problem)は組$< s_s, G >$で表される．ここでs_s($\in S$)は初期状態(start state)，G($\subseteq S$)は目標状態(goal state)の集合である．ただし，$s_s \notin G$とする．また問題のグラフにおいて出発状態s_sから目標状態$s_g \in G$に至る経路を解(solution)と呼ぶ．□

一般に解は複数個存在することもあるが，規則r_{ij}のそれぞれにコスト$c(r_{ij}) \geq \delta$ (> 0)が与えられる場合には最適解が定義される．

［定義２．３］　（解のコストと最適解）

解solのコストはその経路のコストc(sol)として定義する．解となる経路solが存在するときに，c(sol)よりも小さいコストの解が存在しなければ，solは最適解(optimal solution)であるという．□

与えられた知識と問題に対してその解を求めることが問題解決であり，その過程は問題により与えられる初期状態から目標状態までの経路探索と見なすことができる．経路探索に関するいくつかの用語を定義しておく．ある状態s_iから$(s_i, s_j) \in R$となる全ての後継者状態s_jを求めることを状態s_iを展開する(expand)という．また，探索アルゴリズムの性質として完全性と適格性の定義を行う．

［定義２．４］　（完全性と適格性）

ある探索アルゴリズムが解の存在するグラフに対して，必ず解を得て終了するならば，そのアルゴリズムは完全(complete)であるという．また，その時最適解を得て終了するならば適格(admissible)であるという．□

ここで従来の探索手法のまとめをおこなう．単一解を求める経路探索アルゴリズムＰＳＡについて述べる．ＰＳＡで用いる集合，関数には以下のものがある．

ＯＰ：展開の候補となる状態の集合で開状態(open state)集合と

呼ぶ．要素は状態s_i，親状態s_pの組(s_i, s_p)で表す．

ＣＬ：すでに展開を終えた状態の集合で閉状態(closed state)集合と呼ぶ．要素は状態s_i，親状態s_pの組(s_i, s_p)で表す．

SF：開状態集合から次に展開する状態を選び出す関数$2^{OP} \rightarrow OP$で，戦略関数(strategic function)と呼ぶ．ＯＰが空ならばϕを返すとする．

探索は展開の繰り返しであり，以下の手続きとして表される．

［経路探索アルゴリズムＰＳＡ］

0. ［初期化］
   ＯＰ$\leftarrow \{(s_s, \phi)\}$．　ＣＬ$\leftarrow \phi$．
1. ［選択］
   $(s_i, s_p) \leftarrow SF(OP)$．
2. ［失敗］
   if $s_i = \phi$　then exit(fail).
3. ［成功］
   if $s_i \in G$　then exit(success).
4. ＯＰ$\leftarrow$ＯＰ$- \{(s_i, s_p)\}$．
5. ＣＬ$\leftarrow$ＣＬ$+ \{(s_i, s_p)\}$．
6. ［展開］
   for all $s_j \in R(s_i)$
     if $s_j \notin \{s \mid (s, s') \in CL \cup OP\}$　then
       ＯＰ$\leftarrow$ＯＰ$+ \{(s_j, s_i)\}$．
7. 1.へ．

□

ＰＳＡでは戦略関数SFにかかわらず，次の定理が成立する．

［定理２．１］
有限グラフに対してアルゴリズムＰＳＡは完全である．
（証明）

解が存在するときに，ステップ３の成功で終了しなかった場合を考える．その時は（１）ステップ２の失敗で終了したか，（２）終了しないかのいずれかである．

（１）の場合：解となる経路$(s_0,\ldots,s_n)$（ただし，$s_0=s_s, s_n\in G$）に対して，経路上の状態をs_nから逆にたどって$(s_k,*)\in CL$となる最初の状態s_kを求める．s_0は最初にステップ５で$(s_0,\phi)\in CL$となり，$(s_n,*)\notin CL$だからs_kは必ず存在する．また，$(s_{k+1},*)\notin CL$である．$(s_k,*)\in CL$からs_kはステップ６で必ず展開され，$(s_k,s_{k+1})\in R$となる状態s_{k+1}は$(s_{k+1},s_k)\in OP$となる．失敗で終了すると$OP=\phi$だから，s_{k+1}は展開されて$(s_{k+1},s_k)\in CL$となり矛盾する．

（２）の場合：いかなる状態もステップ６より二度展開されることはない．状態の数は有限なのでいつかは$OP=\phi$となり，必ず終了する．

ＰＳＡが成功で終了すればＯＰ，ＣＬの要素である親状態をたどってゆくことにより解となる経路が得られる．□

次に最適解を得る探索アルゴリズムＯＰＳＡを示す．ここでは，ＯＰ，ＣＬの要素を状態s_i，親状態s_p，評価値$f(s_i)$の三組$(s_i,s_p,f(s_i))$に拡張する．ここでの評価値$f(s_i)$はその時点で得られている初期状態s_sから状態s_iに至る経路のコストの中で最小のものである．また，戦略関数は以下のものを用いる．

［定義２．５］（最短優先戦略SFs）

$(s,sp,f(s))\leftarrow SFs(OP)$

ただし，$f(s)=\min_{(s',sp',f(s'))\in OP} f(s')$　である．□

最適解を求める手続きを以下に示す．

［最適経路探索アルゴリズムＯＰＳＡ］

0.　［初期化］

　　ＯＰ←｛(s_s, ϕ, 0)｝.　ＣＬ←ϕ.

1.　［選択］

　　(s_i, s_p, $f(s_i)$)←SFs(ＯＰ).

2.　［失敗］

　　if $s_i = \phi$　then exit(fail).

3.　［成功］

　　if $s_i \in G$　then exit(success).

4.　　ＯＰ←ＯＰ－｛(s_i, s_p, $f(s_i)$)｝.

5.　　ＣＬ←ＣＬ＋｛(s_i, s_p, $f(s_i)$)｝.

6.　［展開］

　　for all $s_j \in R(s_i)$

6.1　　$f(s_j)$←$f(s_i)$+$c(r_{ij})$

6.2　　if ＯＰ(s_j)$= \phi$　then

　　ＯＰ←ＯＰ＋｛(s_j, s_i, $f(s_j)$)｝.

6.3　　if $s_j = s_j' \wedge (s_j', s_p', f(s_j')) \in$ ＯＰ　then

　　if $f(s_j) < f(s_j')$ then

　　ＯＰ←ＯＰ－｛(s_j', s_p', $f(s_j')$)｝

　　＋｛(s_j, s_i, $f(s_j)$)｝.

7.　　1.へ.　□

ＯＰＳＡはＰＳＡと同様に定理2．1が成り立つ．ＯＰＳＡの適格性を証明するために以下の補題を証明する．

［補題2．1］

ＯＰの最小評価値は探索が進んでも非減少である．

（証明）

ＯＰに状態s_jが加わるのはステップ6．2または6．3であるが，その状態の評価値$f(s_j)$は展開した状態s_iの評価値$f(s_i)$に規則r_{ij}の

コストc(r_{ij})＞0を加えたものであり，f(s_i)＜f(s_j)である．従って，ＯＰの最小評価値は非減少である．□

［系２．１］

ＯＰＳＡが状態sを展開した時点で，すでにsまでの最小コストの経路が得られている．

（証明）戦略関数SFsより，ＯＰＳＡでは展開する状態としてＯＰの中で最も評価値の低い状態が選ばれる．補題２．１より状態sの展開後にさらにコストの小さいsまでの経路が得られることはない．□

［補題２．２］

解が存在すればＯＰＳＡは終了する．

（証明）グラフが有限の場合は定理２．１より終了するので，無限グラフの場合を考える．補題２．１より，ＯＰＳＡの実行が進むと，各状態の次数は有限なので，ＯＰの最小評価値は任意の値より大きくなる．解の経路を$p=(s_0, s_1, \ldots, s_n)$とし，ＯＰＳＡが終了しないとすると，ＯＰの最小評価値Mはいずれ解のコストc(p)より大きくなる．しかし，終了していないのでs_0からs_nの中でＯＰに含まれる状態s_kが存在し，f(s_k)≦c(p)である．f(s_k)＜Mとなり，MがＯＰの最小評価値であることと矛盾する．従って，ＯＰＳＡは終了する．□

従って，次の定理が成立する．

［定理２．２］

アルゴリズムＯＰＳＡは適格である．

（証明）

定理２．１と補題２．２より，ＯＰＳＡは解を得て終了する．また，系２．１よりその解は最適解である．□

以上のように問題解決は状態空間における探索として定式化できるが，探索が進むにつれ計算量が急激に増加する組合せ的爆発(combinatorial explosion)が生じる．このような場合には発見的な(heuristic)知識を用いることにより，探索量を減らし，許容可能な時間内に問題を解決することができる．発見的な知識は状態sに対して状態sから目標状態に至る経路のコストの予測h(s)であるといえる．（実際のコストを$h^*(s)$で表す．）そしてその時点で得られている初期状態から状態sに至るコスト最小の経路のコストをg(s)（ＯＰＳＡでの評価値）とし，戦略関数SFsにおける評価値をf(s)＝g(s)+h(s)で与えたＯＰＳＡのことをＡアルゴリズムと呼んでいる．また，$h(s) \leqq h^*(s)$であるならば特にＡ*アルゴリズムと呼ばれ，その適格性が証明されている[17]．ここで，発見的知識が全くなければ（すなわち，h(s)＝0ならば），ＡアルゴリズムはＯＰＳＡと同等になる．また，発見的知識が完全であれば（すなわち，$h(s)=h^*(s)$ならば）探索は解の経路に沿った直線的なものになる．これは解決の過程があらかじめ与えられた決定的な解法であるといえる．人工知能における問題解決は$0<h(s)<h^*(s)$であるような発見的知識を扱うものであり，ＯＰＳＡはその基本的な探索アルゴリズムといえる．

2．3　知識の編成形態

分散型問題解決では知識は複数のエージェントに分散して管理され，それらの協調的な動作により問題解決が行われる．このような問題解決に必要な知識には各エージェント毎に局所的知識と大局的知識がある．局所的知識は自分自身の問題解決に関する知識であり，大局的知識は協調的な問題解決を行うために必要な他のエージェントの解決能力に関する知識である．分散型問題解決システムを構成するエージェントの集合をA，エージェントをa,b,...で表し，これらの定式化を以下に示す．

［定義２．６］　（局所的知識）

エージェントaの局所的知識(local knowledge)をLK(a)＝＜S(a), R(a)＞で表す．S(a)はエージェントaが持つ（局所的）状態の集合，R(a)（$\subseteq S(a)\times S(a)$）はエージェントaが持つ（局所的）規則の集合である．また，以下の条件を満たす状態sの集合をエージェントaの受諾可能状態(acceptable state)集合AS(a)と呼ぶ．

$$s\in S(a)\wedge \exists b\,(s\in S(b)\wedge a\neq b)$$ □

［定義２．７］　（大局的知識）

エージェントaの大局的知識(global knowledge)をGK(a)（$\subseteq AS(a)\times A$）で表す． □

すなわち，知識を表すグラフの部分グラフが局所的知識であり，他のエージェントの局所的知識とのつながりを表したものが大局的知識である．分散している知識にはいくつかの形態が考えられるが，その編成形態として以下の概念を導入する．

［定義２．８］　（半編成システム）

知識＜S*, R*＞に対して以下の条件を満たすならば，半編成システムであるという．

（１）$S^*=\bigcup_{a\in A} S(a)\wedge R^*=\bigcup_{a\in A} R(a)$

（２）$\forall a,b\in A\ (a\neq b\rightarrow R(a)\cap R(b)=\phi)$

（３）$(\forall a\in A)\ GK(a)\subseteq\{(s_i,b)\mid (s_i,s_j)\notin R(a),\ (s_i,s_j)\in R(b),\ b\in A\}$ □

条件（１）により局所的知識の和によりもとの知識が復元可能であること，（２）により局所的知識に重複がないことを示している．（３）により自分自身の局所的知識と他のエージェントの局所的知識との接続関係に関する必要な情報は全て大局的知識に含まれるこ

とを示している．半編成システムの例を図２．１に示す．

［定義２．９］　（全編成システム）

知識＜S*，R*＞に対して以下の条件を満たすならば，全編成システムであるという．

（１）$S^* = \bigcup_{a \in A} S(a) \wedge R^* = \bigcup_{a \in A} R(a)$

（２）$\forall a, b \in A \ (R(a) \cap R(b) = \phi \vee R(a) = R(b))$

（３）$\forall s \in S, \ \exists a \in A, \ (R^*(s) \subseteq R(a, s))$

（４）$(\forall a \in A) \ GK(a) \subseteq \{(s_i, b) \mid (s_i, s_j) \notin R(a), (s_i, s_j) \in R(b), \ b \in A\}$　□

条件（１）と（４）は半編成システムと同じで，（２）により同等の能力を持つエージェントが一つ以上存在することを示している．また，（３）によりいかなる状態も一つのエージェントだけで展開が可能である（すなわち全ての後継者が得られる）ことを示している．全編成システムの例を図２．２に示す．

すなわち，半編成システムはある状態に対してそれを展開に必要な規則が複数のエージェントに分散している場合で，その状態の展開には規則が分散している全てのエージェントの関与が不可欠となる．全編成システムはある状態に対して，展開に必要な全ての規則が複数のエージェントに同じように分散している場合で，単一のエージェントでも解決可能であるが，その負荷を減らすために複数のエージェントが解決に関与する．半編成システムは機能分散システム，全編成システムは負荷分散システムにそれぞれ対応させることができる．

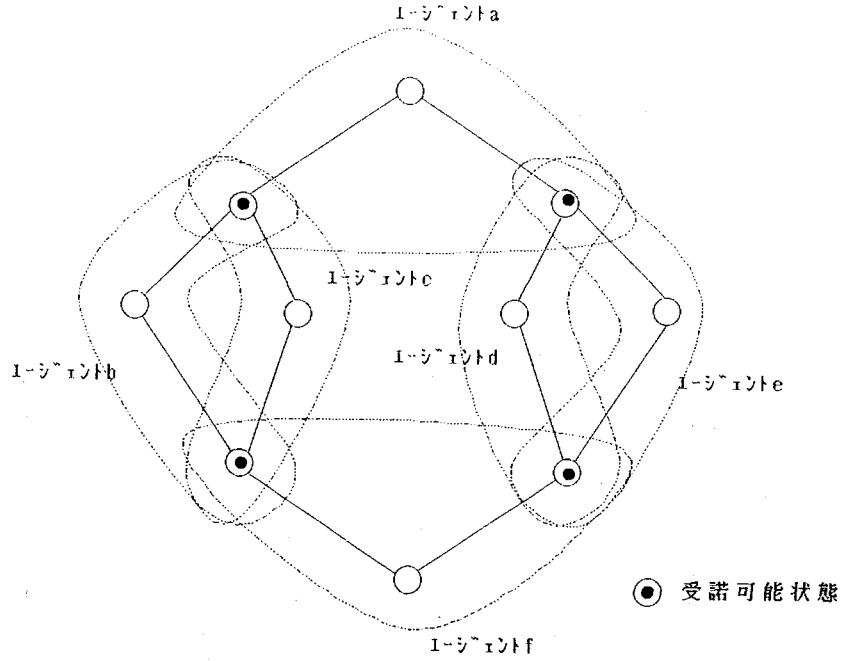


図２．１　半編成システム

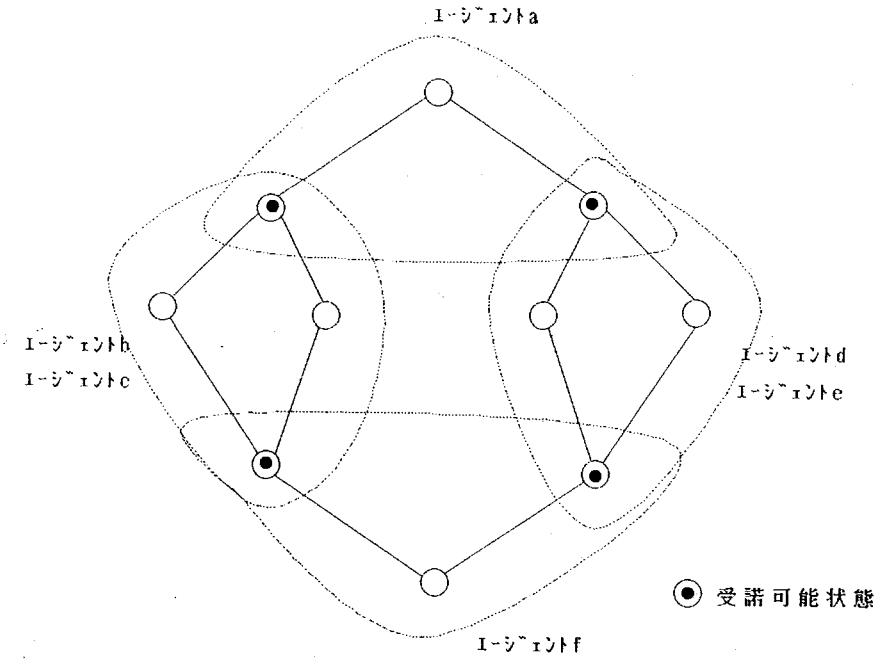


図２．２　全編成システム

第３節　分散型探索アルゴリズム

３．１　エージェントと通信システム

分散型探索では各エージェントが探索できる範囲は局所的知識内に制限されており，局所的知識を越えるような問題に対する解は各エージェントで得られる部分解を統合することにより得なければならない．そこで分散型探索アルゴリズムは，従来の探索アルゴリズムが規定していたエージェントが局所的知識を用いて行う局所的探索に加えて，協調的な探索を行うためのエージェント間のメッセージのやり取りと局所的探索との関係を規定している．そこで分散型探索アルゴリズムについて述べる前に，エージェントの通信能力について，いくつかの仮定を設ける[18]．

［定義２．１０］　（分散型問題解決システム）
分散型問題解決システムはグラフ＜Ａ，Ｌ＞で定義される．Ａはエージェントの集合，Ｌ⊆Ａ×Ａは通信リンクの集合である．グラフ＜Ａ，Ｌ＞は完全グラフであるとする．　□

エージェント間の通信に関して以下の仮定をおく．

［仮定２．１］
通信リンク(communication link)(a,b)は無限長のＦＩＦＯ(first-in first-out)のキューで，エージェントaがメッセージを書き込み，エージェントbは書き込まれた順にメッセージを読み出すことにより通信が行われる．　□

［仮定２．２］
エージェント間には共有メモリは存在せず，通信リンクを介したメッセージのみによって情報交換を行う．　□

［仮定２．３］

エージェントa∈Aは以下の通信命令を用いることができる．

（１）send(b,m)：通信リンク(a,b)にメッセージmを書き込む．

（２）broadcast(m)：∀b∈Aにメッセージmを送る．これは，エージェントの数だけのsend命令が実行されるのと同等である．

（３）receive(A,M)：A,Mは変数で，{(b,a)∈L｜b∈A}から一つの通信リンク(b',a)が適当に選ばれ，Aにb'が，Mに通信リンクの先頭のメッセージまたはφが代入される．通信リンクの先頭のメッセージが代入された時は，それは通信リンクから除かれ，φが代入されたときは通信リンクは変化しない．通信リンクが空の時は常にφが代入される．□

［仮定２．４］

エージェントa∈Aがreceive(A,M)を繰り返し実行すれば，任意の通信リンク(b,a)のメッセージはいつか読み込まれる．すなわち，メッセージは通信中に消失することがない．□

［仮定２．５］

エージェントa∈Aがreceive(A,M)を実行し，通信リンク(b,a)が選ばれた時に，(b,a)が空でないのにもかかわらず変数Mにφが代入されることがある．すなわち，通信遅延は前もってわからない．□

３．２　分散型経路探索アルゴリズムＤＰＳＡ

２．２で述べた経路探索アルゴリズムＰＳＡに対応する分散型経路探索アルゴリズムＤＰＳＡを提案する．分散型探索は最初，初期状態を局所的知識にもつエージェントの局所的探索から始められる．このエージェントのことを根エージェント(root agent)と呼ぶ．局所的探索は（局所的）初期状態から展開が開始され，局所的知識の

中で展開が進められ，目標状態まで到達するか，開状態集合が空になるまで続けられる．その途中で展開して得られた状態sに対して大局的知識(s,a)が存在すれば，探索依頼メッセージをエージェントaに送る．メッセージを受け取ったエージェントも同様に状態sを局所的初期状態として展開を開始する．探索依頼メッセージを送ることを状態sを依頼する(request)といい，状態sを依頼状態という．また，探索依頼メッセージを受け取ることを受諾する(accept)といい，その状態を受諾状態という．さらに探索依頼メッセージを送った方を依頼エージェント，受け取った方を受諾エージェントという．大局的探索は図２．３で示されるように探索依頼メッセージによる局所的探索のつながりを示す木として表され，根エージェントにおける局所的探索が根となる．

このような探索を行うアルゴリズムを示す．まず，メッセージとして以下のものを用いる．メッセージの一般形は<label;parameters>である．

（１）探索依頼メッセージ<search;s>：状態sからの探索依頼に用いる．

（２）解到達メッセージ<solution>：解を得たことを示すために用いる．

また，エージェント毎に集合として以下のものを用いる．

ＯＰ(a)，ＣＬ(a)：ＰＳＡで定義した開状態集合，閉状態集合．

Ｆ(a)：受諾状態sと依頼エージェントaを組(s,a)で記録しておく集合．

Ｃ(a)：依頼状態sと受諾エージェントaを組(s,a)で記録しておく集合．

アルゴリズムは通信処理と探索処理を交互に行う．以下にアルゴリズムを示す．（エージェントを表す添え字は省略してある．）

［分散型探索アルゴリズムＤＰＳＡ］

0.　［初期化（一般エージェント）］

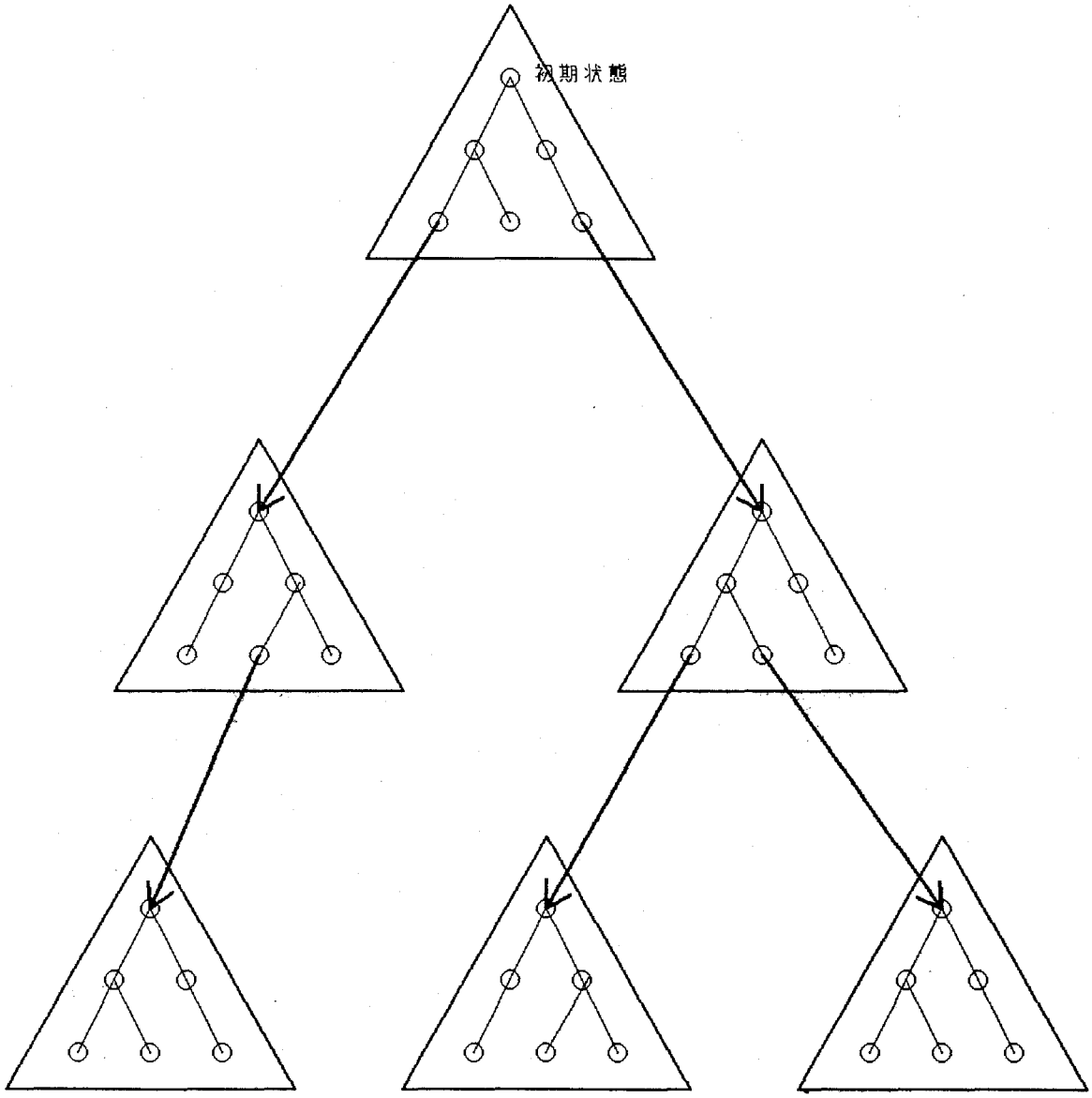


図２．３　局所的探索木と大局的探索木

```
      OP ← φ.  CL ← φ.  F ← φ.  C ← φ.
0'.  [初期化（根エージェント）]
      OP ← {(s_s, φ)}.  CL ← φ.  F ← φ.  C ← φ.
1.   [受信]
      receive(A, M).
1.1  [探索受諾]
      if M=<search; s_k> then
        if s_k ∉ {s | (s, s') ∈ CL ∪ OP} then
          OP ← OP + {(s_k, φ)}.  F ← F + {(s_k, A)}.
1.2  [解到達]
      if M=<solution> then exit.
2.1  [選択]
      (s_i, s_p) ← SF(OP).
2.2   if s_i = φ  then wait.
2.3  [成功]
      if s_i ∈ G  then broadcast(<solution>).  exit.
2.4   OP ← OP − {(s_i, s_p)}.
2.5   CL ← CL + {(s_i, s_p)}.
2.6  [展開]
      for all s_j ∈ R(s_i)
        if s_j ∉ {s | (s, s') ∈ CL ∪ OP} then
2.6.1     OP ← OP + {(s_j, s_i)}.
2.6.2 [依頼]
          for all b ∈ LK(a, s_j)
            send(b, <search; s_j>).  C ← C + {(s_j, b)}.
3.     1.へ.                                              □
```

いずれかのエージェントが目標状態に到達すると，解到達メッセージが全てのエージェントに送られ，受け取ったエージェントは終了する．解はOP，CLの要素である親状態，Fの要素である依頼

エージェントをたどることにより得られる.

ＤＰＳＡは以下で定義する大局的開状態集合が空となったとき，終了とする．（大局的開状態集合が空になれば，全てのエージェントで展開，受諾が行われない．）

［定義２．１１］（大局的開状態集合）

大局的開状態集合は各エージェントの開状態集合と依頼中の状態（依頼エージェントの依頼状態集合Ｃに含まれるが，受諾エージェントの受理状態集合Ｆには含まれていない状態）の和として定義される．□

［定理２．３］　ＤＰＳＡは有限グラフに関して完全である．

（証明）

大局的開状態を開状態と見なすことにより，定理２．１と同様に証明できる．□

３．３　分散型最適経路探索アルゴリズムＤＯＰＳＡ

最適解を求める分散型最適経路探索アルゴリズムＤＯＰＳＡについて述べる．ＤＯＰＳＡはＯＰＳＡに比べると，以下のような違いがあり，それに応じてＤＰＳＡを拡張した．

（１）閉状態の再展開

ＯＰＳＡでは展開する時点ではその状態までの最小コストの経路が確定していたので，閉状態から開状態に変化することはなかった．しかし，ＤＯＰＳＡでは図２．４に示されるように，探索依頼メッセージ（依頼１）を受け取った後に，その評価値よりも低い評価値の探索依頼メッセージ（依頼２）を受け取る場合がある．そこで，すでに閉状態にある状態を再び展開する必要がある．

注意：ここで再展開とはある状態を閉状態から開状態に変更し，もう一度展開し直すことであるが，その他の方法としてすでに得ら

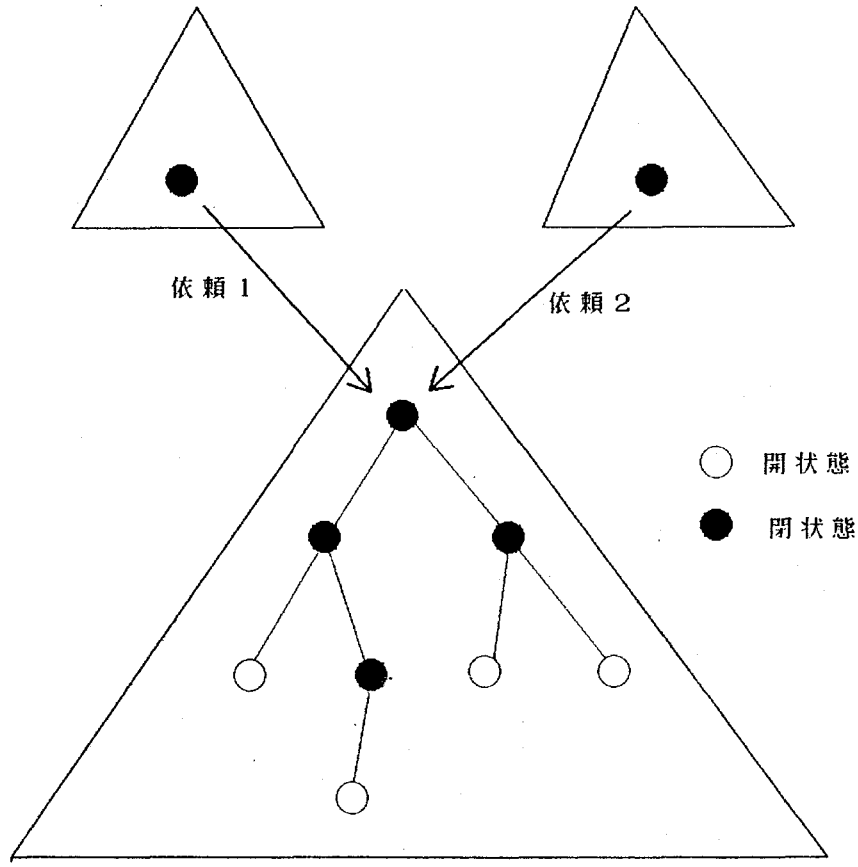


図２．４　閉状態の再展開

れている閉状態や開状態の評価値を探索木にそって変更する方法も考えられる．どちらの方法が適しているかは問題に応じて異なると予想されるが，ここでは前者を再展開の方法として用いる．

（２）最適解の確定

分散型探索では各エージェントがそれぞれ独立に探索を行うので，解が得られてもそれが最適解かどうか確認しなければならない．そのための機能として以下のものを加えた．

（ａ）大局的枝刈り

目標状態を局所的知識に含まず，無限グラフを扱うエージェントでは探索が終了しない．そこで，コストｃの解が得られたエージェントはそのことを解到達メッセージを用いて全てのエージェントに知らせ，受け取ったエージェントはそのコストを用いて探索の枝刈りを行う．コストｃよりも小さい解を得る可能性のあるエージェントは展開を続け，解の可能性のある経路が全てｃよりも大きくなると終了する．このことにより有限グラフでない場合でも必ず全てのエージェントは終了する．

（ｂ）大局的終了判定

全てのエージェントが終了した時点で最小コストの解が最適解であるが，全てのエージェントが終了したことを判定するためにDijkstraとScholtenの方法[13]を用いた．各エージェントは目標状態に到達するか，すべての探索経路が行き詰まってしまうことにより探索を終了する．このとき局所的探索は他のエージェントに探索の依頼中である場合の「継続」，それ以外の場合の「未継続」のいずれかで終了している．すなわち探索が終了すると大局的探索木の葉は「未継続」で，その他は「継続」でラベル付けできる．そこで，「未継続」で終了した局所的探索は依頼エージェントに探索終了メッセージを送り，「継続」で終了した局所的探索は全ての依頼に対して探索終了メッセージを受け取ると，さらに探索終了メッセージをその依頼エージェントに送るようにする．すなわち，探索終了メッセージを大局的探索木の葉から根に向かって送るようにすれば，根

エージェントが全ての依頼に対する探索終了メッセージを受け取った時点で，全てのエージェントが終了していることを判定できる．

［補題２．３］

エコーアルゴリズムにより，根エージェントが全ての依頼に対する探索終了メッセージを受け取った時点で，全てのエージェントの局所的探索が終了している．

（証明）

文献(13)参照． □

（３）複数の局所的探索の処理

一つのエージェントで複数の依頼を受諾した場合には，大局的終了判定を行うためにそれぞれの依頼に対して探索終了メッセージを送る必要がある．複数の依頼を逐次的に処理すると，グラフが有限でない場合は探索が終了しないことがあるので，以下のように複数の探索を並列的に扱っている．

（ａ）複数の探索依頼を別々の局所的探索として扱う．

（ｂ）複数の探索を同時に扱うために，全ての局所的探索の中で最もコストの低い開状態から先に展開する．

（ｃ）無駄な探索を減らすために，各状態の評価値を共通に利用して局所的な枝刈りを行う．

以下，最適解を求めるアルゴリズムを示す．エージェント毎の集合を以下のようにを拡張，追加する．

ＯＰ(a)，ＣＬ(a)：開状態，閉状態はそれぞれ，局所的探索識別子t，状態s，親状態s_p，評価値f(s)の組(t, s, s_p, f(s))で表す．

Ｆ(a)：受諾状態集合で，局所的探索識別子t，受諾状態s，依頼エージェントa_p，受諾状態の評価値f(s)の組(t, s, a_p, f(s))を要素とする．

Ｃ(a)：依頼状態集合で，局所的探索識別子t，依頼状態s，受諾エージェントa_s，依頼状態の評価値f(s)の組(t, s, a_s, f(s))を要素とす

る.

局所的探索はその根状態を識別子として用いる. また, 依頼エージェントはエージェント名aと局所的探索識別子tの組(a,t)で表す.

また変数, 関数として以下のものを用いる.

GCUT: 大局的枝刈り用変数(初期値$+\infty$).

LCUT(s): 局所的枝刈りに用いる関数で, 全ての局所的探索のOPまたはCL中で最小コストの状態sを返す関数. すなわち,

$$LCUT(s) = \min_{(t',s',p',c') \in OP \cup CL \wedge s' = s} c'$$

(s=s'となる$(t',s',p',c') \in OP \cup CL$が存在するとき)

$= +\infty$

(それ以外の時)

SFsd: 戦略関数は以下のように定義する.

$(t,s,sp,f(s)) \leftarrow SFsd(OP)$

ただし, $f(s) = \min_{(t',s',sp',f(s')) \in OP} f(s')$

である. すなわち, 全ての局所的探索におけるOPの中で最小の評価値の要素を返す関数である.

DOPSAでは以下のメッセージを用いる.

(1) 探索依頼メッセージ<search;t,s,f(s)>: 局所的探索tの状態s(評価値f(s))からの探索依頼に用いる.

(2) 探索終了メッセージ<echo;r,s,c>: <search;r,s,c>に対応する探索終了を示すために用いる.

(3) 解到達メッセージ<solution;c>: コストcの解を得たことを示すために用いる.

(4) 終了メッセージ<halt>: 全てのエージェントを終了させるために用いる.

[分散型最適経路探索アルゴリズムDOPSA]

0. [初期化(一般エージェント)]

```
        OP ← φ.   CL ← φ.   F ← φ.   C ← φ.
1.    [受信]
        receive(A, M).
1.1   [探索受諾]
        if M = <search; t_k, s_k, f(s_k)> then
1.1.1     if min(GCUT, LCUT(s_k)) > f(s_k) then
1.1.1.1     ∀(s_k, (b, t_k'), f'(s_k)) ∈ F(s_k)に対し,
              send(b, <echo; t_k', s_k, f'(s_k)>).
1.1.1.2     OP(s_k) ← {(s_k, φ, f(s_k))}.
1.1.1.3     F(s_k) ← {(s_k, (A, t_k), f(s_k))}.
1.1.1.4     C(s_k) ← φ.
1.1.2     if min(GCUT, LCUT(s_k)) ≦ f(s_k) then
            send(A, <echo; t_k, s_k, f(s_k)>).
1.2   [探索終了]
          if M = <echo; t, s, f(s)> then
            if ∃(s, A, f(s)) ∈ C(t) then
1.2.1         C(t) ← C(t) − {(s, A, c)}.
1.2.2         if C(t) = φ ∧ OP(t) = φ then
                for all (s', (b, t'), f(s')) ∈ F(t)
                  send(b, <echo; t', s', f(s')>).
1.3   [解到達]
        if M = <solution; f> then
          if GCUT > f then GCUT ← f.
1.3   [終了]
        if M = <halt> then exit.
2.1   [選択]
        (t, s_i, s_p, f(s_i)) ← SFsd(OP).
2.2     if s_i = φ   then wait.
2.3   [成功]
        if s_i ∈ G   then
```

broadcast(<solution;$f(s_i)$>).

$GCUT \leftarrow f(s_i)$. $OP(t) \leftarrow \phi$. 2.7へ.

2.4 $OP(t) \leftarrow OP(t) - \{(s_i, s_p, f(s_i))\}$.

2.5 $CL(t) \leftarrow CL(t) + \{(s_i, s_p, f(s_i))\}$.

2.6 [展開]

for all $s_j \in R(s_i)$

2.6.1 $f(s_j) \leftarrow f(s_i) + c(r_{ij})$.

2.6.2 if $\min(GCUT, LCUT(s_j)) > f(s_j)$ then

2.6.2.1 for all $(s_j', s_p', f(s_j')) \in CL(t)$

if $s_j = s_j'$ then

$CL(t) \leftarrow CL(t) - \{(s_j', s_p', f(s_j'))\}$.

2.6.2.2 for all $(s_j', s_p', f(s_j')) \in OP(t)$

if $s_j = s_j'$ then

$OP(t) \leftarrow OP(t) - \{(s_j', s_p', f(s_j'))\}$.

2.6.2.3 $OP(t) \leftarrow OP(t) + \{(s_j, s_i, f(s_j))\}$.

2.6.2.4 for all $b \in LK(a, s_j)$

2.6.2.4.1 send(b, <search;t, s_j, $f(s_j)$>).

2.6.2.4.2 $C(t) \leftarrow C(t) + \{(s_j, b, f(s_j))\}$.

2.7 if $OP(t) = \phi \wedge C(t) = \phi$ then

for all $(t, (b, t'), f(t)) \in F(t)$

send(b, <echo;t', t, $f(t)$>).

3. 1.へ.

ただし, 根エージェントにおいて, ステップ0, 1.2.2, 2.7は以下の手続きとする.

0. [初期化(根エージェント)]

$OP(s_s) \leftarrow \{(s_s, \phi, 0)\}$. $CL \leftarrow \phi$. $F \leftarrow \phi$. $C \leftarrow \phi$.

1.2.2 と 2.7

if $OP(t) = \phi \wedge C(t) = \phi$ then

if $t = s_s$ then

broadcast(<halt>). exit.

if $t \neq s_s$ then

$\forall (t,(b,t'),f(t)) \in F(t)$に対し,

send(b,<echo;t',t,f(t)>). □

ＤＯＰＳＡでは根エージェントにおいて初期状態を識別子とする局所的探索が終了する（開状態集合と依頼状態集合が空になる）と，終了メッセージを全てのエージェントに送り，終了し，受け取ったエージェントも終了する．本アルゴリズムにより必ず最適解が展開されること，またアルゴリズムが終了することを証明しておく．まず，大局的開状態集合に関しては以下の補題が成立する．

［補題２．４］

大局的開状態集合の最小評価値は非減少である．

（証明）

ＤＯＰＳＡにおいて大局的開状態集合の最小評価値が減少する可能性のある操作が存在するのは，（１）開状態s_iを展開することにより得られた状態s_jが開状態に加えられる場合（ステップ2.6.2.3），（２）開状態が依頼中の状態に加えられる場合（ステップ2.6.2.4.2），（３）依頼中の状態が開状態に加えられる場合(ステップ1.1.1.1)のいずれかである．（１）では状態sjの評価値$f(s_j)$を$f(s_j)=f(s_i)+c(r_{ij})$として計算し，$c(r_{ij})>0$なので大局的開状態の最小評価値は非減少である．（２），（３）ではともに評価値は変化しないので非減少である．ゆえに，大局的開状態集合の最小評価値は非減少である．□

［補題２．５］

各エージェントで探索依頼される状態の評価値，展開される状態の評価値は大局的開状態の最小評価値よりも小さくはない．

（証明）

探索の依頼は依頼エージェントの開状態が依頼中の状態を経由して受諾エージェントの開状態に加えられることであり，大局的開状態集合の最小評価値以下の状態の探索依頼は行われない．展開される状態の評価値が大局的開状態の最小評価値よりも小さくないことも定義よりあきらか．　□

次に以下の定理が証明される．

［定理２．４］

ＤＯＰＳＡにより最適解は展開される．

（証明）

最適解である状態の系列を（$s_0, s_1, \ldots, s_n$）とする．これを部分系列$p_0, p_1, \ldots, p_m$（$m \geq 0$）に分割する．ただし，$p_i = (s_{i0}, s_{i1}, \ldots, s_{in_i})$とすると，以下の条件を満たしている．

（ａ）$\exists a \in A\ (\forall j\ (0 \leq j \leq n_i - 1)\ (s_{ij}, s_{ij+1}) \in R(a))$

（ｂ）$s_{00} = s_0$，$s_{mn_m} = s_n$，$\forall k\ (1 \leq k \leq m-1)\ s_{kn_k} = s_{k+1,0}$

以下，p_k（$0 \leq k \leq m$）が得られることを帰納的に証明する．

k＝0の場合：p_0が得られなかったとすると，p_0の中でＣＬに含まれない状態が存在する．（初期化によりs_{00}（$= s_0$）はＯＰに含まれ，最初の展開でＣＬに含まれる．）ｐのうちでs_{00}から始めて最後にＣＬに含まれる状態をs_{0p-1}とする．p_0が得られない理由としては，（１）無限ループに陥った．（２）枝刈りされた．のいずれかである．

（１）はs_{0p}がＯＰに入ったまま戦略関数により選択されない場合である．エージェントの数と全ての状態の次数は有限であるので，同一評価値の大局的開状態も有限である．したがって，補題２．４と補題２．５から，大局的開状態の最小評価値はいずれ$f(s_{0p})$より大きくなり，s_{0p}は展開されるはずだから（１）は成り立たない．

（２）の場合は，ＬＣＵＴあるいはＧＣＵＴによる枝刈が考えられるが，初期状態から最適解上の各状態まで経路のコストは最小なのでＬＣＵＴによる枝刈りはない．また，最適解のコストを$c(p_o)$とすると，最適

解上の状態sの評価値f(s)はf(s)＜c(po)となる．さらに，最適解でない解のコストc(p)はc(po)＜c(p)であるので，その解が得られてそのコストがGCUTに代入されたとしてもf(s)＜GCUTとなり，GCUTによる枝刈りもないので（２）は成り立たない．

従ってp$_0$は必ず展開される．

p$_k$が展開されたとすると半編成システムの定義よりs_{kn_k}はそのエージェントの大局的知識に含まれるので，次のエージェントに依頼されp$_{k+1}$もp$_0$と同様に展開される．

従って最適解は必ず展開される．　□

［補題２．６］

解が存在すれば，全ての局所的探索は終了する．

（証明）

いずれかのエージェントが解を見つけるとGCUT＜＋∞となる．エージェントの数と全ての状態の次数は有限であり，同一評価値の大局的開状態も有限である．したがって，補題２．３と補題２．４から大局的開状態の最小評価値はGCUTより大きくなり，いずれは開状態が空となり，局所的探索は終了する．　□

［定理２．５］

ＤＯＰＳＡは適格である．

（証明）

補題２．３と補題２．６から解が存在すればＤＯＰＳＡは終了し，定理２．４からＤＯＰＳＡは適格である．　□

第４節　考察

一般の分散システムでは通信速度がシステム全体のボトルネックになることが多く，通信量は分散型探索アルゴリズムを評価する上での重要な要因になる．まず通信量を求めるために知識とその分割に関して以下の前提を設ける．

（１）規則にはすべて同一のコストがかかる．

（２）局所的探索はすべて局所的初期状態から１以上の長さの経路が得られた後，ｍ個の探索依頼を生成する．

（３）大局的探索は深さｎの木になり，解は一つだけ存在する．

ＤＰＳＡでは探索依頼メッセージと解到達メッセージのみを用いるが，メッセージの到着の順序により通信量は変化する．最善は図２．５に示すように大局的探索が広がる前に解が得られる場合で，ｎｍ個の探索依頼メッセージと１個の解到達メッセージが通信される．また最悪は図２．６に示すように全ての経路が展開されてから解が得られるような場合で，$\sum_{k=1}^{n} m^k$個の探索依頼メッセージと１個の解到達メッセージが通信される．

ＤＯＰＳＡでは解のコストよりも小さいコストの経路はすべて展開されるので，最低でも通信量は探索依頼メッセージと探索終了メッセージが$\sum_{k=1}^{n} m^k$個，解転送メッセージと終了メッセージが各１個通信される．しかし，これに加えて再依頼に必要な通信量が増える可能性がある．例えば，図２．７に示される場合には依頼１の受諾による展開がなされ，それにともなう依頼２がなされた後，評価値の低い依頼３を受諾すると，それにより全ての状態が再展開され，依頼４が再び行われる．このような再依頼は次々と伝搬したり，何重にも発生し，通信量が急激に増加する場合がある．このような再展開，再依頼は，大局的にコストの大きい依頼がコストの小さい依頼よりも先に行われることが原因となる．従って，コストの小さい順に依頼がなされるようになれば，再依頼が減少し，全体的な探索時間が短縮されることが予想される．このような大局的制御は分散シ

ステムの独立性から完全に実現することは難しいが，通信に優先度を設け，それを用いた依頼をすることにより実現することができる．

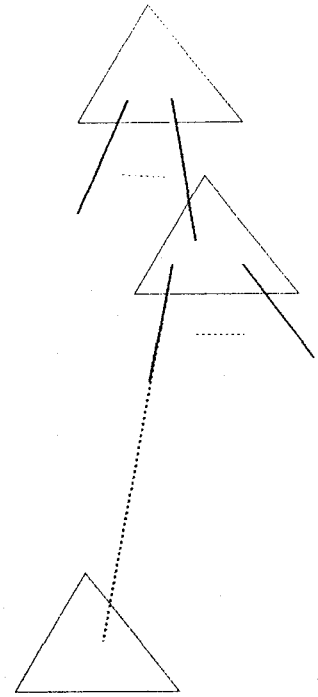


図２．５　ＤＰＳＡの通信量（最善）

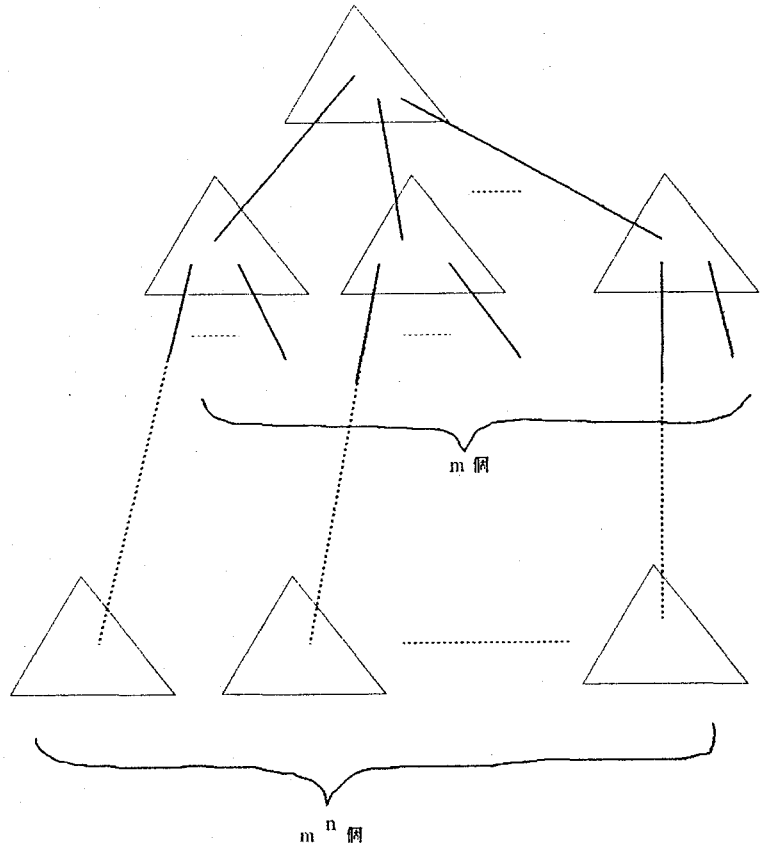


図２．６　ＤＰＳＡの通信量（最悪）

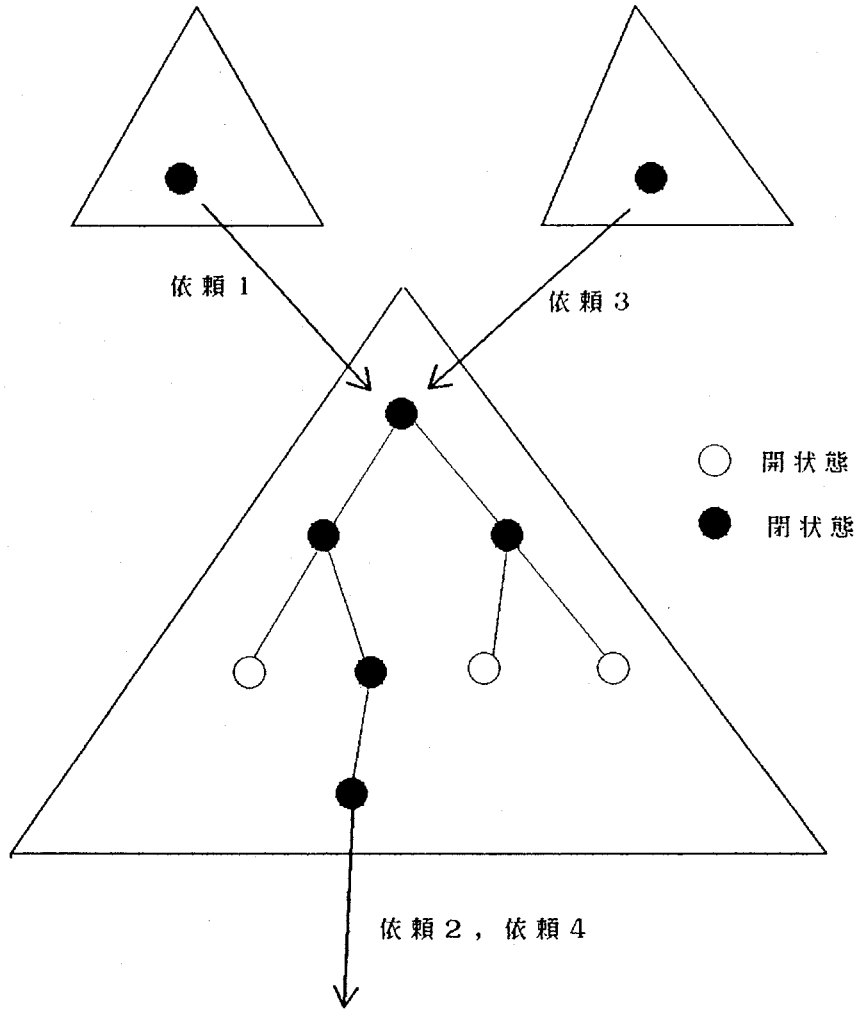


図２．７　再依頼

第5節　結言

本章では分散型問題解決の定式化とその推論方式である分散型探索アルゴリズムについて述べた．まず定式化は問題解決を知識を表す状態空間グラフにおける探索問題であるとし，分散型問題解決はこのようなグラフが各エージェントに分散していると考えている．このような知識の編成形態として半編成システム，全編成システムの概念を定義し，以後の議論を半編成システムを前提として行った．分散型探索アルゴリズムは単一解を求めるＤＰＳＡと最適解を求めるＤＯＰＳＡを提案したが，ＤＯＰＳＡでは大局的終了判定を実現するためにDijkstraとScholtenの方法を用い，それに応じてＤＰＳＡを拡張した．従来の分散型探索アルゴリズムに関連する研究として分散アルゴリズムの研究[13,15,16]があるが，これらは局所的探索を単なる展開ととらえて，大局的探索のみを扱っているといえる．分散型探索では大局的探索と局所的探索が相互に影響し合い，アルゴリズムも複雑になる．最後に通信量の観点から両アルゴリズムの評価を行った．分散型探索では問題の依頼の順序によっては通信量が大きく変化し，大局的に重要な問題から先に依頼するといった大局的制御が重要になる．

本章では知識はすでに分割され，各エージェントに分散されているという前提のもとで議論を進めたが，このような分割法に関する研究も重要である．これからの課題としたい．

第３章　分散型問題解決のための一通信方式

第１節　序言

分散型問題解決では分散しているエージェント間で，いかに協調して一貫性のある問題解決を行うかが課題である．すなわち各エージェントにおいて生成された問題を適切なエージェントに割り当て，その結果を転送するといったメッセージのやり取りを規定する必要がある．このような問題解決プロトコルの機能には以下のものがあげられる．

（１）動的な問題割り当て

従来の分散システムでは他のエージェントの能力に関する情報を大局ディレクトリとして持っており，その情報に従って問題を静的に割り当てていた．しかし人工知能の扱う問題には環境が常に変化してゆくようなものがあり静的な情報では対応できない場合がある．例えば移動作業ロボットを扱うような場合には，ロボットの位置により対象物体をつかめたり，つかめなかったりするように，状況に応じて解決できるエージェントが異なったり，以前に解決できたエージェントが解決できなくなることがある．また扱う問題が多種，多様な場合にはディレクトリのサイズが大きくなり，情報を更新する場合などの管理が困難になる．そこでこのような問題に対処するものとして，メッセージのやりとりによって動的に問題を割り当てるＣＮプロトコル（Contract Net Protocol）がある[19]．ＣＮプロトコルは人間社会における「契約」をメタフォとしており，以下に示すような３段階のメッセージのやり取りで問題割り当てを実現している．

（ａ）問題を割り当てる依頼エージェントは解決依頼メッセージを全てのエージェントに送る．

（ｂ）受諾可能なエージェントは受諾メッセージを依頼エージェ

ントに送る.

（ｃ）依頼エージェントはその中から解決に適切なエージェント（解決エージェント）を選び，通知メッセージを送る.

（２）知識の編成形態に応じた問題割り当て

分散型問題解決では各エージェントは知識を分散して管理している．第２章でも述べたように知識の編成形態には全編成システム，半編成システムがある．全編成システムはひとつの問題に対して同等の解決能力を持ったエージェントが複数存在する負荷分散システムであるので，その割り当ては受諾可能なエージェントのうち最も負荷の低いエージェント一つだけへの割り当てとなる．また半編成システムは一つの問題に対して異なる解決能力を持ったエージェントが存在し，それらの部分解の組合せにより全体解を得るといった機能分散システムであるので，その割り当ては受諾可能な全てのエージェントへの割り当てとなる.

（３）必要な解の個数に応じた実行制御

要求される解の数に応じて，問題のタイプは単一解を求めるsingle型と複数解を求めるmulti型に分類できる．single型ではひとつのエージェントの結果が得られれば，他のエージェントの実行を停止させ，multi型は適当な数の結果が得られた時点で，他の実行中のエージェントを停止させる.

（４）大局的な問題解決制御

人工知能における問題解決においては，適用する規則の組合せ的爆発に対処するための問題解決の制御は重要な課題である．また，一般に分散システムを用いたアプリケーションでは，通信が全体のパフォーマンスのボトルネックになることが多く，分散型問題解決では適用する規則の組合せ的爆発とともに，通信量の組合せ的爆発が生じ，問題解決のパフォーマンスに大きな影響が生じる可能性がある．そこで各エージェントで行われる局所的な問題解決制御とともに，分散型問題解決システム全体でも大局的な問題解決制御を行う必要がある．この問題は重要であるにもかかわらず従来の問題解

決プロトコルであるＣＮプロトコルでも考慮されていなかった.

第２節では以上の機能をみたす回覧板方式に基づく問題解決プロトコルを提案する. 回覧板方式ではメッセージを各エージェントに巡回させ必ず送信したエージェントに戻ってくるので, 問題割り当てに対する受諾エージェントが存在しない場合にも容易に判定できる. また, 優先度付き通信により大局的な問題解決制御も実現できる. 第３節では第２章で述べた分散型探索アルゴリズムＤＰＳＡのシミュレーションにより優先度付き通信の評価をおこなっており, 特に通信速度が遅いほど優先度付き通信の有効性が高まることを示している. また第４節ではＣＮプロトコルを始めとする従来の問題解決プロトコルとの比較をおこなっており, 第５節でまとめとする.

第２節　回覧板方式に基づく問題解決プロトコル

２．１　設計方針と概要

問題割り当てに必要なエージェント間での１対多，多対１のグループ通信を実現するために，人間社会のグループ通信に用いられている「回覧板」をメタフォとした回覧板方式を用いる．回覧板方式では送信エージェントは情報を回覧板によって各エージェントに巡回させ，受信エージェントはその情報を得て，それに対する返答を回覧板に書き込んでゆく．一巡してきた回覧板により送信エージェントはその返答を得ることができる．このような回覧板の巡回により，各エージェントは１対多，多対１のグループ通信を行うことができる．このような回覧板方式によって依頼エージェントが問題を回覧板を用いて巡回させ，その返答を各エージェントが書き込むことにより，ＣＮプロトコルに基づくメッセージのやりとりによる動的な問題割り当てが実現できる．また，必ず回覧板が巡回して戻ってくるので問題割り当てにおいて受諾エージェントが存在しない場合にも容易に判定できる．

また，知識の編成形態に応じてた問題の割り当てはＣＮプロトコルの拡張により実現している．全編成システムの場合は解決依頼，受諾，通知の３段階で，半編成システムの場合は受諾可能な全てのエージェントに問題を割り当てるので解決依頼，受諾の２段階で割り当てを行う．さらに問題のタイプに応じた実行制御も解を回覧板を用いて巡回させることにより実現している．

回覧板方式を用いた問題割り当てのための通信プロトコルは，図３．１に示すように三つの階層から構成されている．基本通信プロトコル(basic communication protocol)は１対１のエージェント間の物理的な通信を規定するもので，通信エラーによるメッセージの消失は自動的に回復され，誤りのない通信路を実現すると仮定している．回覧板プロトコル(circulation board protocol)は基本通信

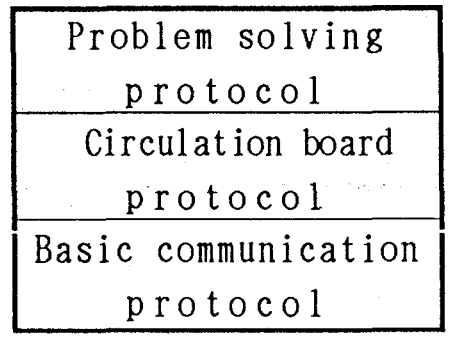


図３．１　通信プロトコル

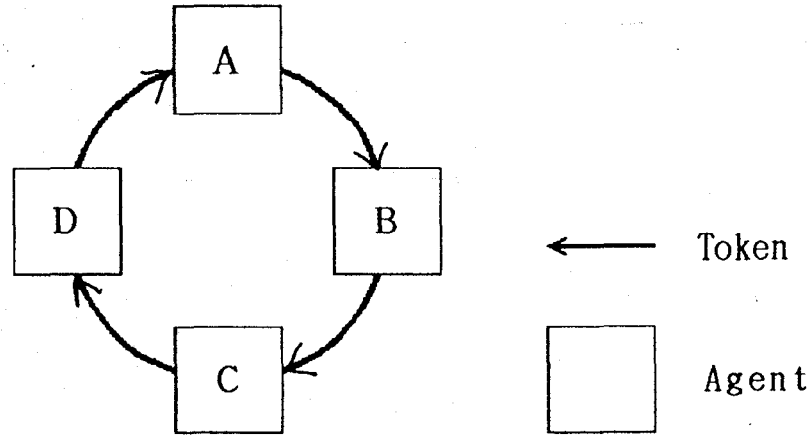


図３．２　分散型問題解決システム

プロトコルを用いて，回覧板方式による1対多，多対1の優先度付きエージェント間通信を規定する．問題解決プロトコル(problem solving protocol)は問題の割り当て，解決結果の転送に必要なメッセージのやりとりを規定する．以下，エージェントの構成，回覧板プロトコル，問題解決プロトコルの詳細を述べる．

2．2　エージェント

エージェントは大きく推論部と通信部に分けられ，推論部では与えられた問題を解決したり，他のエージェントに割り当てるために分割する．エージェント間は図3．2に示すように論理的にリング状に結合されており，推論部は通信部を介して他のエージェントとメッセージのやり取りを行う．

エージェントの通信部を図3．3に示す．ネットワークインタフェース(network interface)は計算機ネットワークなどの物理的通信媒体と接続され，メッセージ処理システム(message processor)と送信，受信用の通信バッファ(buffer)を通して接続されている．関連域データベース(interest area database)にはエージェントの受諾可能な問題が記述されている．基本通信プロトコルはネットワークインタフェース内で，回覧板プロトコル，問題解決プロトコルはメッセージ処理システム内で実現される．

2．3　回覧板プロトコル

エージェントがメッセージ通信に用いるパケットの形式は図3．4に示すように，優先度(priority)，メッセージを送出する送信エージェント名(source agent)，メッセージの宛先となるエージェントのリストを記述した受信エージェントリスト(destination agent list)からなるヘッダ部(header)，主メッセージ(main message)と副メッセージ(sub message)からなる可変長のデータ部(data)から構

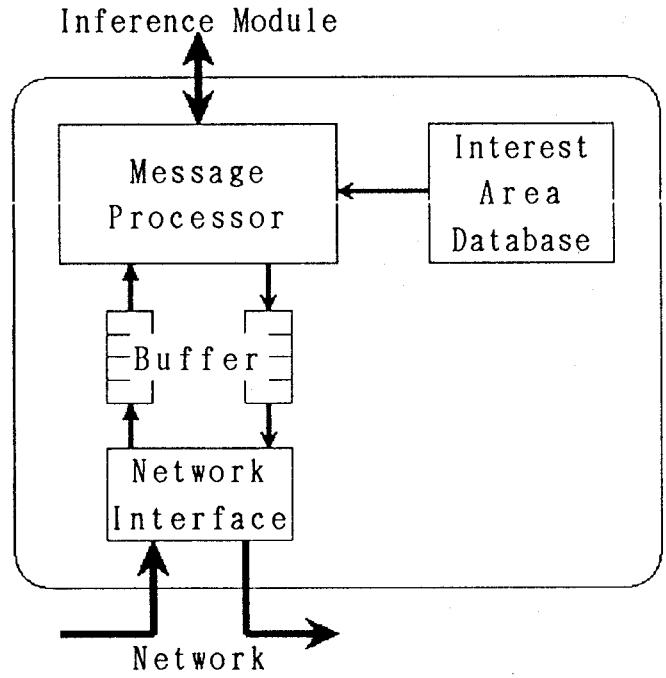
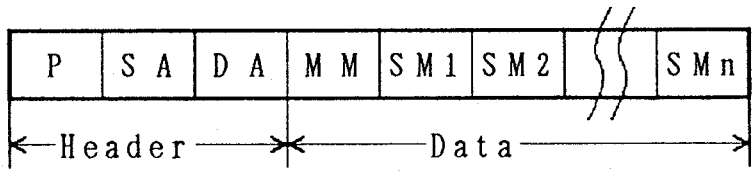


図３．３　通信部の構成



P:Priority
SA:Source agent　DA:Destination agent(s) list
MM:Main Message　SMi:Sub message i

図３．４　パケットの形式

成されている．リング状の通信路には送信権を示すトークンが巡回している．トークンを受け取ったエージェントは送信バッファにパケットが存在すればトークンとともに次のエージェントに送る．ただし，送信用通信バッファのパケットの中でもっとも優先度の高いパケットを選んで送信することにより優先度付き通信を行う．

グループ通信は三つのフェーズより成り立っている．

（１）パケットの生成

送信エージェントはヘッダと主メッセージからなるパケットを生成し，送出する．

（２）パケットの解釈・操作

受信エージェントは主メッセージを解釈し，副メッセージに操作をおこなう．操作には図３．５に示されるような連結型(concatenation)，置換型(replacement)，通過型(pass)などがある．

（３）パケットの削除

送信エージェントまでパケットが一巡してくると，パケットを削除する．

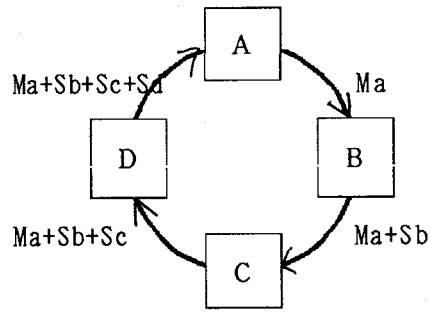
以上のようなパケットの巡回により，送信エージェントは情報を他のエージェントに知らせ，副メッセージにより他のエージェントの状態を知るといった１対多，多対１のグループ通信が実現できる．また，パケットは受信エージェントリストで示されるエージェントを一巡し，必ず送信エージェントに戻ってくるので，送信エージェントは各受信エージェントとの同期が容易にとれる．

２．４　問題解決プロトコル

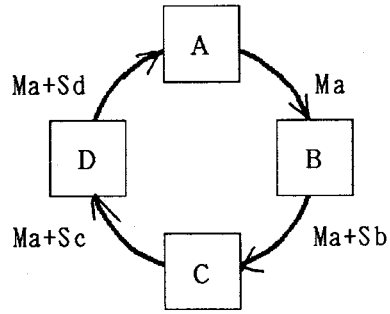
回覧板プロトコルを用いた問題解決プロトコルは問題割り当て，結果転送，終了の三つのフェーズから成り立っている．それぞれで用いるメッセージを以下に示す．

（１）解決依頼メッセージ（主メッセージ，連結型）

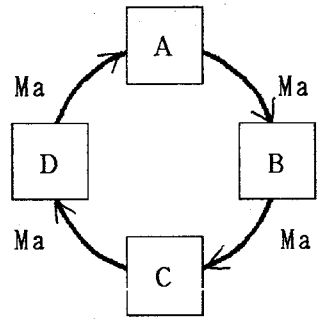
<REQUEST,problem_id,problem,type,exec>



(a)Concatenation



(b)Replacement



(c)Pass

図３．５　回覧板プロトコルによるグループ通信

problem_id: 問題識別子

problem: 問題

type: 解のタイプ

（単一解(single)／複数解(multi)）

exec: 割り当て法

（２段階(go)／３段階(wait)）

（２）受諾メッセージ（副メッセージ）

<ACCEPT,agent,info>

agent: 受諾エージェント名

info: 付加情報

（３）通知メッセージ（主メッセージ，通知型）

<AWARD,problem_id,agent>

problem_id: 問題識別子

agent: 解決エージェント名

（４）結果転送メッセージ（主メッセージ，通知型）

<RESULT,problem_id,result>

problem_id: 問題識別子

result: 結果

（５）終了メッセージ（主メッセージ，通知型）

<END,problem_id>

problem_id: 問題識別子

依頼エージェントが問題を割り当てるプロトコルを以下に示す.

［問題割り当てフェーズ］

①依頼エージェントは解決依頼メッセージ送出する.

②受信エージェントは以下の手続きを実行し，次のエージェントにメッセージを送る.

1. 問題が関連域に含まれているならば受諾メッセージを連結する. 問題のタイプを記録しておく.

1.1 exec=goならば解決を開始する.

1.2　exec=waitならば通知メッセージが来るまで，解決を保留する．

2.　問題が関連域に含まれていなければ何もしない．

③一巡してきた解決依頼メッセージと連結された受諾メッセージにより，依頼エージェントは受諾エージェントを知る．解決依頼メッセージに受諾メッセージが連結されていなければ問題の割り当てに失敗したことがわかる．

／＊　exec=waitの場合，以下を実行する　＊／

④依頼エージェントは受諾メッセージの付加情報をもとに適切なエージェントを決定し，通知メッセージを送る．

⑤受信エージェントは以下の手続きを実行し，次のエージェントにメッセージを送る．

1.　解決エージェントとして選ばれれば，解決を開始する．

2.　解決エージェントとして選ばれなければ，問題を破棄する．

□

以上のメッセージのやりとりにより，知識の編成形態に応じた2段階（半編成システム），3段階（全編成システム）の動的な問題割り当てを実現している．さらに，解決依頼メッセージの優先度に問題の重要度を用いることにより，大局的な問題解決制御が可能になる．

解決エージェントは問題解決が終了すると，問題のタイプがmulti型ならば依頼エージェントに直接，結果転送メッセージを送る．single型ならば，以下のプロトコルに従って結果を転送する．

［結果転送フェーズ］

①解決が終了した解決エージェントは結果転送メッセージを送出する．

②受信エージェントは以下の手続きを実行し，次のエージェントにメッセージを送る．

1. 依頼エージェントであれば，結果を取り込む．
2. 受諾エージェントであれば，解決を中止する． □

問題のタイプがsingle型であれば，結果の転送と問題解決の終了を同時に行える．問題のタイプがmulti型であれば，依頼エージェントは必要な結果が得られた時点で終了メッセージを用い，受諾エージェントで行われている問題解決を終了させる．

［終了フェーズ］

① 依頼エージェントは終了メッセージを送る．

② 受信エージェントは解決を中止し，次のエージェントにメッセージを送る． □

第３節　回覧板プロトコルの評価

分散型探索のシミュレーションを行い，回覧板方式における優先度付き通信の効果を調べた．探索アルゴリズムは第２章で述べたＤＰＳＡで，図３．６に示す整数組(x,y)（ただし，$|x|,|y| \leqq 8$）からなる格子状グラフにおいて，初期状態$s_s=(0,0)$から目標状態$s_g=(4,4),(4,-4),(-4,4),(-4,-4)$までの四つの問題を扱う．各エージェントへの規則の割り当ては半編成システムとして以下に示すように行った．

エージェント数をk，各エージェントを$a_1, a_2, \ldots, a_k$とした時，以下の手続きにより操作子を割り当てる．

①図３．７に示すように状態(x,y)に対して，その操作子をα，β，γ，δとして，それぞれに次のように番号pを与える．

$$p=\begin{cases} 4*(x+y) & (\alpha\ \text{のとき}) \\ 4*(x+y)+1 & (\beta\ \text{のとき}) \\ 4*(x+y)+2 & (\gamma\ \text{のとき}) \\ 4*(x+y)+3 & (\delta\ \text{のとき}) \end{cases}$$

②操作子番号pに対してi=mod(p,k)ならばa_{i+1}に割り当てる．

エージェント数が４の場合の割り当てを図３．８に示す．

また，探索の評価値としては

$$f(x,y)=\sqrt{(x-m)^2+(y-n)^2}$$

ただし，$s_g=(m,n)$

を用いた．シミュレーションは一つの操作子を適用するのに要する推論時間t_iと，エージェント間の通信時間t_cの比$R=t_c/t_i$，エージェント数Nに関して，分散型探索アルゴリズムにより初期状態から目標状態までの探索に要する時間を求めた．また，対比のために優先度付き通信を行わない場合と$t_c=0$の理想時間も同時に求めた．R=

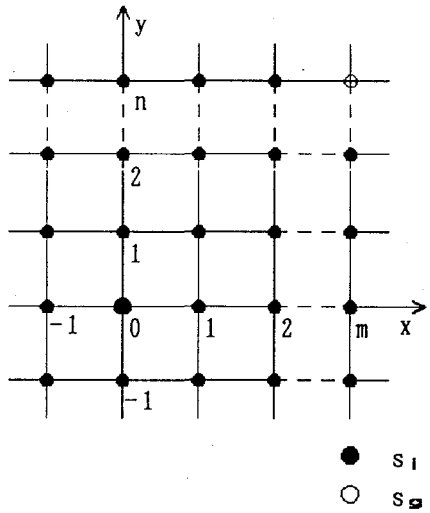


図３．６　問題のグラフ

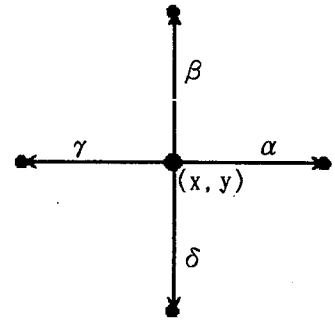


図３．７　操作子

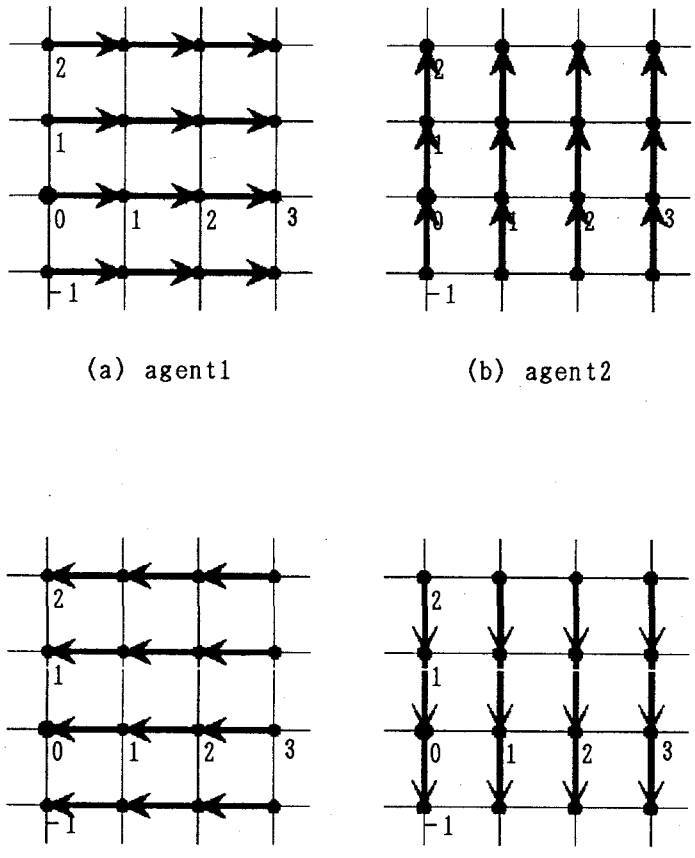
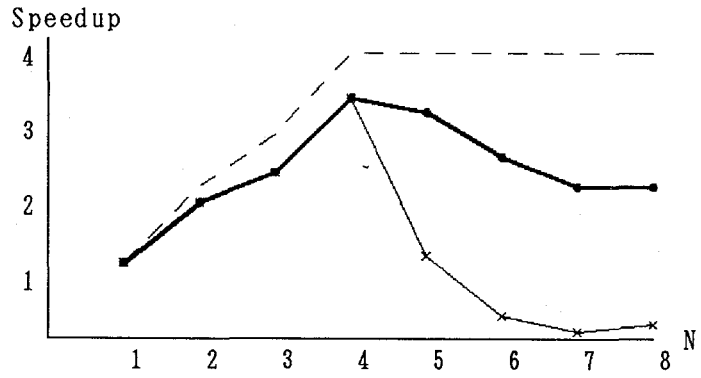


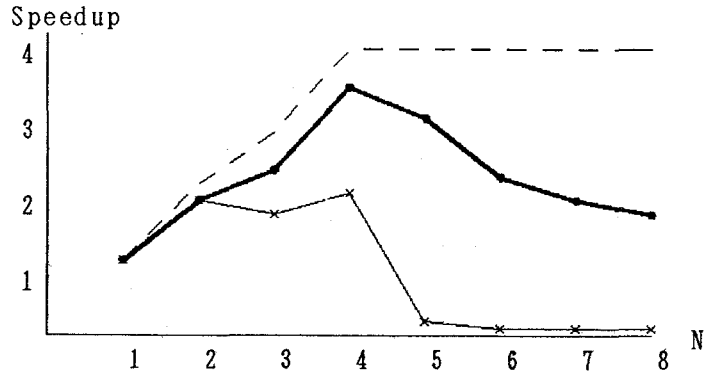
図３．８　問題分割の例

0.1, 0.5, 1, N＝1～8に関して4つの問題のシミュレーション結果の平均を図3．9に示す．縦軸はN＝1の場合との時間比で表している．

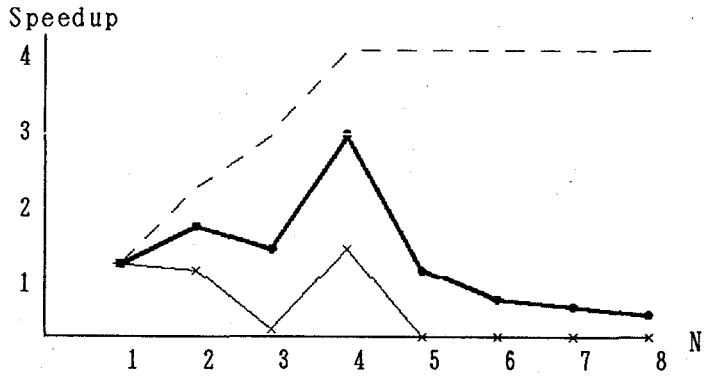
全体にRが小さいほど，理想時間に近くなっていることがわかる．また，エージェント数がある程度増えると速度が低下していく．これはエージェントが増えるにつれ，メッセージが一巡するのに時間がかかることが原因にあげられる．また，グラフからもわかるように優先度付き通信の効果は，Rが大きくなるほど（通信がボトルネックになるほど）大きくなっている．これは大局的に評価値の高い問題が優先的に割り当てられるためであり，これがさらに評価値の低い問題の生成を抑え，通信量を減らすことにもなっている．



(a) R=0.1



(b) R=0.5



(c) R=1

ideal

with priority

without priority

図3.9 シミュレーション結果

第４節　考察

本節では大局的な問題解決制御と通信量に関して，回覧板方式と他の問題解決プロトコルの比較について述べる．

ＣＮプロトコルでは送信権を規定していないので各エージェントはそれぞれ局所的な判断に基づき自由に通信できる．従って，局所的には重要であっても大局的にそうでない問題の割り当てが行われると，その解決に必要な通信が増加し，大局的に重要な問題の割り当てを妨げるといった悪循環が生じてしまう．一方，回覧板方式では優先度付き通信により，重要な問題から先に割り当てられ，通信路は常に重要な問題の解決に利用される．人間社会の比喩を用いれば，ＣＮプロトコルは１対１の関係により成立する「契約」をメタフォとしたプロトコルであるので，社会全体を考慮しておらず，ある二者間にとっては有益であるが，社会にとって有害な契約が結ばれたとしてもその社会にはそれを阻止する能力を持たず，それがさらに有害な契約を発生させることがある．それに対して回覧板方式では第三者がその契約を常に確認する機能を持っているので，社会にとって有害な（優先度の低い）「契約」は後回しにして，社会にとって有益な（優先度の高い）「契約」が優先されているといえる．

半編成システムにおける問題割り当てを通信量から考察すると，回覧板方式ではエージェントの数をｎとしたとき，受諾エージェントの数にかかわらず解決依頼メッセージを巡回させるためにｎ回の通信が必要である．それに対してＣＮプロトコルでは受諾エージェントの数をｍとすれば，１回の解決依頼メッセージとｍ回の受諾メッセージを用いるが，$m \leq n-1$ よりＣＮプロトコルの方が通信量は少なくてすむ*．しかし，ＣＮプロトコルでは受諾エージェントが存在しなかった場合には受諾メッセージが返ってこないので，依頼エージェントは他のエージェントがメッセージを処理中なのか，受諾できないのかが判定できないといった問題が生じる．これに対処する方法には二通り考えられる．

一つは締切時刻(expiration time)を設けて，その時間T内に到着した受諾メッセージのみを有効とする方法である．しかし，このようなプロトコルを実行する時にTを決定することはエージェントの状態や通信路の状態により難しく，Tが長すぎると無駄な待ち時間が多くなり，短か過ぎると依頼すべきエージェントに依頼できない可能性が増えてしまう．

もう一つの方法は解決依頼メッセージを受け取った全てのエージェントが受諾不可能であったとしてもその旨のメッセージを返すという方法である．しかし，この方法ではメッセージ数が1＋（n－1）＝n回と多くなり，割り当て毎に返答のメッセージを受け取ったエージェントを記録しておかなければならずその管理も容易でない．

それに対して回覧板方式では解決依頼メッセージは受諾メッセージとともに必ず一巡して戻ってくるので依頼エージェントは容易に受諾エージェントを決定できる．また，回覧板方式における通信量を減らす方法としては，問題に対して受諾の可能性がないことがあらかじめわかっておれば受信エージェントリストからそれらのエージェントを除くことにより，通信量を少なくするということも考えられる．

---

*ここで，同報通信を用いる解決依頼メッセージを1回と計算しているが，同報通信を実現していない通信システムではm回と計算しなければならない．

第５節　結言

分散型問題解決における問題割り当てのための通信方式を提案し，その評価として分散型探索を取り上げ，シミュレーションにより優先度付き通信の有効性を示した．

本方式には次のような長所がある．

（１）人工知能における問題解決ではエージェント自身の能力が環境の変化に応じて変化するような動的な問題割り当てを実現しなければならず，問題の割り当てを従来の静的なディレクトリをもとに決定する方法では対応しきれない．そこでメッセージのやりとりにより動的な問題割り当て法を知識の分割法，必要な解の個数に応じて提案した．

（２）回覧板方式ではメッセージは受信エージェントを巡回して送信エージェントに戻ってくるので，問題割り当てに対して受諾エージェントが存在しない場合にも容易に判定できる．

（３）優先度付き通信により大局的な問題解決制御が実現でき，問題解決の効率が改善される．

逆に，エージェントの数に応じてメッセージを巡回させるのに時間がかかるといった短所もある．巡回時間Ｔはエージェント数をｎ，エージェント間の通信時間をｔとした場合，

$$(n^2+n)\,t > T \geqq n\,t$$

となる[20]．

これからの課題としては，分散型問題解決における回覧板方式の応用を進めてゆくことがあげられる．

第４章　分散型問題解決システムのマイコンネットワーク上への実現

第１節　序言

分散型問題解決に関する研究は歴史が浅く，その方法論も確立していない．そこで，理論的な考察と共に，実際にシステムを構築し，実用上の観点から考察を進めることも有益である．従来の分散型問題解決システムとして黒板システムをもとにした分散型解釈モデル，Yangらのシステムがあげられるが，いずれもミニコンやワークステーションをエージェントとした大規模なシステムであった．本章では分散型問題解決に必要な機能を明確にし，さらに拡張が容易なように低コストの分散型問題解決システムをマイコンネットワーク上ＡＩＤ－Ｎｅｔに試作した．本システムの特徴は以下にあげられる．

（１）分散型問題解決の基本的機能として，エージェントは問題の解決と共に，問題分割を行い，他のエージェントに依頼する協調的な問題解決が可能である．このような問題分割法として，ＡＮＤ／ＯＲ型，逐次／並列型の組合せである４種類が可能である．

（２）分散型問題解決システムに必要な機能が知識源，データ部，解決部，問題管理部，エージェント管理部，通信部の各部に明確にモジュール化されている．

（３）研究室内マイコンネットワークＡＩＤ－Ｎｅｔ上にシステムを構築することにより，低コストで分散型問題解決システムが実現できる．また，ＡＩＤ－Ｎｅｔには通信プログラムを動的に変更するダイナミックダウンロード機能があり，分散型問題解決システムの開発や保守が容易になる．

本章では第２節で分散型問題解決システムを実現するために必要となる機能について述べ，第３節でエージェントを構成する各モジュールの詳細について述べる．さらに第４節では列車乗り継ぎ問題

を通して本システムでの記述例を示している．第5節ではマイコンネットワーク上へのインプリメントについて述べ，第6節で全体的なまとめとする．

第２節　分散型問題解決システムの機能

分散型問題解決は問題の部分問題への分割，部分問題の各エージェントへの割り当て，部分問題の解決，結果の転送といった手続きを繰り返すことにより行われ，以下の機能が必要である．

（１）問題の分割

問題を部分問題に分割する時にはその分割方法とその記述が問題点となる．分割法には問題を次々と分解してゆくトップダウン的な分割法としてのＡＮＤ／ＯＲ型分割と，問題解決の並列性をあらわす逐次／並列型分割がある[21]．以下，分割される問題を親問題，親問題が分割されることにより生じた部分問題のことを子問題といい，順序が定まっている．また前提として分割の最小単位となる解決モジュールのことを知識源と呼び，それらの組合せにより問題は解決されるとする．

ＡＮＤ／ＯＲ分割による問題解決は容易に推察されるように，ＡＮＤ型分割は分割された子問題の全てが解決されたとき，親問題も解決されたとし，ＯＲ型は分割された子問題のいずれかが解決されたとき，親問題が解決されたとする．前提から問題を子問題が知識源で解決できるまで分割してゆけばよい．

また，逐次型分割は子問題の解決に順序付けが必要な場合であり，並列型分割は順序付けが必要でなく，任意の順序で解決してよい場合である．例えば，子問題の解決に相互干渉が生じる場合は逐次型分割となる．

（２）問題の割り当て

知識源は各エージェントに分散しており，与えられた問題を解決する知識源がエージェント内に存在しない場合には他のエージェントにその問題の解決を依頼しなければならない．そこで，外部エージェントの問題解決能力（知識源のリスト）を各エージェントが記述しておき，それに応じて解決の依頼を行う．

（３）部分問題の解決

エージェントは最小単位にまで分割された問題を知識源を用いて解決する．知識源は問題を解決するための知識の集合であり，知識表現のための枠組みを提供する必要がある．

（４）制御と問題の管理

問題の分割，知識源による問題の解決，外部エージェントへの問題の依頼と解の転送といった処理がエージェントで行われ，その制御機構が必要である．また，外部エージェントからの依頼や問題分割により生じる複数の問題を管理する機能も必要である．

（５）通信

エージェント間のメッセージのやり取りを実現する通信機能が必要である．

次節では以上の機能を実現する分散型問題解決システムについて述べる．

第３節　分散型問題解決システム

分散型問題解決システムを構成するエージェントは図４．１で示されるように知識源，データ部，問題管理テーブル，解決部，問題管理部，エージェント管理部，通信部の各モジュールから成り立っている．以下，それぞれの説明を加える．

３．１　知識源とデータ部

本システムにおいては問題解決の最小単位は知識源と呼ばれ，特定の問題を解決する規則の集まりである．知識源が問題解決に必要な事実や中間結果をデータ部としてまとめてあり，規則を適用することによりその内容が変更されてゆく．知識源の中の問題解決ルールは次のように記述される．

rule(<知識源名>,<ﾙｰﾙ名>,<条件式>,<結論式>).

<知識源名>は知識源の名前，<ﾙｰﾙ名>はルールの名前，<条件式>,<結論式>には単一の論理式または複数の論理式を[]で囲むことにより記述され，左側から処理される．<結論式>ではデータの追加と削除のために述語assertdata(<ﾃﾞｰﾀ>)とdeletedata(<ﾃﾞｰﾀ>)が使用できる．

またデータ部の内容は次のように記述される．

data(<ｴｰｼﾞｪﾝﾄ名>,<ﾃﾞｰﾀ>).

<ｴｰｼﾞｪﾝﾄ名>はエージェントの名前，<ﾃﾞｰﾀ>には論理式が記述される．知識源同士はデータ部を通して情報交換を行うことができる．

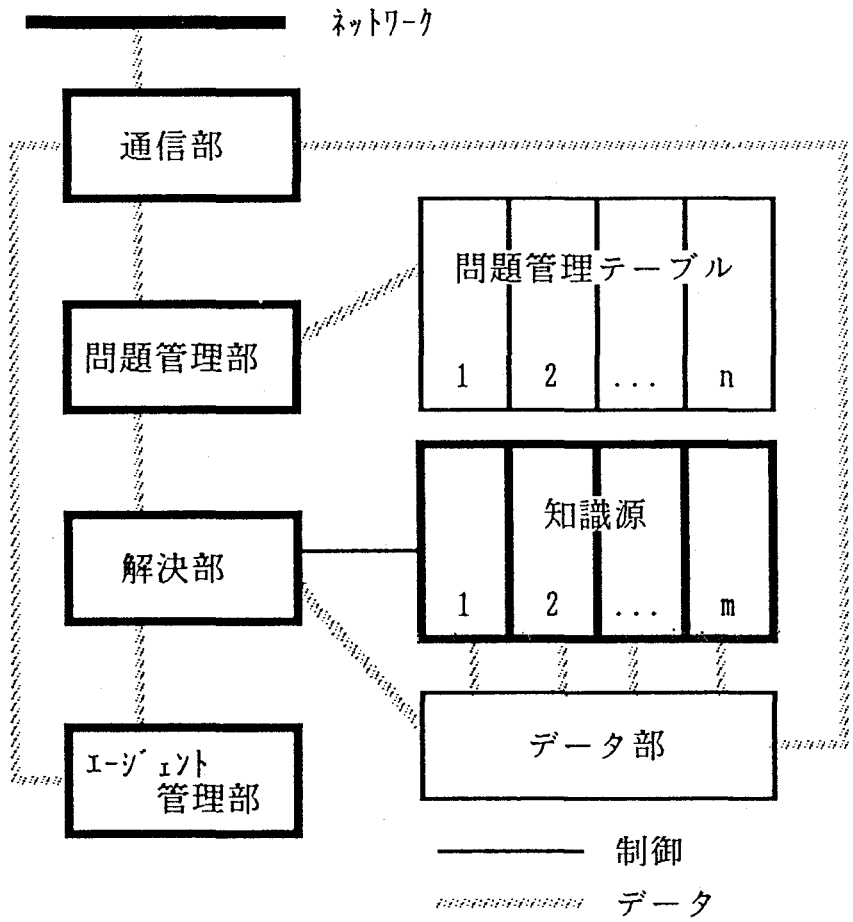


図４．１　エージェントの構成

3．2　解決部

解決部には与えられた問題に対して，適切な知識源を起動したり，問題を分割するといったメタ知識を以下のように記述する．

```
rule($<ｴｰｼﾞｪﾝﾄ名>,<ﾙｰﾙ名>,
        [<前提条件>,
        <前処理>,
        infer(<知識源名>,<ｺﾞｰﾙ>,<推論方法>),
        <後処理>],
        solve(<問題>,<分割法>)).
```

<ｴｰｼﾞｪﾝﾄ名>はエージェントの名前で，その前の$は解決部のルールであることを示す．<ﾙｰﾙ名>はルールの名前である．メタ知識には知識源の起動と問題の分割法を一つのルールとして記述することができる．<前提条件>が成り立てば，まず<前処理>が実行され，<知識源名>で示される知識源が起動され，<問題>が<分割法>に従って分割される．その後で<後処理>が実行される．<前処理>と<後処理>にはデータ部への操作を記述することができる．

<分割法>には以下の4種類がある．

（1）逐次AND　s_and([p1,p2,...,pn])

子問題p1からpnまで順番に解決する．p1からpnまで全て成功した時にのみ親問題は成功する．

（2）並列AND　p_and([p1,p2,...,pn])

子問題p1からpnを並列に解決可能である．p1からpnまで全て成功した時にのみ親問題は成功する．

（3）逐次OR　s_or([p1,p2,...,pn])

子問題p1からpnまで順番に解決する．p1からpnまでのいずれかが成功した時にのみ親の問題は成功する．

（４）並列OR　　p_or([p1,p2,...,pn])

子問題p1からpnまでを並列に解決可能である．p1からpnまでのいずれかが成功した時にのみ親の問題は成功する．

知識源の起動には<知識源名>，<ｺﾞｰﾙ>，<推論方法>を記述する．<ｺﾞｰﾙ>には変数を含む記述が可能で解決終了後，解とパターンマッチされる．<推論方法>にはfw（前向き推論），fwa（前向き推論で全ての解を求める），bw（後向き推論）の３種がある．

３．３　問題管理部

問題管理部は分割や依頼により生じる問題を問題管理テーブルにより管理する．問題管理テーブルは以下のように記述される．

```
data(<ｴｰｼﾞｪﾝﾄ名>,
        table(  <問題番号>,
                <依頼ｴｰｼﾞｪﾝﾄ>,
                <内部/外部>,
                <問題>,
                <状態>)).
```

<ｴｰｼﾞｪﾝﾄ名>はエージェントの名前を示す．<問題番号>は問題管理部が各問題に対して固有に与えるもので，問題管理部はこの番号によって問題を識別する．<依頼ｴｰｼﾞｪﾝﾄ>は与えられた問題が外部エージェントからのものであれば，そのエージェント名，内部で問題が分割されることにより生じたものであれば，分割された問題の問題番号が記述される．<内部/外部>には問題が外部エージェントからのものか(outside)，内部で分割されたものか(inside)が記述される．<問題>は問題の記述である．<状態>はその問題に対する解決の状態を示し，以下のものがある．

a．実行に関するもの

（１）実行中(executing)　実行中である．

（２）実行可(executable)　実行可能である．

（３）実行待(waiting)　実行を待っているである．

（４）依頼中(requesting(<ｴｰｼﾞｪﾝﾄ名>))　現在<ｴｰｼﾞｪﾝﾄ名>で示されるエージェントに実行依頼している．

b．結果に関するもの

（５）成功(success)　成功した．

（６）失敗(fail)　失敗した．

c．分割に関するもの

（７）逐次AND(s_and([P1,...,Pn])　逐次ANDに分割した．

（８）並列AND(p_and([P1,...,Pn])　並列ANDに分割した．

（９）逐次OR(s_or([P1,...,Pn])　逐次ORに分割した．

（１０）並列OR(p_or([P1,...,Pn])　並列ORに分割した．

外部エージェントから新しい問題解決の依頼があった場合には問題管理部は問題管理テーブルを登録する．問題管理部ではその問題に対して<問題番号>に新しい問題番号を割り当て，<依頼ｴｰｼﾞｪﾝﾄ>にエージェント名を，<内部/外部>に外部を登録する．<状態>はexecutableになる．

解決部によって問題番号aの問題pが問題番号b,cの子問題qとrに分割された場合，子問題qとrが新たに問題管理テーブルが登録され，問題pの管理テーブルの<状態>が分割法に従って変更される．子問題qとrの<依頼ｴｰｼﾞｪﾝﾄ>がa，<内部/外部>がinsideになる．<状態>は逐次型分割ならば，問題qがexecutableに，問題rがwaitingになる．並列型ならば共にexecutableになる．

問題管理部では問題管理テーブルの<状態>の項うち，executableのものを解決部へ渡し，executingとする．問題解決が終了すると，その結果に応じてsuccessまたはfailが<状態>項に書き込まれる．そ

の問題が外部依頼のものならば，結果を示すメッセージを通信部を通して送る．問題分割により生じた内部依頼のものならば，その分割法に応じて親の問題の<状態>項を変化させる．

3．4　エージェント管理部

エージェント管理部には外部のエージェントの問題解決能力（知識源）についての以下のような記述がされる．

```
data(<ｴｰｼﾞｪﾝﾄ名>,
        agent(<外部ｴｰｼﾞｪﾝﾄ名>,<知識源名>)).
```

3．5　通信部

通信部は外部のエージェントとの通信を行い，外部からの問題解決依頼の受諾，外部への依頼を行う．エージェント間の通信には次に示すメッセージを用いる．

（１）依頼　　REQUEST(問題番号,問題)
（２）受諾可　　ACK(問題番号)
（３）受諾拒否　NAK(問題番号)
（４）結果　　INFORM(問題番号,結果)
（５）成功　　SUCCESS(問題番号)
（６）失敗　　FAIL(問題番号)

エージェントAとエージェントBの間での解決依頼のためのメッセージのやり取りは図４．２のようになる．

エージェントAはBに依頼メッセージを送る．Bは実行が可能ならばＡＣＫを返し，問題管理部に登録を依頼する．不可能ならばＮＡＫメッセージを返す．解決が終了するとその結果を結果メッセージと

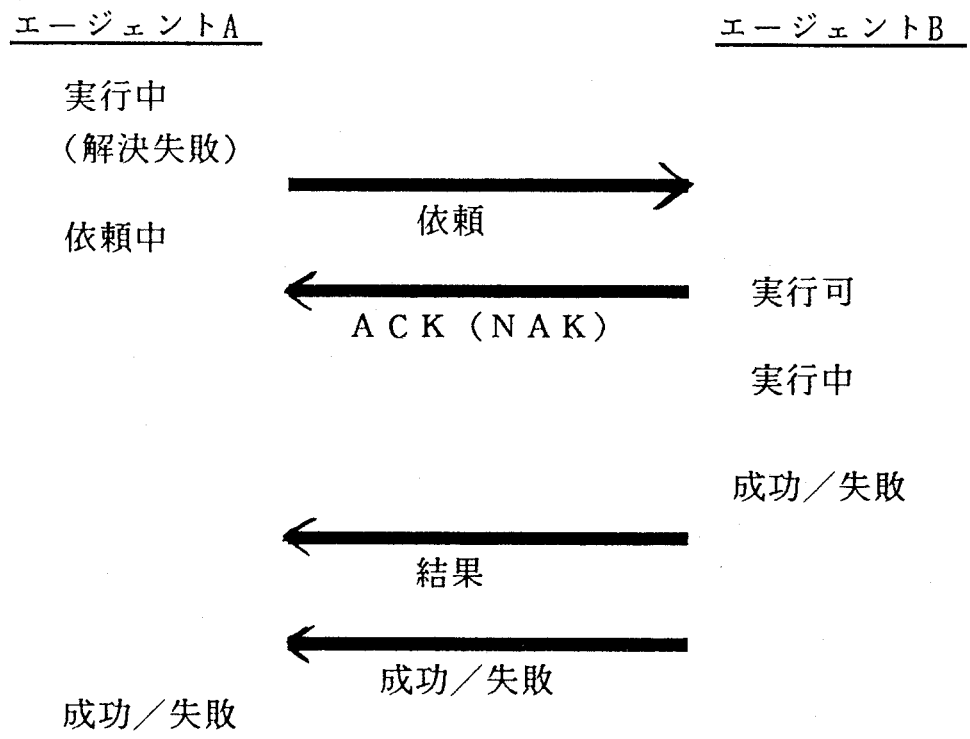


図４．２　通信プロトコルとエージェントの状態

成功，失敗のいずれかのメッセージによってエージェントAに知らせる．

3．6　制御

エージェントは次のようなアルゴリズムに従って，問題解決を進める．

ステップ1　問題管理部がいくつかの問題管理テーブルの中からexecutableの問題を選ぶ．解決すべき問題がなければ終了．

ステップ2　解決部が知識源を起動するか，問題をいくつかの子問題に分割する．分割ならステップ3へ．知識源により解決に成功すればステップ4へ．失敗すればステップ5へ．

ステップ3　問題管理部が分割法に応じて子問題を問題管理テーブルに登録する．ステップ1へ．

ステップ4　問題管理部が解決した問題が外部からのものならば，その結果を通信部を通して依頼エージェントに送る．分割により生じたものなら分割法に従って，問題管理テーブルを操作する．ステップ1へ．

ステップ5　エージェント管理部ではその問題を解決することのできる外部エージェントを決定して，依頼メッセージを通信部を通して送信する．依頼できればステップ1へ．解決できなければ解決は失敗であったとしてステップ4へ．

さらに外部からのメッセージに対して次のよう動作する．

ステップ1　通信部がメッセージを受信し，依頼メッセージであ

ればステップ２へ．成功または失敗メッセージであればステップ３へ．結果メッセージであればステップ４へ．

ステップ２　問題を解決する知識源を持てば，問題管理部が新たな問題として登録し，ＡＣＫメッセージを返す．そうでなければＮＡＫメッセージを返す．

ステップ３　問題管理テーブルを操作する．その結果に応じてメッセージを送る．

ステップ４　データ部に結果を追加する．

第４節　例題　－列車乗り継ぎ問題－

分散型問題解決システムの応用例として，列車乗り継ぎ問題があげられる．列車を利用してある地点間を旅行する計画を立てる場合，会社や路線によって複数の経路があるが，各会社線毎に時刻表を調べて，所要時間や料金をもとに最適な経路，列車を決定する．ここでは図４．３に示すように各会社ごとのエージェント(JR,HANKYU,etc.)，各路線のつながりかたを知っているエージェント(MAP)，計画を立てる立案エージェント(PLANNER)を設け，立案エージェントが各会社にデータ検索の要求を出すことにより，問題解決をすすめる．

例えば，料金による最適経路を決定する手順は次のようになる．

[1]２地点間の経路を求める．
[2]乗り換え地点を決定する．
[3]鉄道会社を決める．
[4]部分経路の料金を求める．
[5]最も安い経路を導き出す．

図４．４で示されるような鉄道路線において，京都から和歌山までの列車乗継ぎの料金を計算する場合を例に用いる．

[1]２地点間の経路を求める．

PLANNERがMAPに問い合わせることにより，京都から和歌山までの経路を求める．大阪経由と奈良経由の２経路ある．

1.京都-大阪-和歌山
2.京都-奈良-和歌山

[2]乗り換え地点を決定する．

PLANNERが[1]のそれぞれの経路に関して乗り換え地点を決定する．

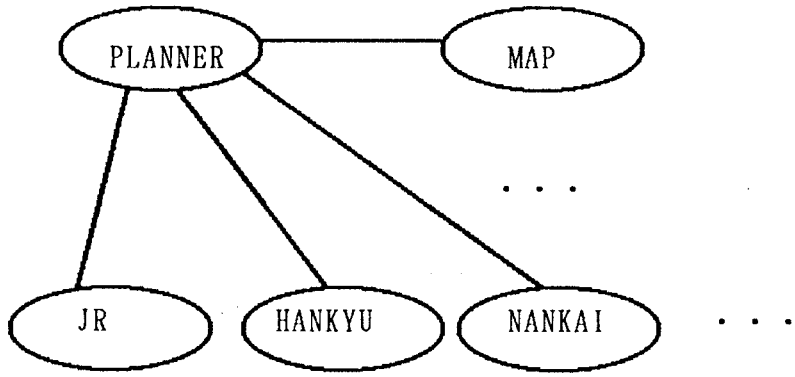


図４．３　解決エージェント

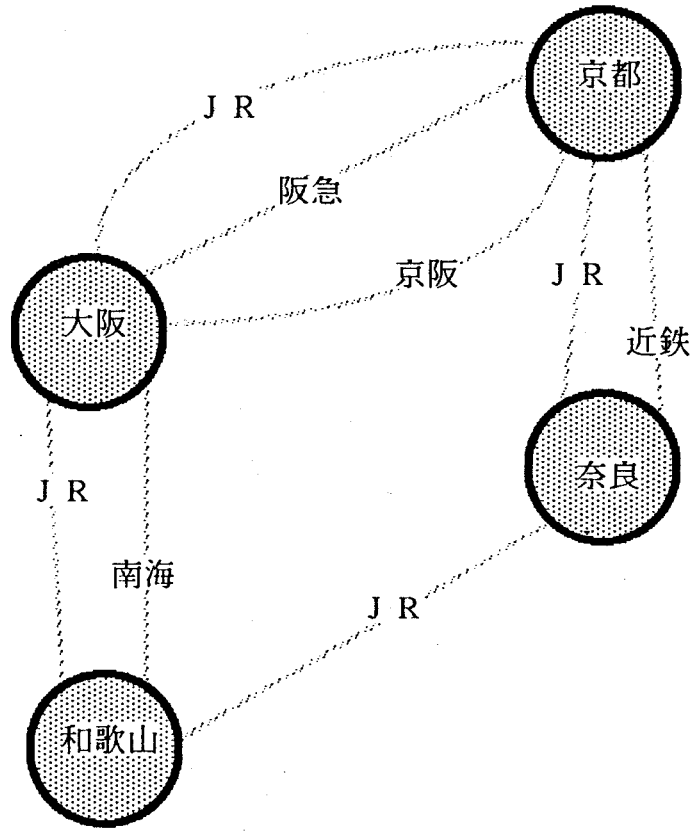


図４．４　鉄道路線図

大阪経由ならば大阪が乗り換え地点となる．また，乗り換えずに直接行く方法もある．

1.1.京都-和歌山（大阪経由）
1.2.京都-大阪-和歌山
2.1.京都-和歌山（奈良経由）
2.2.京都-奈良-和歌山

[3]鉄道会社を決める．

PLANNERが四つの経路をMAPに問い合わせる．MAPは問い合わせに対して逐次的に処理を進める．京都-大阪間はＪＲ，京阪，阪急が利用可能で，大阪-和歌山間はＪＲ，南海がある．また，京都-奈良間はＪＲ，近鉄が利用可能で，奈良-和歌山間はＪＲのみである．従って次の８経路がある．

1.1.1.京都-(ＪＲ)-和歌山（大阪経由）
1.2.1.京都-(ＪＲ)-大阪-(南海)-和歌山
1.2.2.京都-(阪急)-大阪-(ＪＲ)-和歌山
1.2.3.京都-(阪急)-大阪-(南海)-和歌山
1.2.4.京都-(京阪)-大阪-(ＪＲ)-和歌山
1.2.5.京都-(京阪)-大阪-(南海)-和歌山
2.1.1.京都-(ＪＲ)-和歌山（奈良経由）
2.2.1.京都-(近鉄)-奈良-(ＪＲ)-和歌山

[4]部分経路の料金を求める，

PLANNERが各区間の料金をそれぞれのエージェントに並列に問い合わせる．

1.京都-(ＪＲ)-和歌山（大阪経由）　1700円
2.京都-(ＪＲ)-大阪　510円

3.京都-(阪急)-大阪　300円

4.京都-(京阪)-大阪　310円

5.大阪-(ＪＲ)-和歌山　1310円

6.大阪-(南海)-和歌山　700円

7.京都-(ＪＲ)-和歌山(奈良経由)　2300円

8.京都-(近鉄)-奈良　660円

9.奈良-(ＪＲ)-和歌山　1700円

[5]最も安い経路を導き出す.

PLANNERが得られた料金から最も安い経路を導き出す.

1.1.1.京都-(ＪＲ)-和歌山(大阪経由)　1700円

1.2.1.京都-(ＪＲ)-大阪-(南海)-和歌山　1210円

1.2.2.京都-(阪急)-大阪-(ＪＲ)-和歌山　1610円

1.2.3.京都-(阪急)-大阪-(南海)-和歌山　1000円

1.2.4.京都-(京阪)-大阪-(ＪＲ)-和歌山　1620円

1.2.5.京都-(京阪)-大阪-(南海)-和歌山　1010円

2.1.1.京都-(ＪＲ)-和歌山(奈良経由)　2300円

2.2.1.京都-(近鉄)-奈良-(ＪＲ)-和歌山　1360円

従って, 京都-(阪急)-大阪-(南海)-和歌山が1000円で最も安い経路である.

以上の問題の記述例を示す. [2][3][4]の各プロセスは逐次的に処理されるので逐次ANDの分割である. これは以下のように記述される.

```
rule('$PLANNER',rule_1,
        true,
        solve(route_fare,
        s_and([ trasfer_station,
                select_company,
                ask_each_fare])).
```

経路の料金を求める(route_fare)は，乗り換え地点を決め(transfer_station)，電鉄会社を決め(select_company)，各料金を求める(ask_each_fare)ことを逐次的(s_and)に行うことが表現されている．ここでは知識源は起動されないので知識源関する記述は省略されている．（trueとだけ書かれている．）この規則が実行されると，問題管理テーブルは以下のように変化する．

```
data('PLANNER',table(
        2,
        1,
        inside,
        route_fare,
        executable)).
```

このような問題管理テーブルに対して，executableの状態から次のように分割される，

```
data('PLANNER',table(
        2,
        1,
        inside,
        route_fare,
        s_and([3,4,5]))).

data('PLANNER',table(
        3,
        2,
        inside,
        transferStation,
```

```
        executable)).

data('PLANNER',table(
        4,
        2,
        inside,
        select_company,
        waiting)).

data('PLANNER',table(
        5,
        2,
        inside,
        ask_each_fare,
        waiting)).
```

このように，３個の問題に分割され最初の問題のみ実行可能となる．

次に知識源の起動に関する記述を示す．

```
rule('$PLANNER',rule_2,
        [infer('FARE',min_fare(Route,Fare),bw),
        check_result],
        solve(best_fare,s_and([])).
```

これは[5]で最も安い経路求めるためのもので，知識源'FARE'が最小料金の経路を求めるために後向き推論で起動されている．ここでは知識源起動に関して，〈前提条件〉，〈前処理〉は省略されている．

最後にエージェント管理部の表現を示す．

```
data('PLANNER',agent(
        Company,
        fare(Route,Company))).
```

これは電鉄会社の料金fare(Route,Company)を知りたければ，その電鉄会社のエージェントCompanyに依頼すればよいことを表している．

第５節　インプリメント

エージェントはPrologベースの汎用問題解決システムＰＳＡ[22]により記述されている．ＰＳＡはメタルール部，ルール部，データ部から構成されており，メタルール部には推論制御，デモン処理，ＰＳＡ間のメッセージ処理を行うルールの記述，ルール部には目的や対象毎にグループにまとめられた推論ルールが記述される．データ部には対象問題に関する情報や推論での途中経過が記述され，ルール部により変更される．エージェントのデータ部，問題管理テーブルはデータ部で，知識源，解決部，問題管理部，エージェント管理部，通信部がルール部により記述され，全体的な制御がメタルール部に記述される．

ＰＳＡは図４．５に示されるようにマイクロコンピュータまたはミニコンピュータであるホスト計算機(Host)上にインプリメントされており，ＡＩＤ－Ｎｅｔと呼ばれる低コストの研究室内ネットワーク[24,25,26]により結合され分散型問題解決システムを構成している．ＡＩＤ－Ｎｅｔはマイクロコンピュータを通信制御装置（Communication Control Processor: ＣＣＰ）として用い，マイクロコンピュータ間はＰＣ－ＮＥＴ[23]と呼ばれる低コストＬＡＮにより接続されている．ホスト計算機はＲＳ－２３２Ｃインタフェースを通してＣＣＰにアクセスすることにより他のホスト計算機との通信を行い，分散処理システムを構成する．また，システム開発者や利用者はＡＩＤ－Ｎｅｔに接続された端末(Terminal)からホスト計算機にアクセスする．また，通信に関する違いがＣＣＰにより吸収されるので，異機種のホスト計算機，端末が接続可能である．

ＣＣＰ内の通信プログラムはKernelとD-Shellの二つの部分に分かれており，ダイナミックダウンロード機能によりD-Shellの部分が変更可能である．この機能により，分散型問題解決システム開発における通信プログラムの変更や，保守プログラムの遠隔ＣＣＰへのダウンロードによるシステムの集中的な管理などの多様なアプリケー

ションが実現できる.

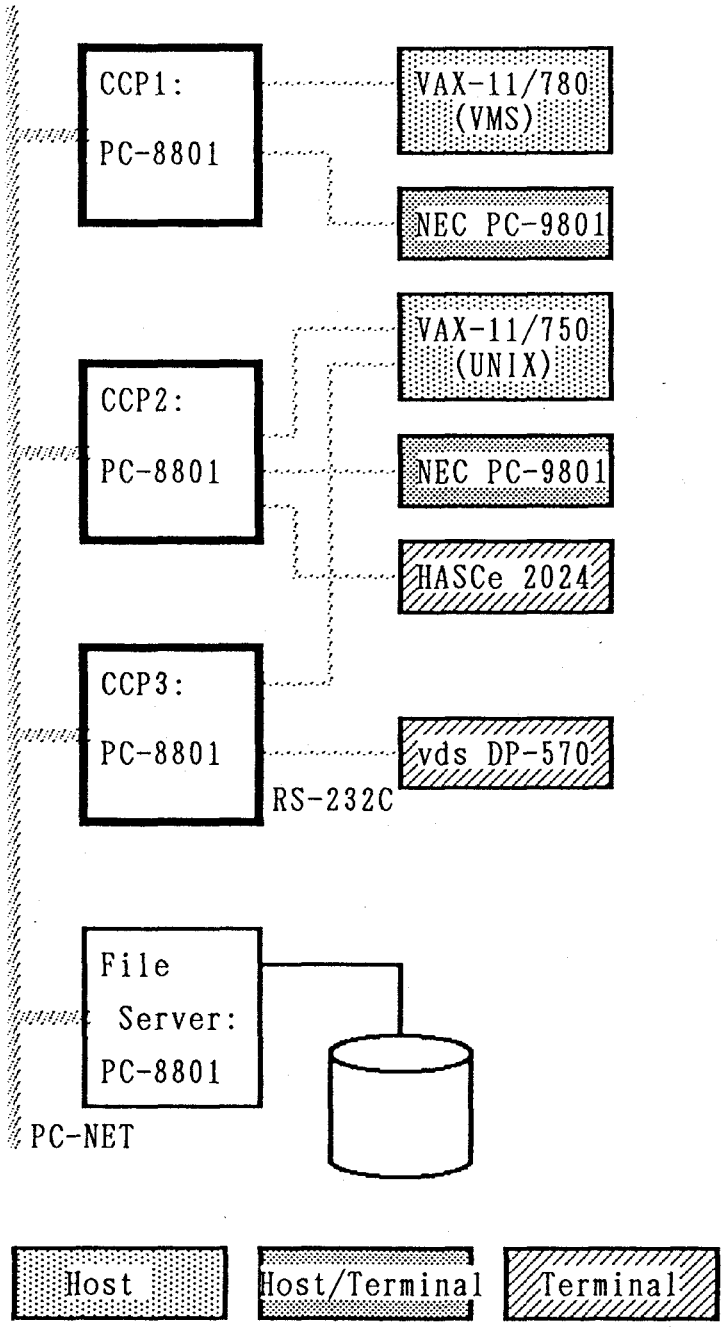

CCP1:
PC-8801
VAX-11/780
(VMS)
NEC PC-9801
VAX-11/750
(UNIX)
CCP2:
PC-8801
NEC PC-9801
HASCe 2024
CCP3:
PC-8801
vds DP-570
RS-232C
File
Server:
PC-8801
PC-NET
Host
Host/Terminal
Terminal

図４．５　ＡＩＤ－Ｎｅｔ

第６節　結言

分散型問題解決システムのマイコンネットワーク上への試作について述べた．本システムの特徴としては以下のものがあげられる．

（１）問題分割の方法として問題を部分問題に分解するＡＮＤ／ＯＲ型と並列実行に必要な逐次／並列型の両方が可能である．

（２）システムを複数の知識源，データ部，解決部，問題管理部，エージェント管理部，通信部にモジュール化することで分散型問題解決に必要な能力を別々に記述でき，その変更も容易である．

（３）エージェントをパソコン，通信システムをマイコンネットワークにより実現することにより，低コストの分散型問題解決システムが実現できた．また，ネットワークのもつダイナミックダウンロード機能によりシステムの開発，保守が容易である．

これからの課題としては試作したシステムの拡張がある．特に，第３章で述べた回覧板方式を通信方式として組み入れることが考えられる．さらに本システムを用いて分散型問題解決の応用と問題に応じた推論方式や通信方式の開発と評価がある．

第 5 章　結論

本論文では，分散型問題解決における諸問題のうち，分散型問題解決自身の定式化，分散型探索アルゴリズム，回覧板方式を用い優先度付き通信により大局的問題解決制御が可能な通信方式について述べた．また，マイコンネットワーク上への分散型問題解決システムの実現についても述べた．

これからの課題としては，提案した推論方式，通信方式を用いた分散型問題解決システムを実現し，具体的な問題に応用することが考えらる．さらに，本研究では触れなかった知識の分割法に関する研究も課題となる．

分散型問題解決の一般的な課題はまだまだ多く残されており，ここでは今後の課題と展望について簡単にまとめておく．

（１）推論法

本論文では推論方式として分散型探索について述べてきたが，この手法は解が得られるまでしらみつぶし的に推論を進めるボトムアップ推論方式である．もう一つの方法としてトップダウン推論方式がある．トップダウン推論方式では問題を部分問題に分割し，それをエージェントに割り当て，その部分解を統合することにより推論を進める．トップダウン推論では，分割した目標間で生じる相互干渉の回避がメカニズムが重要となる[10]．

（２）表現法

第２章で述べた定式化では知識は状態と状態遷移を表す規則として表現していたが，人間にとって理解しやすく，また知識を表現しやすいものとするために構造化する必要がある．また，論理表現との関係も重要になる．

（３）アーキテクチャ

推論法，表現法を実証するために実際に分散型問題解決システムを構築することは重要である．このようなシステムを構築するためにはＬＡＮをはじめとする通信システム，分散形データベースシス

テムなどの手法を組み合わせる必要がある.

（４）評価

分散型問題解決システムを評価する要因には，各エージェントで行われる計算量とともに全体的な問題解決の速度に影響を及ぼすエージェント間の通信量がある．このような観点から各手法を評価する必要がある.

謝　辞

本研究に関して，直接理解あるご指導を賜り，常に励ましていただいた北橋忠宏教授に心から深謝致します．

大学院前期及び後期課程においてご教示，ご指導頂いた情報工学科の嵩忠雄教授，藤沢俊男教授，都倉信樹教授，鳥居宏次教授，谷口健一教授，産業科学研究所の角所収教授，豊田順一教授，大型計算機センターの宮原秀夫教授，並びに故高島堅助教授に心から感謝します．

本研究を通して直接ご助言，ご指導頂いた情報工学科の田村進一助教授，産業科学研究所の小川均講師に感謝します．

本研究に関して有益なご討論，ご助言頂いた沖電気工業株式会社山崎晴明氏，静岡大学工業短期大学部情報工学科新妻清三郎助教授，大阪大学基礎工学部情報工学科萩原兼一助教授，下条真司助手に感謝します．

種々の面でご助言，ご援助頂いた産業科学研究所平井誠助手，内尾文隆技官，工学部博士後期課程淡誠一郎氏に心から感謝します．

さらに，筆者の在学中，ご討論頂いた旧情報基礎論講座，産業科学研究所北橋研究室の方々に心から感謝します．

最後に，本研究完成のために常に励ましてくれた妻の美恵子に感謝します．

文 献

(1) R. Davis: "Report on the Workshop on Distributed AI" SIGART Newsletter, No.73, pp.42-52 (1980).

(2) R. Davis: "Report on the Second Workshop on Distributed AI", SIGART Newsletter, No.80, pp.13-23 (1982).

(3) M. Fehling and L. Erman: "Report on the Third Annual Workshop on Distributed Artificial Intelligence", SIGART Newsletter, No.84, pp.3-12 (1983).

(4) R.G. Smith: "Report on the 1984 Distributed Artificial Intelligence Workshop", AI Magazine, 6, 3 pp.234-243 (1985).

(5) R.G. Smith: "A framework for distributed problem solving", UMI Research Press, Ann Arbor (1981).

(6) V.R. Lesser and D.D. Corkill: "The distributed vehicle monitoring testbed, AI Magazine, 4, 3, pp.15-33 (1983).

(7) S. Cammarata, D. McArthur and R. Steeb: "Strategies of cooperation in distributed problem solving", Proceedings of the 8th International Joint Conference on Artificial Intelligence, pp.767-770 (1983).

(8) 山崎晴明: "分散型演えきデータベースシステム: SD^{3}とそのプロトコル", 情報処理学会論文誌, 26, 2, pp.288-295(1985).

(9) M. Suwa et al.: "Knowledge Base Mechanisms", Fifth Generation Computer Systems, North-Holland Publishing Company, pp.139-145 (1982).

(10) K. Konolige and N.J. Nilsson: "Multiple-agent

planning systems", The 1st Annual National Conference on Artificial Intelligence, pp.138-142 (1980).

(11) J.D. Yang, M.N. Huhns and L.M. Stephens: "An architecture for control and communications in distributed artificial intelligence systems", IEEE Transactions of System, Man, and Cybernetics, SMC-15 3, pp.316-326 (1985).

(12) W.A. Kornfeld: "Combinatoriarlly implosive algorithms", Communications of the ACM, 25, 10, pp.734-738 (1982).

(13) 徳田雄洋: "分散型アルゴリズムの基礎", 情報処理, 24, 4, pp.474-479 (1983).

(14) R.B. Banerji: "Artificial intelligence : a theoretical approarch", Elsevier North Holland, New York (1980).(高橋, ほか訳: "人工知能: コンピュータによるゲーム", 共立出版).

(15) K.M. Chandy and J. Misra: "Distributed Computation on Graphs: Shortest Path Algorithms", Communications of the ACM, 25, 11, pp.833-837 (1982).

(16) T. Cheung: "Graph Traversal Techniques and the Maximum Flow Problem in Distributed Computation", IEEE Transactions of Software Engineering, SE-9, 4, pp.504-512 (1983).

(17) N.J. Nilsson: "Principles of artificial intelligence", Tioga Publishing Co. (1980). (白井, 他訳: "人工知能の原理", 日本コンピュータ協会(1983)).

(18) 増沢, 萩原, 都倉: "辺故障を考慮したある分散アルゴリズム", 電子通信学会論文誌, J-69D, 10, pp.1394-1405,

(1986).

(19) R.G. Smith : "The Contract Net Protocol: High-level communication and control in a distributed problem solver", IEEE Transactions of Computer, C-29, 12, pp.1104-1113 (1980).

(20) 北村, 小川, 北橋: "分散型問題解決のため一通信方式", 情報処理学会知識工学と人工知能研究会資料, 52-9(1987).

(21) 相田, 田中, 元岡: "並列処理向き論理型言語の提案", 情報処理学会ソフトウェア基礎論研究会資料, 10-5 (1984).

(22) 小川, 北村, 田村: "混成型問題解決機ＰＳＡについて", 情報処理学会知識工学と人工知能研究会資料, 44-6 (1986).

(23) "ＰＣＮＥＴユーザーズガイド", ＮＥＣ.

(24) 北村, 谷中, 横山, 小川, 田村: "研究室内マイコンネットワークＡＩＤ－Ｎｅｔの作成", 情報処理学会第３１回全国大会, 7Q-2 (1985).

(25) 横山, 北村, 小川, 田村: "ＡＩＤ－Ｎｅｔにおけるダイナミックダウンロード機能について", 情報処理学会第３２回全国大会, 1D-4 (1986).

(26) 北村, 横山, 横山, 小川, 田村: "研究室内マイコンネットワークＡＩＤ－Ｎｅｔの作成", 情報処理学会関西ソフトウェア研究会資料 (1986).