# 分散問題解決のための波及型探索とその評価*

北村泰彦

1996年 2月 5日

**Abstract**

計算機システムの利用形態は集中から分散へと移行しつつあるが，さらに大規模な問題解決を行うために，エージェントと呼ばれる自律的な解決器の協調による分散問題解決が注目されている．従来の研究は特定のアプリケーションやアーキテクチャに基づく実験的なアプローチがほとんどであったが，本研究では汎用的な推論手法である波及型探索方式を提案している．本研究ではまず状態空間表現による分散問題解決の定式化を行い，分散問題解決を，探索空間が複数のエージェントに分割されているという前提のもとで，初期状態から目標状態までの経路探索と定義している．したがって解を求めるためには複数のエージェントの協力による分散探索が不可欠になる．波及型探索は従来の探索手法を分散システムのために拡張したもので，本稿では単一解と最適解を求めるアルゴリズムを提案し，その完全性，適格性，また解析的にアルゴリズムの計算複雑度，メッセージ複雑度，近似並列度を明らかにしている．さらに，より実際的な評価を行うために分散迷路探索問題を例題とし，波及型探索アルゴリズムの性能と迷路の複雑度，エージェント間の通信コストとの関係を明らかにしている．その結果，探索空間が広くなる問題に対しては波及型探索による並列性の向上が示されたが，通信コストの大きさが性能低下に大きな影響を与えることも明らかになった．

Computer systems have been evolving from centralized ones to distributed ones, and furthermore, cooperating systems, which consists of intelligent and autonomous agents cooperating with each other, are drawing attention to cope with large scale and complex problems. Most of the conventional approaches have been experimental ones based on specific application or architecture, but, in this paper, we propose a generic distributed inference scheme wich is called diffusing search. At first, we formalize distributed problem solving as a search process to find a path from the initial state to a goal state on a state space graph which is divided and assigned to each agent a priori, and propose two diffusing search algorithms to find a single path and an optimal path respectively. Then, we prove their completeness and admissibility and analyze their computational and message complexities and approximate speed-up. To evaluate their actual performance, we make simulation experiments on distributed maze problems and show its performance concerning the complexity of maze and the communication cost. As results, we show that the diffusing search is effective for problems whose search space becomes large, on the other hand, ineffective when the communication cost is high.

## 1 まえがき

計算機システムの利用形態は資源の分散管理，負荷の分散，システムの信頼性などの利点により集中から分散へと移行しつつある．しかしながら分散データベースなどの従来の分散システムは論理的には集中的な視点を保ったまま物理的な分散を図っている．今後，システムがさらに大規模化，複雑化してゆくときに，集中的視点を保つための通信量やシステムの柔軟性の面から，このようなアプローチには限界がある．そこで，分散システムを構成する各ノードに自律性を与えることにより物理的にも論理的にも分散化を行い，大局処理にはノード間の協調により対処する分散協調システムが近年注目を浴びている [DAI 91b].

一方，人間社会も一種の分散システムと見なすことができ，各個人は自律的で協調的な問題解決を日常的に行っている．このような人間社会をメタファとして分散協調システムを構築しようとするアプローチが分散人工知能 (Distributed Artificial Intelligence: DAI) あるいは分散問題解決 (Distributed Problem

* TR93-1R2: 情報処理学会論文誌に採録

Solving: DPS)[Bond and Gasser 88b, Huhns 87, Gasser and Huhns 89, DAI 91a, 石田 92] である．しかしながら，いくつかの例外 [横尾 92, MacIntosh *et al.* 91] を除いて，従来の研究は特定のアプリケーションやシステムに基づく実験的なアプローチがほとんどで，システムや解決手法の一般的な比較や評価を行うことが困難であった．そこで分散問題解決の定式化や汎用的な問題解決手法に関する研究の重要性が認識されるようになってきた [Durfee 91, 石田 92]．

従来の人工知能研究において問題解決が探索として定式化され [Banerji 83]，ヒューリスティックに基づく探索アルゴリズムが提案されてきた [Pearl 84] ように，本研究では分散問題解決を分散探索として定式化を行い，その汎用的な推論手法として複数のエージェント[1]の協力により系統的に経路探索を行う波及型探索 (diffusing search) アルゴリズムを示し，その定性的，定量的な性質を明らかにすることを目的としている．

本稿ではまず 2章において状態空間表現による分散問題解決の定式化とその分類について述べる．次に 3章においては単一解と最適解を求める波及型探索アルゴリズムを示し，その完全性や適格性といった定性的な性質を明らかにする．4章では波及型探索アルゴリズムの解析的評価を計算複雑度，メッセージ複雑度，近似並列度の面から行う．5章では実際的な評価を行うために，分散迷路探索のシミュレーションを通して通信コストを含めた波及型探索アルゴリズムの性能を示す．6章では問題分割の視点から従来の手法との違いを示し，最後に 7章において本研究のまとめとする．

# 2 分散問題解決の定式化

分散問題解決は複数のエージェント (解決器) による協調的な問題解決手法を研究の対象としている．そこでは各エージェントは個別に解決すべき問題を持っているのではなく，全エージェントに共通な単一の初期問題が存在すると仮定している[2]．ここで協調的とは，初期問題が単一のエージェントによってのみ解決されるのではなく，複数のエージェントの関与が必要であることを意味している [Smith and Davis 81]．

一般的な問題解決における問題の表現は OR グラフに基づく状態空間表現と AND/OR グラフに基づく問題置換表現に大きく分類される [Nilsson 73]． Lesser は AND/OR グラフ表現に基づく分散問題解決の定式化の試みを示している [Lesser 90, Lesser 91] が，本研究では以下に示す理由により分散問題解決の定式化を従来の状態空間表現に基づく問題解決の定式化 [白井 82, Banerji 83] を拡張することにより行なっている．

- 与えられた問題をどのように自動的に AND 分割するかに関する確立された一般的方法論が存在していない．多くの AND 分割はあらかじめ外部から与えられることを前提としているが，分割が明示的でない問題も多い．
- AND 分解の方法が与えられていたとしても，分散問題解決では副問題のエージェントへの割り当てが課題となる [Davis and Smith 83]．また，分割や割り当てが不適切であった場合の対処法に関しても研究の途上にあり，その一般的方法論は確立していない．

## 2.1 数学的準備

まず，数学的な準備としてグラフに関する基本的な用語を定義しておく．

グラフ (graph) は節点 (node) の集合 $N(\neq \emptyset)$ と（有向）辺 (directed link) の集合 $L(\subseteq N \times N)$ の組 $< N, L >$で表される．$N$が有限であるグラフを有限グラフという．また $(n, n) \notin L$ とする．グラフ$< N, L >$の部分グラフとは $N' \subseteq N, L' \subseteq L, L' \subseteq N' \times N'$を満たす$< N', L' >$のことである．

[1] 本稿ではエージェントとノードを等価な意味で用いている．

[2] それぞれのエージェントが個別の問題をもち，共通の制約に対して協調してそれらの達成を図るようなシステムのことをマルチエージェントシステム [Bond and Gasser 88a] と呼び，本稿では区別している．

あるグラフ$< N, L >$に対して，$(n_i, n_j) \in L$ であれば，$n_j$ は $n_i$の子 (child)，$n_i$ は $n_j$の親 (parent) と呼ぶ．節点の列 $(n_0, n_1, \ldots, n_m)$ $(m \geq 1)$ は $\forall k(0 \leq k \leq m-1) : \ell_{k,k+1}(= (n_k, n_{k+1})) \in L$ の時に $n_0$から $n_m$ への経路 (path) と呼び，$m$ を経路の長さ (length) と呼ぶ．辺にコスト (cost) が $c : L \rightarrow \mathbf{R}^+$(ただし $\mathbf{R}^+$は正の実数の集合) として与えられる場合には，経路 $(n_0, n_1, \ldots, n_m)$ のコストは $\sum_{k=0}^{m-1} c(\ell_{k,k+1})$ で与える．

次に，集合の記法に関して以下のような記法を用いる．集合 $A_1, A_2, \ldots, A_n$に対して，$n$ 項関係 $R \subseteq A_1 \times A_2 \times \ldots \times A_n$が定義されるとき，$e_1 \in A_1, \ldots, e_i \in A_i (i < n)$ に対して，$R(e_1, \ldots, e_i) = \{(e_{i+1}, \ldots, e_n) | (e_1, \ldots, e_n) \in R\}$ と定義する．

## 2.2 問題と解

問題と解は状態空間表現により以下のように定式化される．

**定義 1** *(問題と解)* 問題 *(problem)* は *4* 項組 $< S, O, s_I, G >$ により与えられる．ここで $S$は空でない状態 *(state)* の集合，$O \subseteq S \times S$はオペレータ *(operator)* の集合，$s_I \in S$は初期状態 *(initial state)*，$G \subseteq S$ は目標状態 *(goal state)* 集合と呼ぶ．組$< S, O >$は $S$を節点，$O$を辺と見なすことによりグラフとなり，状態空間グラフ *(state space graph)* と呼ぶ．状態空間グラフが有限な問題のことを有限問題 *(finite problem)* と呼ぶ．状態空間グラフ$< S, O >$において，初期状態 $s_I$から目標状態 $s_G \in G$ に至る任意の経路を問題の解 *(solution)* と呼ぶ．また，オペレータにコスト $c(> 0)$ が割り当てられているとき，解のコストはその経路のコストとして与えられる．また，あるコストの解が存在し，それよりも小さいコストの解が存在しなければ，その解は最適解 *(optimal solution)* であるという．

すなわち問題解決とは，状態空間グラフにおいて初期状態に適用可能なオペレータを次々に適用して目標状態に変換することであるといえる．従来の非人工知能的な問題解決はあらかじめ解決の手順が自明であり，プログラムによって表現されたその手順に従って処理を進める決定的な手続きであった．したがって，その状態空間グラフは初期状態から目標状態に至る一本の経路によって表現できる．それに対して人工知能が扱う問題解決では解決の手順をあらかじめ求めることが困難であり，試行錯誤的な処理の繰り返しが伴う非決定的な手続きとなる．したがってそのような問題解決は状態空間グラフにおける経路探索として定式化される．

## 2.3 エージェントとコミュニティ

分散問題解決は複数のエージェント (agent) による協調的な問題解決を扱うものである．本稿では以下，このエージェントの集合をコミュニティ(community) と呼び，$C$で表す．エージェントは計算プロセスであり，探索アルゴリズムの実行や他エージェントとのメッセージのやり取りが可能であるとする．

各エージェントは領域知識 (domain knowledge) と接続知識 (connection knowledge) を持つ． 領域知識とは，状態空間グラフに対して，そのエージェントが探索（問題解決）可能な領域をあらわすもので以下のように定義される．

**定義 2** *(領域知識)* エージェント $a$ の領域知識 $DK_a$は組$< S_a, O_a >$ で表す．$S_a$をエージェント $a$ の *(局所)* 状態集合，$O_a (\subseteq S_a \times S_a)$ をエージェント $a$ の *(局所)* オペレータ集合と呼ぶ．また，コミュニティ$C$の領域知識 $DK_C$は組$< S_C, O_C >$で表し，コミュニティを構成するエージェントの知識の和 $S_C = \bigcup_{a \in C} S_a, O_C = \bigcup_{a \in C} O_a$として定義する．

また，接続知識は他エージェントの領域知識との関係を示すもので以下のように定義される．

**定義 3** *(接続知識)* ある状態が複数のエージェントの領域知識に含まれているとき，その状態を接続状態 *(connective state)* と呼ぶ．また，その状態によってそれらのエージェントの領域知識は互いに接続 *(connect)* しているという．接続知識は局所状態とエージェントの組の集合 $CK_a \subseteq S_a \times C$で表され，$(s,b) \in CK_a$ ならば，エージェント $a$（の領域知識）はエージェント $b$ （の領域知識）と状態 $s$ で接続しているという知識を持っていることを意味している．

## 2.4 知識分散

分散問題解決では各エージェントにどのように知識が分散しているかによって問題解決に対するコミュニティの能力や性質は異なってくる．そこでそのような知識分散 (knowledge distribution) を定義する概念として，問題とエージェントとの関係を示す被覆性 (coverage)，エージェント同士の関係を示す接続性 (connectivity) と冗長性 (redundancy) の三つの概念を導入する．

**定義 4** *(被覆性)* 問題 $p =< S, O, s_I, G >$とエージェント $a$ の領域知識 $DK_a =< S_a, O_a >$に対して，$S \subseteq S_a$かつ $O \subseteq O_a$ならばエージェント $a$ の領域知識 $DK_a$は問題 $p$ をおおう *(cover)* という．コミュニティ$C$の領域知識 $DK_C$に関しても同様に定義する．

被覆性とは与えられた問題に対してエージェントあるいはコミュニティがそれを解決するのに十分な知識を保持しているかどうかという問題解決の可能性を示すものである．従来の（非分散）問題解決では単一のエージェントだけにおおわれる問題のみを扱ってきたといえるが，分散問題解決では一般に，単一エージェントだけではおおわれないが，コミュニティによりおおわれる問題を扱うことになる．

**定義 5** *(接続性)* エージェント $a$ の接続知識 $CK_a$ が以下を満たすとき完全 *(complete)* であるという．

$$\forall s \in S_a, \forall b \in C - \{a\} : s \in S_a \cap S_b \Leftrightarrow (s,b) \in CK_a.$$

すなわちエージェントの接続知識が完全であるとは，その接続知識がそのエージェントの接続状態と接続されるエージェントの全ての対を含んでいることを意味している．

接続性はエージェントの協調可能性を表しているといえる．すなわち接続性が不完全であれば，コミュニティとして問題解決が潜在的には可能であっても，その解をエージェントの協力によって導き出せないことを意味している[3]．

**定義 6** *(冗長性)* コミュニティ$C$に対して，もし $\forall a,b \in C : a \neq b \Rightarrow O_a \cap O_b = \emptyset$ を満たすならば，そのコミュニティは非冗長 *(non-redundant)* であるという．もし $\forall a,b \in C : O_a = O_b$を満たすならば，そのコミュニティは完全冗長 *(completely redundant)* であるという．それ以外の場合は部分冗長 *(partially redundant)* であるという．

冗長性はコミュニティの問題解決に対する効率や信頼性に関連する概念である．完全冗長なコミュニティはエージェントの持つ領域知識が完全に重複しているので，負荷分散による問題解決の並列性やエージェントの障害に対する信頼性が高くなる．一方で知識の変更に対する同一性の保証や問題解決時における冗長処理の除去を行なう必要がある．それに対して非冗長なコミュニティは領域知識の重複がない専門エージェントからなるコミュニティであり，局所的な処理は完全に独立している機能分散システムであるといえる．したがってシステム開発のためのモジュール性や拡張性が高く，また各エージェントの知識が頻繁に更新されるような場合の対処は容易になるが，単一のエージェントの障害は致命的になりうる．部分冗長なコミュニティはその中間的な性質を示しているといえる．

[3]接続知識が不完全な場合にはエージェント間のメッセージのやり取りにより接続知識の不完全性を補う方法が考えられる [Smith 80].

# 3　波及型探索アルゴリズム

分散問題解決では解くべき問題が与えられ，それに必要な領域知識が複数のエージェントに分散して存在する場合に，どのようにエージェントが協力して解を導き出すかが課題となる．本章では従来の探索アルゴリズムを分散環境に拡張し，単一解と最適解を求める波及型探索アルゴリズムとその完全性と適格性について示す．

## 3.1　コミュニティに対する仮定

波及型探索アルゴリズムについて述べる前に議論の前提となるコミュニティに関する一連の仮定をまとめておく．

- コミュニティは有限個のエージェントにより構成される．そのひとつのエージェントをクライアントエージェントと呼び，初期問題を持っているとする．その他のエージェントのことをワーカーエージェントと呼ぶ．
- コミュニティ内のエージェントは通信ネットワークにより結合され，各エージェントは任意の他のエージェントとメッセージのやり取りによってのみ情報交換ができる．メッセージは通信中に消失することはないが，その到着順序は保証されない．
- 問題は有限でコミュニティの領域知識によりおおわれる．また目標状態はアルゴリズム実行開始前に各エージェントに周知されているとする．
- 各エージェントは完全な接続知識を持っている．
- コミュニティは非冗長である．

## 3.2　単一解探索アルゴリズム

まず状態空間グラフにおける初期状態から目標状態までの単一解を求める波及型探索アルゴリズムを図1に示す．

一般に探索は初期状態から始めて，適用可能な状態にオペレータを繰り返し適用して目標状態に到達しようとする手法である [白井 82]．ここではオペレータ適用の候補となる状態のことを開状態と呼び，データベース $OP$に記録しておく．ある状態に対して適用可能なすべてのオペレータを適用して子状態の全てを発見することを展開する (expand) という．展開された状態のことを閉状態と呼び，同じ状態を二度展開しないようにデータベース $CL$ に記録しておく．開状態の集合の中から次にどの状態を展開するかは探索戦略関数 $SS$に委ねられている．各開状態は評価値$\hat{f}$により評価され，その値の最も小さいものが次の展開の対象となる．したがって，この評価値の設定によって縦型，横型，最良優先型，A* などの探索戦略を用いることができる．

波及型探索はこのような探索が複数のエージェントによって分散的に実行される．各エージェントは状態空間グラフの部分グラフのみの探索しか実行できないので，接続知識を用いたエージェントの協力により初期状態から目標状態までの解経路発見の保証をしなければならない．アルゴリズムはクライアントエージェントから，初期状態を局所状態に含む根 (root) エージェントに初期状態 $s_I$を与える探索依頼メッセージが送られることにより開始される．探索依頼メッセージを受け取ったエージェントはその状態がまだ発見されたものでなければ開状態に加える．各エージェントは開状態の集合が空でない間，探索戦略に従って探索を領域知識の範囲内で進める．接続状態を発見すればその接続知識に基づき，そこから先の探索を探索依頼メッセージを用いてその状態が接続しているエージェントに依頼する．いずれかのエージェント

が目標状態を発見すると解発見メッセージの全エージェントへの同報を行い，全エージェントの探索を停止させる．解は複数のエージェントの $PN$ と $RQ$ を逆にたどることにより得ることができる．

ここで，波及型探索の具体例を示す．図 2に状態空間グラフで表現された問題の例を示す．この例では $S = \{s1, \ldots, s15\}$，$O = \{o1, \ldots, o14\}$ である．初期状態を $s1$，目標状態を $s15$ とすると，解は $(s1, s4, s6, s8, s13, s15)$ となる．この問題は 4 つのエージェント $a, b, c, d$ からなるコミュニティの領域知識により覆われており，このコミュニティは非冗長である．それぞれの領域知識は $DK_a =< \{s1, s2, s3, s4, s5\}, \{o1, o2, o4\} >$，$DK_b =< \{s4, s6, s7, s8\}, \{o3, o5, o6\} >$，$DK_c =< \{s5, s9, s10, s11\}, \{o7, o8, o9\} >$，$DK_d =< \{s8, s11, s12, s13, s14, s15\},$ である． $s4, s5, s8, s11$ はそれぞれ $a$ と $b$, $a$ と $c$, $b$ と $d$, $c$ と $d$ との接続状態である．エージェント $a$ の接続知識 $CK_a$ は $\{(s4, b), (s5, c)\}$ のとき完全である．

この問題は $a, b, c, d$ のいずれのエージェントも単独では解経路全体を発見することができないので，エージェントの協力が必要になる．波及型探索アルゴリズムはクライアントエージェントからの探索依頼メッセージにより，初期状態を持つ（根）エージェント $a$ から探索が開始され，エージェント $a$ はエージェント $b$ への接続状態 $s4$ を発見し，エージェント $b$ に残りの探索を依頼する．同様にエージェント $b$ もエージェント $d$ への接続状態 $s8$ を発見し，残りの探索をエージェント $d$ に依頼する．最終的にエージェント $d$ が目標状態 $s15$ を発見することにより解経路が得られる．

次にアルゴリズムが解の存在する問題に対しては必ず解を得て終了するという完全性について議論する．波及型探索アルゴリズムは複数のエージェントにより実行される分散アルゴリズムであり，アルゴリズムの停止に関しては従来のものを拡張する必要がある．それぞれのエージェントの状態は，停止命令を実行した明示的な停止状態，アルゴリズムを実行している活性状態，そしてメッセージ待ちの状態である不活性状態に分類される．そして波及型探索アルゴリズムの停止を以下のように定義する．

**定義 7** 波及型探索アルゴリズムは通信ネットワークが定常状態（それ以降ネットワークで通信されるメッセージが存在しなくなる状態）になったときに停止したという *[萩原 90]*.

これは全エージェントが停止状態にあるだけでなくエージェントが不活性状態であっても，コミュニティ内にそれ以後変化がない場合にはアルゴリズムは停止していると見なすことである[4]．

問題が有限の場合，同一の状態を二度以上展開しないこと，また接続知識が完全で，メッセージは有限時間内に到着することが保証されるので単一解探索アルゴリズムの完全性を示すことができる．

**定理 8** 解が存在する有限問題に対して波及型単一解探索アルゴリズムはかならず解を発見し，終了する．
*(*証明*)* 文献 *[Kitamura and Okumoto 91]* 参照.

ここで問題が有限でない場合の考察を行う．3.1節に示されているように通信ネットワークにおいてメッセージ到着の順序が保証されない場合は解になりえない経路に関する探索依頼メッセージが無限に発生し，そのことが解経路の探索を妨げることがありうるので，完全性を保証することができない．もし通信ネットワークにおいてメッセージは有限時間内に到着するという前提がつけ加えられ，エージェント内では無限探索に落ち込まないような適切な探索戦略（例えば，横型や最良優先型）が用いられるなら完全性を保証することは可能となる．

## 3.3 最適解探索アルゴリズム

次に最適解を求める波及型探索について議論する．従来の非分散型のアルゴリズムとして代表的なものに Dijkstra 法 [野下 85] がある．そこでは初期状態から探索が開始され，初期状態からの最小コスト経路の確定した状態を次々と展開してゆくことによって目標状態までの最小コスト経路（最適解）を求めるよう

[4] 分散アルゴリズムでは問題を弱く解くという [萩原 90]．全エージェントが不活性状態になったかどうかを判定する分散アルゴリズムは文献 [Raynal 88] を参照されたい.

にしている．この手法では展開の対象となる状態は全て，初期状態からその状態までの最小コスト経路が確定しているので同一状態を二度以上展開することはない．ところが波及型探索においては各エージェントは非同期に探索を行い，通信ネットワークの仮定より探索依頼メッセージの到着順序を保証がすることができないので，展開の対象となる状態に対して，初期状態からその状態までの最小コスト経路が発見されているという保証ができず，同じ状態を複数回展開することが起こりうる．

このような点に対処した波及型最適解探索アルゴリズム (単一解アルゴリズムからの変更/追加分のみ) を図 3に示す．単一解アルゴリズムとの違いは状態 $s$ のコストを記録するデータベース $COST(s)$ を付加した点である．状態 $s$ のコストとはそれまでに発見された初期状態 $s_I$から $s$ までの経路のコストの中で最小のものを表している．また探索依頼メッセージを受け取った時や状態を展開した時には状態のコストが小さくなる場合に限り，$RQ$，$PN$ の変更を行う．最後に，目標状態が発見された場合にはその解のコストを解発見メッセージにより全エージェントに同報する．そのメッセージを受け取ったエージェントは解の最小コストを記録する変数 $MIN$を更新する．各エージェントは$OP$ 内の状態の最小コストが $MIN$の値以上になった時点で不活性状態になり，全エージェントが不活性状態になったときにアルゴリズムは終了する．

次に本アルゴリズムに関して有限の問題に解が存在すれば必ず最適解が発見できるという適格性 (admissibility) について証明する．まず補題としてアルゴリズムの停止性について証明する．

**補題 9** 有限問題に対して波及型最適解探索アルゴリズムは必ず停止する．

*(*証明*)* 探索が目標状態に到達しない場合を考えよう．本アルゴリズムでは同じ状態を二度以上展開する場合があるが，その回数は有限回である．なぜなら状態空間グラフの状態数 $|S|$ は有限であり，初期状態からある任意の状態までの経路の数は高々 $\sum_{i=0}^{|S|-2} {}_{|S|-2}\mathbf{P}_i$ で有限である [5]．したがって最悪の場合でも同一状態に対する再展開の回数はその値を越えることはないので，いずれのエージェントにおいてもいずれは展開すべき状態がない不活性状態となり，アルゴリズムは停止する．

**定理 10** 有限問題に解が存在するとき，波及型最適解探索アルゴリズムは必ず最適解を発見して停止する．

*(*証明*)* アルゴリズムの停止は補題 *9*より証明され，解の発見は定理 *8*と同様に証明されるので，ここでは最適解が必ず得られることを背理法を用いて証明する．まず最適解以外の解を発見してアルゴリズムが終了したと仮定する．最適解経路を $(s_0, s_1, \ldots, s_n)$ で表わし，そのコストを $c_{opt}$とする．最適解経路は発見されていないので，この状態の中で少なくとも一つはいずれのエージェントの $CL$ にも含まれていないことになる．そのような状態の中で最も添え字の小さいものを $s_i$とする．経路と状態のコストの定義から $COST(s_i) < c_{opt}$である．ところでアルゴリズムはすでに解が得られて停止しているので$COST(s_i) \geq MIN$が成り立つ．したがって $c_{opt} > MIN$となるが，これはすでに得られている解のコスト $MIN$が最適解のコストより小さく，$c_{opt}$が最適解のコストであることと矛盾している．したがって最適解は必ず発見される．

# 4 波及型探索アルゴリズムの解析的評価

本章では波及型探索アルゴリズムの性能を計算複雑度，メッセージ複雑度，並列性について解析的な考察を加える．波及型探索アルゴリズムの性能は状態空間グラフや知識分散の形態に依存し，一般的な解析を行うことは困難であるので，問題と知識分散に以下のような仮定をおく．

- コミュニティ内のワーカーエージェント数は $|C|$ で任意の二つのエージェントは一つの接続状態により接続されているとする．

[5] ${}_x\mathbf{P}_y$は $x$ 個から $y$個の順列の個数である．

- いずれのエージェントの領域知識も $n$ 個の局所状態，$m$ 個の局所オペレータから成り立っているとする．
- 領域知識内の任意の二状態間の最小コスト経路は必ず領域知識に含まれるとする．すなわちその経路は他のエージェントの領域知識内を経由することがない．この性質を経路の局所最適性と呼ぶことにする．

## 4.1 単一解アルゴリズムの複雑度

まず局所探索における計算複雑度を示す．単一解アルゴリズムでは最悪の場合，初期状態から接続する領域知識内の全ての状態を一度だけ展開し，全てのオペレータを適用するので計算複雑度は $O(m)$ となる．次にメッセージ複雑度に関しては，まずクライアントエージェントが 1 個，根エージェントが $|C|-1$ 個の探索依頼メッセージを送る．最悪の場合，それぞれのエージェントは探索依頼メッセージに対して，そのメッセージを送信したエージェントと自分自身以外の全てのエージェントに $|C|-2$ 個の探索依頼メッセージを送るので，その総数は $1+(|C|-1)+(|C|-1)\cdot(|C|-2)$ である．したがって，メッセージ複雑度は $O(|C|^2)$ となる．

## 4.2 最適解アルゴリズムの複雑度

最適解アルゴリズムにおいては状態は $COST$ 値変更のために何度も再展開される可能性がある．この原因としてはエージェント内の局所探索によるものと，エージェント間でのメッセージ到着順序の非決定性によるものが考えられる．ここでは Dijkstra 法 [6] 等を局所探索のために用いると仮定して，後者の影響のみを考慮することにする．

まずメッセージ複雑度に関しては探索依頼メッセージの数のみを対象とする．最初にクライアントエージェントは 1 個，初期状態をもつ根エージェントは $|C|-1$ 個の探索依頼メッセージを送信する．経路の局所最適性を仮定をすれば，根エージェントを除いたエージェントは最悪の場合，展開を引き起こす探索依頼メッセージを $\sum_{i=0}^{|C|-2} {}_{|C|-2}\mathbf{P}_i$ 回受信することになる．そして探索依頼メッセージのそれぞれに対して，自分自身と探索依頼メッセージを送ったエージェント以外の $|C|-2$ 個のエージェントに探索依頼メッセージを新たに送るので，探索依頼メッセージの総数は $1+(|C|-1)+(|C|-2)\cdot(|C|-1)\cdot\sum_{i=0}^{|C|-2} {}_{|C|-2}\mathbf{P}_i$ となり，メッセージ複雑度は $O(|C|!)$ となる．

図 4(a) の $|C|=4$ のコミュニティを用いて，例を示そう．ここで初期状態をもつ根エージェントは a であるとする．クライアントエージェントは 1 個の探索依頼メッセージを a に送り，a は b,c,d にそれぞれ探索依頼メッセージを送る．探索経路はループすることはないので，メッセージの到着順序が最悪となる場合，展開を引き起こす探索依頼メッセージの流れは図 4(b) のようになる．このような探索依頼メッセージを受信すると最悪の場合，2 個の新たな探索依頼メッセージが生じる．したがってメッセージの総計は $1+3+2\cdot 3\cdot\sum_{i=0}^{2} {}_2\mathbf{P}_i = 34$ となる．

次に，局所探索の計算複雑度であるが，Dijkstra 法を用いれば一つの探索依頼メッセージに対して $O(n^2)$ となる[7]．さらに再展開による局所探索の繰り返しを考慮しなければならないが，1 エージェントあたりの局所探索数は先の議論より最悪の場合，$\sum_{i=0}^{|C|-2} {}_{|C|-2}\mathbf{P}_i$ であり，その結果，計算複雑度は $O(n^2\cdot(|C|-2)!)$ となる．

[6] 探索戦略関数 $SS$ において状態の評価値を $\hat{f}(s)=COST(s)$ とする．

[7] 計算複雑度は開状態とそのコストを記録するデータ構造にも依存し，ここでの値は一次元配列を用いた場合である．データ構造として二分木を用いれば $O(m\log n)$ となる [野下 85]．

### 4.3 アルゴリズムの並列度

波及型探索アルゴリズムの並列度を近似的に求めてみる．波及型探索アルゴリズムでは初期状態を持つ根エージェントから探索が徐々に他のエージェントに広がってゆく．この探索にかかわったエージェントの数を $|C_a|$ とする．探索はいくつかのエージェントの局所探索を経て目標状態に到達することになるが，この解の経路上のエージェントの数を $|C_s|$ とする．エージェント内で実行される局所探索時間を一様に $T_p$，探索依頼メッセージのエージェント間転送のための通信時間を一様に $T_c$ とすれば，解を得るまでの時間は $|C_s|T_p + (|C_s| - 1)T_c$ となる．もし一台で全ての探索を行うとすれば，その時間は $|C_a|T_p$ となるので並列度は

$$\frac{|C_a|T_p}{|C_s|T_p + (|C_s| - 1)T_c} \tag{1}$$

で表される．これは，$T_c \ll T_p$ のような密結合システムならば，その並列度は $|C_a|/|C_s|$ となることを意味している．また，$T_p \ll T_c$ のような疎結合システムにおいては，$|C_s|$ が大きくなるにつれ，その並列性は通信オーバヘッドにより大きく減少することがわかる．ここで $T_p$ は局所探索の計算複雑度とエージェントの計算速度，$T_c$ はメッセージ複雑度とネットワークの通信速度，$|C_s|$ は知識分散に依存する要因であり，厳密な解析的評価はきわめて複雑になる．そこで次章では分散迷路探索を例題として実験的にアルゴリズムの性能評価を行う．

## 5 波及型探索アルゴリズムの実験的評価

前章では波及型探索の解析的評価を与えたが，最適解を求めるアルゴリズムに関しては計算複雑度，メッセージ複雑度がともに階乗オーダーであり，悲観的な結果となっている．しかしながら前章での解析は最悪時でのものであり，本章では迷路探索を例題として波及型探索アルゴリズムの実際的な評価を行う．ここでは迷路の複雑度やネットワークの通信コストに応じた性能の変化，またヒューリスティック探索の効果について示す．

### 5.1 分散迷路探索問題

分散迷路探索問題 [Ishida and Korf 91] は図 5に示すような格子状迷路において，入口から出口までの経路を求める問題であり，$(x, y)$ 座標を状態，座標間の移動をオペレータ，入口を初期状態，出口を目標状態と見なせば，状態空間表現により自然に表現できる．迷路の複雑度をランダムに発生させた障害物の比率と見なし，0%から 40%まで変化させている．すなわち迷路の複雑度が大きくなるほど，解経路を発見することが困難になる．実験には解経路の存在する $120 \times 120$ の 10 個の迷路を用いている．初期状態は $(1, 1)$，目標状態は $(120, 120)$ である．エージェントのオペレータとしては上下左右のみの移動を許し，斜め方向には移動できないとしている．また，オペレータ適用のコストを一律に 1 としている．エージェント数は 1,4,9,16,25,36,64,100 台の場合があり，それぞれ格子状に領域知識を分割している．例えば，エージェント数 100 台の場合には，エージェントは $10 \times 10$ の格子状に配置され，それぞれのエージェントは $12 \times 12$ の局所状態に接続状態を加えた迷路を探索することになる．

### 5.2 実験 1

まず通信コストを 0 とした場合においてエージェント数と迷路の障害率に応じたシミュレーション評価を行う．探索戦略としては状態 $s = (x, y)$ の評価値としてヒューリスティックを加え，単一解探索の時は $\hat{f}(s) = \sqrt{(x_g - x)^2 + (y_g - y)^2}$，最適解探索の場合は $\hat{f}(s) = COST(s) + \sqrt{(x_g - x)^2 + (y_g - y)^2}$ となる A*を用いた．ここで目標状態を $(x_g, y_g)$ とする．エージェント数 (No. of Agents) と迷路の障害率 (Obstacle

Rate) に応じた単一解探索と最適解探索の探索時間 (Search Time) を図 6と図 7に示す．探索時間は 10 個の迷路の平均である．

単一解探索においては探索戦略が強力なので，迷路の複雑度が小さい場合は探索経路は初期状態から目標状態までのほぼ一直線になり，探索空間の広がりは小さい．したがって多数のエージェントで探索しても，その並列性は上がらない．しかし迷路が複雑になるにしたがって探索空間が広がり，並列性が高くなる．

最適解探索の場合は得られた解が最適であるかどうかを確定する必要があるので，最適解のコスト以下の経路は全て探索されることになる．したがって迷路全体に探索が広がるので並列性は高くなる．最適解探索では迷路が複雑になると迷路中の障害物の割合が増加し，それだけ探索すべき状態空間自体が相対的に狭くなるので，最適解確認のための時間は短くなり並列性が低下している．

以上の実験結果から，探索空間が大きくなるような問題に対しては複数エージェントによる波及型探索による効果が大きくなることが明らかになった．ここで最適解探索の速度向上 (Speed-up) を示すグラフを図 8に示し，前章で示した近似並列性の議論をこの実験結果に適用してみることにする．迷路の複雑度が 0%の最適解探索では探索領域は全エージェントにわたるが，その解経路は初期状態から目標状態までの対角線で与えられる．探索にかかわったエージェント数を $|C|$ とすれば，解経路上のエージェント数は近似的に$\sqrt{|C|}$であり，式 (1) から近似並列性は$\sqrt{|C|}$となる．その結果 (approx.) をグラフ上に示している．迷路の障害率が小さい場合，近似並列性以上の速度向上が得られているのは探索戦略で用いているヒューリスティックの効果である．このことは評価値として $\hat{f}(s) = COST(s)$ を用いた最良優先戦略 (BF) との比較を表す表 1を参照することにより明らかになる．探索は図 9 に示すように，BF の場合は目標状態に向かって横一列に，A*の場合は目標状態に向かって尖ったような形状で探索が進められてゆく．迷路の形状は正方形であるので，1 台で探索を行う場合には探索戦略の違いはそれほど影響が出ないが，多数のエージェントで探索を行う場合には A*の方が初期状態から目標状態にいたる対角線上のエージェントにより速く探索が波及し，探索時間が短縮される．

### 5.3 実験 2

次に通信にコストがかかる場合の影響を示す．迷路は最も簡単なもの (障害率 0%) を用いた．全エージェントは 1 本のバス型通信路により接続されているとし，近似的に図 10で示されるような通信モデルを用いる．すなわち，各エージェントで送信されるメッセージは 1 本のキュー (queue) に入れられ，通信コストパラメータにより定められる時間毎[8]にメッセージが取り出され，受信エージェントに渡される．したがって，このモデルにおいては，エージェントで送信されるメッセージが単位時間あたり増加すると，キューが長くなり，メッセージ送信から受信までの通信遅延は大きくなる．通信コスト (Comm. Cost) は 0 から 5 まで変化させた最適解探索の結果を図 11に示す．

この結果は複数のエージェントにより探索する場合の並列性と通信オーバヘッドのトレードオフの関係を表しているといえる．通信コストが小さい場合は通信オーバヘッドの影響が小さいので多数のエージェントで探索した方が効果的である．ところが通信コストが増加するにつれ通信オーバヘッドの影響が大きくなり，エージェント数が増加するにしたがって著しくなる．最適解探索では通信コストによって探索時間を最小にするエージェント数が異なり，通信コストが大きくなるにしたがって最適なエージェント数は少なくなる．波及型探索においての単位時間あたりの通信量は一般に探索が進むにつれて増加し，このことは最適なエージェント数を解析的に求めることを困難にしている．

[8] エージェントが一つの状態の展開に要する時間を一律に 1 単位時間とした．

# 6　考察と関連研究

本章では波及型探索と従来の分散問題解決手法との違いを問題分割の視点から述べるが，その前に問題分割の定義を明らかにしておく．2章でも述べたように一般に問題は 4 項組$< S, O, s_I, G >$で表されるが，ここでは $S, O$に関しては自明，また $G$ は単一の要素 $s_G$からなると仮定して，二項組$< s_I, s_G >$で表すことにする．問題$< s_I, s_G >$が与えられたとき，問題分割とはその問題から二つの副問題$< s_I, s_M >$と$< s_M, s_G >$を得ることであるが[9]，このような中間状態 $s_M$ は一般に問題解決以前には自明ではない．

契約ネットプロトコル (contract net protocol)[Smith 80] では，問題はいくつかの独立な副問題にあらかじめ分割されていることが前提となっており，副問題を適切なエージェントに割り当てる通信プロトコルを定義している．また分散制約充足 (distributed constraint satisfaction) アルゴリズム [横尾 92] でも，問題は不完全であってもあらかじめ分割されていることが前提となっている．この場合，問題は$< s_I, S_M >$と$< S_M, s_G >$に分割されており，中間状態候補が複数存在する（$S_M$は中間状態候補の集合）．ここで分散制約充足とは，これらの副問題を共通の中間状態を得る（制約を充足する）ようにして分散的に解決することである，と見なされる．したがって以上の手法はいずれも問題はあらかじめ副問題に分割可能であることが前提となっていたといえる．

一方で DVMT[Lesser and Corkill 83] などの分散型黒板モデルによる問題解決では問題分割は必ずしも前提とされていない[10]．各エージェントは機会主義的 (opportunistic)[Erman *et al.* 81] な局所推論をもとに得られたで部分解 (仮説 ) を互いに交換し，それらを組み合わせることによりボトムアップに全体解を導き出している．この手法は部分解が全体解に自然に収束してゆくような問題には適しているが，一般的には解を得るために多量の部分解交換が必要になり，推論の完全性や最適性に関する保証は困難である．

本稿で議論した波及型探索は分散型黒板モデル同様，問題の分割をあらかじめ与えておくことを前提としていない．ただしエージェントは機会主義的に探索を行うのではなく，与えられた探索戦略に従って系統的に探索を進めてゆくので，探索の完全性や解の最適性の保証が可能となっている．

最後に，波及型探索の前提となる領域知識への分割の可能性について議論しておこう．5章で示したような分散迷路探索問題は空間的な領域分割が可能であり，波及型探索に適した問題であるといえる．このような空間的分割が可能な問題にはこれまでの分散問題解決で扱われてきたセンサネットワーク [Lesser and Corkill 83] や通信網経路設定問題 [Conry *et al.* 88] などがあげられる．一方，同じような探索問題であっても，8 パズルなどのような問題は空間的な分割は容易ではなく，それに代わる分割手法としてはオペレータの種類（例えば，タイル移動の上，下，左，右）による機能分割が考えられる．この場合には状態空間は細切れにされ，波及型探索では多量のメッセージ交換が必要になり，適切な領域分割法とはいえない．したがって波及型探索の性能は領域分割の方法に依存するといえる．

# 7　むすび

本稿では状態空間表現に基づく分散問題解決の定式化を行い，その汎用的な推論手法として単一解と最適解を複数のエージェントの協力により求める波及型探索アルゴリズムを示し，その定性的，定量的な性質を明らかにした．また実際問題における評価をおこなうために，分散迷路探索のシミュレーションを通して迷路の複雑さ，エージェント数，通信コストとその探索性能との関係を明らかにした．一般に，探索空間が広くなる問題に関しては並列性が高くなるという結果が得られたが，通信コストの大きさが波及型探索アルゴリズムの性能を大きく低下させることも明らかになった．

波及型探索では探索が複数のエージェントで分散して実行されており，従来の非分散探索手法と比較して，(1) 局所探索では状態空間グラフの部分グラフしか探索することができない，(2) 局所探索は各エー

[9] 一般には二つ以上の問題に分割可能であるが，ここでは簡単のため二つの場合に限っている．
[10] ただし領域知識はあらかじめ分割されている．

ジェントで非同期に実行される，という特徴がある．(1) のゆえに波及型探索では必然的にエージェント間での協力が不可欠であり，そのためにエージェント間でメッセージ通信が行われる．また (2) のために最適解を求めるような要求に対しては最悪の場合，エージェント間では階乗オーダーの通信量が必要になるということが導かれ，波及型探索では通信オーバヘッドへの対処が重要な課題になる．そのために，(a) 全てのメッセージを対等に扱うのではなく，探索依頼の重要性によって優先度付きのメッセージ通信を行う [北村 88]，(b) エージェント自らが探索依頼の重要度に応じて選択的に通信を行う [北村 94]，などの手法に関して研究中である．また，(c) あらかじめ分散しているエージェントにどのように状態空間グラフを割り当てるか，(d) どの程度通信を減少させれば，どの程度の近似解が得られるか，(e) 問題分割に関する知識が与えられた場合にどのように波及型探索と組み合わせてゆくか，(f) コミュニティが非冗長でない場合にどのようにすればさらに並列性を向上できるか，などの点も興味ある課題であり今後の研究課題としたい．

## 参考文献


[Banerji 83] Ranan B. Banerji. 人工知能. 共立出版, 東京, 1983.

[Bond and Gasser 88a] Alan H. Bond and Les Gasser. An analysis of problems and research in DAI. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[Bond and Gasser 88b] Alan H. Bond and Les Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[Conry *et al.* 88] Susan E. Conry, Robert A. Meyer, and Victor R. Lesser. Multistage negotiation in distributed planning. In Alan H. Bond and Les Gasser, editors, *Readings in Distributed Artificial Intelligence*. Morgan Kaufman Publishers, Inc., San Mateo, California, 1988.

[DAI 91a] Special section on distributed artficial intelligence. *IEEE Transactions on System, Man, and Cybernetics*, Vol. 21, No. 6, November/December 1991.

[DAI 91b] 特集 · 協調と分散. コンピュータ科学, Vol. 1, No. 3, 1991.

[Davis and Smith 83] Randall Davis and Reid G. Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, Vol. 20, pp.63–109, 1983.

[Durfee 91] Edmund H. Durfee. The distributed artificial intelligence melting pot. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 6, pp.1301–1306, November/December 1991.

[Erman *et al.* 81] L. D. Erman, F. Hayes-Roth, V. R. Lesser, and D. R. Reddy. Hearsay-ii 音声認識システム：不確定性の解決のための知識の統合. bit 別冊コンピュータサイエンス, pp. 53–91. 共立出版, 1981.

[Gasser and Huhns 89] Les Gasser and Michael N. Huhns, editors. *Distributed Artificial Intelligence, Volume II*. Morgan Kaufman Publishers, Inc., San Mateo, California, 1989.

[Huhns 87] Michael N. Huhns, editor. *Distributed Artificial Intelligence*. Morgan Kaufman Publishers, Inc., Los Altos, California, 1987.

[Ishida and Korf 91] Toru Ishida and Richard E. Korf. Moving target search. In *Proceedings of the 1991 International Joint Conference on Artificial Intelligence*, pp. 204–210, 1991.

[Kitamura and Okumoto 91] Yasuhiko Kitamura and Takaaki Okumoto. Diffusing inference: An inference method for distributed problem solving. In S. M. Deen, editor, *Cooperating Knowledge Based Systems 1990*, pp. 79–94. Springer-Verlag, 1991.

[Lesser 90] Victor R. Lesser. An overview of DAI: Viewing distributed AI as distributed search. *Journal of Japanese Society for Artificial Intelligence*, Vol. 5, No. 4, pp.392–400, 1990.

[Lesser 91] Victor R. Lesser. A retrospective view of FA/C distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 6, pp.1347–1362, November/December 1991.

[Lesser and Corkill 83] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *The AI Magazine*, Vol. 4, No. 3, pp.15–33, 1983.

[MacIntosh *et al.* 91] Douglas J. MacIntosh, Susan E. Conry, and Robert A. Meyer. Distributed automated reasoning: Issues in coordination, cooperation, and performance. *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 6, pp.1307–1316, November/December 1991.

[Nilsson 73] Nils J. Nilsson. 人工知能. コロナ社, 1973.

[Pearl 84] Judea Pearl. *Heuristics*. Addison-Wesley, Reading, Massachusetts, 1984.

[Raynal 88] Michel Raynal. *Distributed Algorithms and Protocols*. John Wiley & Sons, Chichester, 1988.

[Smith 80] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, Vol. C-29, No. 12, pp.1104–1113, December 1980.

[Smith and Davis 81] Reid G. Smith and Randall Davis. Framework for cooperation in distributed problem solving. *IEEE Transactions on System, Man, and Cybernetics*, Vol. SMC-11, No. 1, pp.61–70, January 1981.

[横尾 92] 横尾真, エドモンド H. ダーフィ, 石田亨, 桑原和宏. 分散充足制約による分散協調問題解決の定式化とその解法. 電子情報通信学会論文誌 D-I, Vol. J75-D-I, No. 8, pp.704–713, 1992.

[石田 92] 石田亨, 桑原和宏. 分散人工知能 (1):協調問題解決. 人工知能学会誌, Vol. 7, No. 6, pp.945–954, 1992.

[萩原 90] 萩原兼一. 分散アルゴリズム. 人工知能学会誌, Vol. 5, No. 4, pp.430–440, 1990.

[白井 82] 白井良明, 辻井潤一. 人工知能. 岩波書店, 東京, 1982.

[北村 88] 北村泰彦, 小川均, 北橋忠宏. 分散型問題解決における問題割当てのための一通信方式. 電子情報通信学会論文誌 D, Vol. J71-D, No. 2, pp.439–447, 1988.

[北村 94] 北村泰彦, 寺西憲一, 辰巳昭治, 奥本隆昭. 波及型探索における通信制御法とその評価. マルチエージェントと協調計算 III（出版予定）. 近代科学社, 1994.

[野下 85] 野下浩平, 高岡忠雄, 町田元. 基本的算法. 岩波書店, 東京, 1985.

- メッセージ
  - <search; $s$>:探索依頼メッセージ．状態 $s$ からの探索を依頼する．
  - <found>:解発見メッセージ．目標状態が発見されたことを知らせる．
- データベース
  - $OP(\subseteq S_a)$: 展開される状態の候補となる開状態の集合．
  - $CL(\subseteq S_a)$: すでに展開された閉状態の集合．
  - $PN(\subseteq S_a \times S_a)$: 二つの状態間のポインタの集合．$(s_i, s_j) \in PN$ ならば，$s_i$から $s_j$への経路が発見されていることを示す．
  - $RQ(\subseteq S_a \times C)$: 探索を依頼したエージェントの記録．$(s, b) \in RQ$ は状態 $s$ からの探索がエージェント $b$ により依頼されたことを示す．
- 関数
  - $SS$: $OP$の中から次に展開する状態を選択する関数．$s' = SS(OP)$ なら$\hat{f}(s') = \min_{s \in OP} \hat{f}(s)$ が成り立つ．ただし$\hat{f}(s)$ は状態 $s$ の評価値である．
- 手続き

＜クライアントエージェント＞

**State1** 初期化．

**Step1** <search; $s_I$>を初期状態 $s_I$を局所状態に持つエージェント $a_I$に送る．

**State2** <found>を受信した時．

**Step1** 停止．

＜ワーカーエージェント $a$ ＞

**State1** <search; $s$>をエージェント $b$ から受信した時．

**Step1** もし $s \notin CL \cup OP$なら，$s$ を $OP$に加える．$(s, b)$ を $RQ$ に加える．

**State2** <found>を受信した時．

**Step1** 停止．

**State3** $OP \neq \emptyset$ の時．

**Step1** $s \leftarrow SS(OP)$．$CL$ に $s$ を加える．$OP$から $s$ を除く．

**Step2** もし $s \in G$ なら<found>を全てのエージェントに送る．停止．

**Step3** 各 $s' \in O_a(s) \cap O(s)$ に対して，

(a) 全ての $c \in CK_a(s')$ に対して<search; $s'$>を送る．

(b) $s' \notin CL \cup OP$なら $OP$に $s'$，$PN$に $(s, s')$ を加える．

図 1: 単一解探索アルゴリズム

Single solution algorithm.

| 探索戦略 | エージェント数 | 展開状態数 | 探索時間 | 速度向上 |
|---|---|---|---|---|
| A* | 1 | 14163 | 14164 | 1.0 |
| | 100 | 14379 | 1296 | 10.9 |
| BF | 1 | 14400 | 14401 | 1.0 |
| | 100 | 14400 | 1450 | 9.9 |

表 1: 探索戦略の比較

Comparison of search strategies.

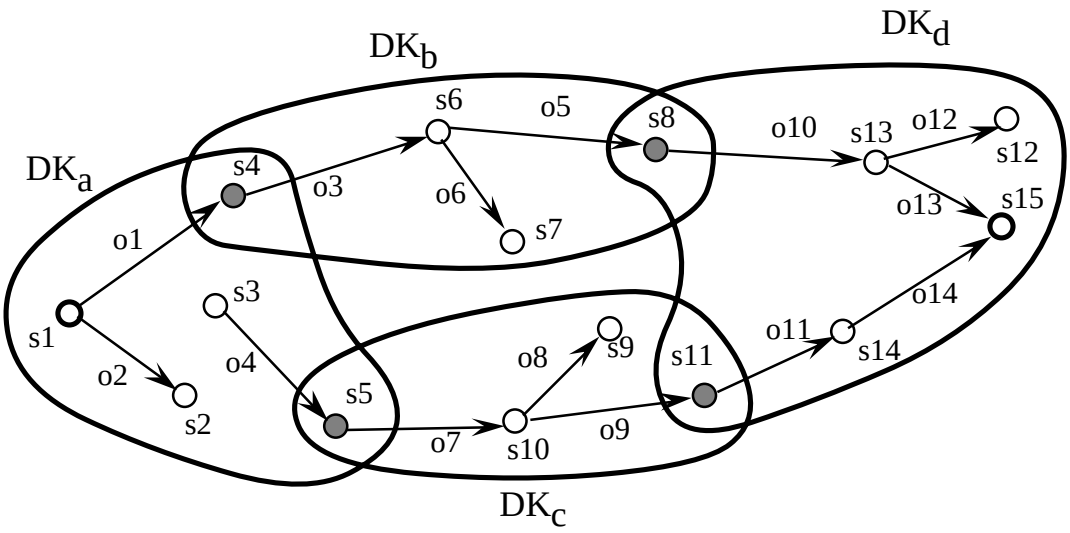



図 2: 状態空間グラフ
A state space graph.

- メッセージ (変更)
  - <search; $s, c$>: 探索依頼メッセージ．状態 $s$(コスト $c$) からの探索を依頼する．
  - <found; $c$>: 解発見メッセージ．コスト $c$ の解が発見されたことを知らせる．
- データベース (追加)
  - $COST(\subseteq S_a \times \mathbf{R})$: 状態のコストの記録．コストの初期値は$+\infty$.
  - $MIN(\subseteq \mathbf{R})$: それまでに発見された解の最小コストの記録．初期値は$+\infty$.
- 手続き (変更)

  <クライアントエージェント>

  **State1** 初期化.

  **Step1** <search; $s_I, 0$)>を初期状態 $s_I$を局所状態に持つエージェント $a_I$に送る．

  <ワーカーエージェント $a$ >

  **State1** <search; $s, c$>をエージェント $b$ から受信した時.

  **Step1** もし $s \notin CL \cup OP$なら，$s$ を $OP$に加える．$(s, b)$ を $RQ$ に加える．

  **Step2** もし $s \in CL \cup OP$かつ $COST(s) > c$ なら，$COST(s) = c$ とする．また，$s \in CL$ の場合は，それを除き，$OP$に加える．$(s, b') \in RQ$ なら，それを削除する．$(s', s) \in PN$ なら，それを削除する．$(s, b) \in RQ$ とする．

  **State2** <found; $c$>を受信した時.

  **Step1** $MIN > c$ なら，$MIN = c$ にする．

  **State3** $OP \neq \emptyset$ あるいは $OP$中の状態の最小コストが $MIN$ より小さい時.

  **Step1** $s \leftarrow SS(OP)$．$CL$ に $s$ を加える．$OP$から $s$ を除く．

  **Step2** もし $s \in G$ なら $MIN = COST(s)$ として，<found; $COST(s)$>を全てのエージェントに送る．

  **Step3** 各 $s' \in O_a(s) \cap O(s)$ に対して，$COST(s') > COST(s) + c(s, s')$ なら，$COST(s') = COST(s) + c(s, s')$ として，以下を実行する．

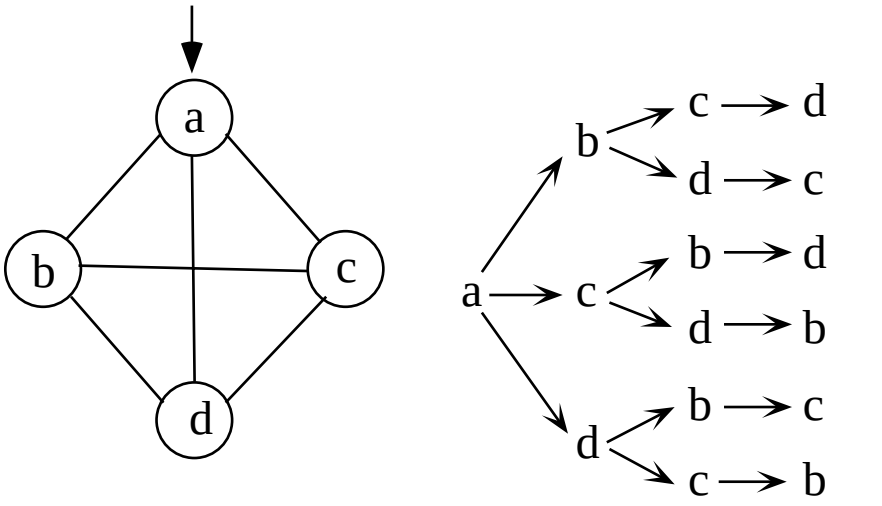
  (a) 全ての $b \in CK_a(s')$ に対して<search; $s', COST(s')$>を送る．

  (b) $s' \notin CL \cup OP$なら $OP$に $s'$，$PN$に $(s, s')$ を加える．

  (c) $s' \in CL$ の場合は，それを除いて $OP$に加える．$(s', b') \in RQ$ なら，それを除く．$(s'', s') \in PN$なら，それを除き $(s, s') \in PN$とする．

図 3: 最適解探索アルゴリズム

Optimal solution algorithm.




(a) A community (b) Search message flow

図 4: 例
Example.

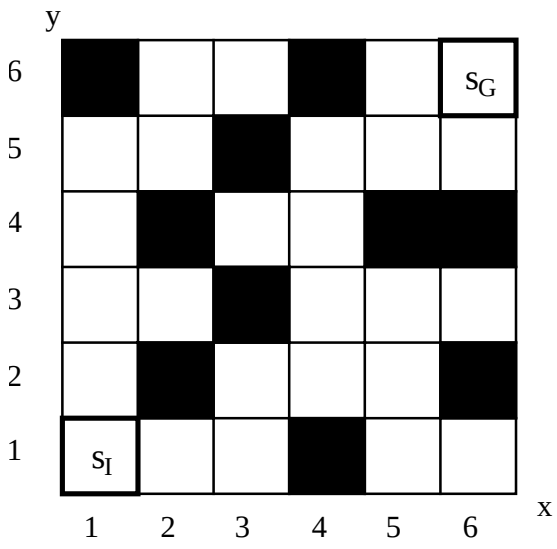



図 5: 6 × 6 の迷路
6 × 6 maze.

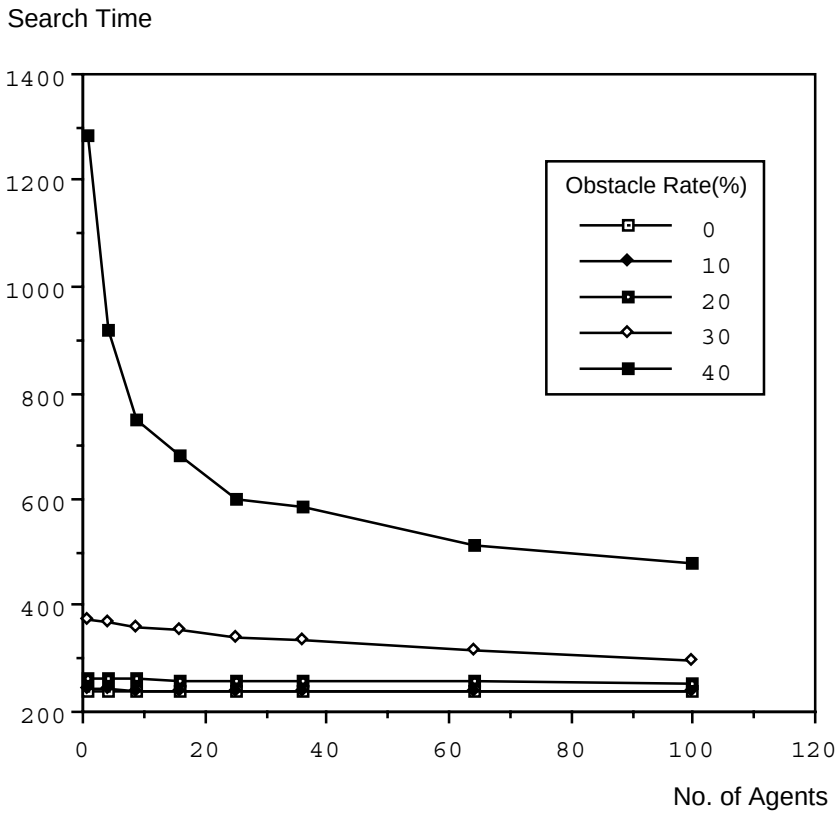



図 6: 単一解アルゴリズムにおける探索時間
Search time of single solution algorithm.

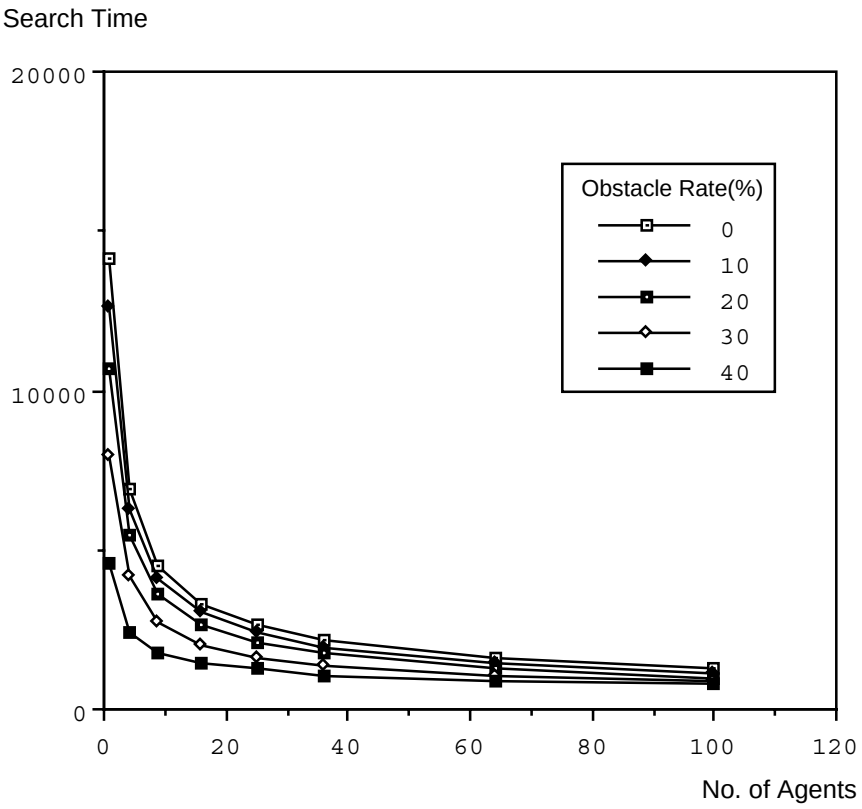



図 7: 最適解アルゴリズムにおける探索時間
Search time of optimal solution algorithm.

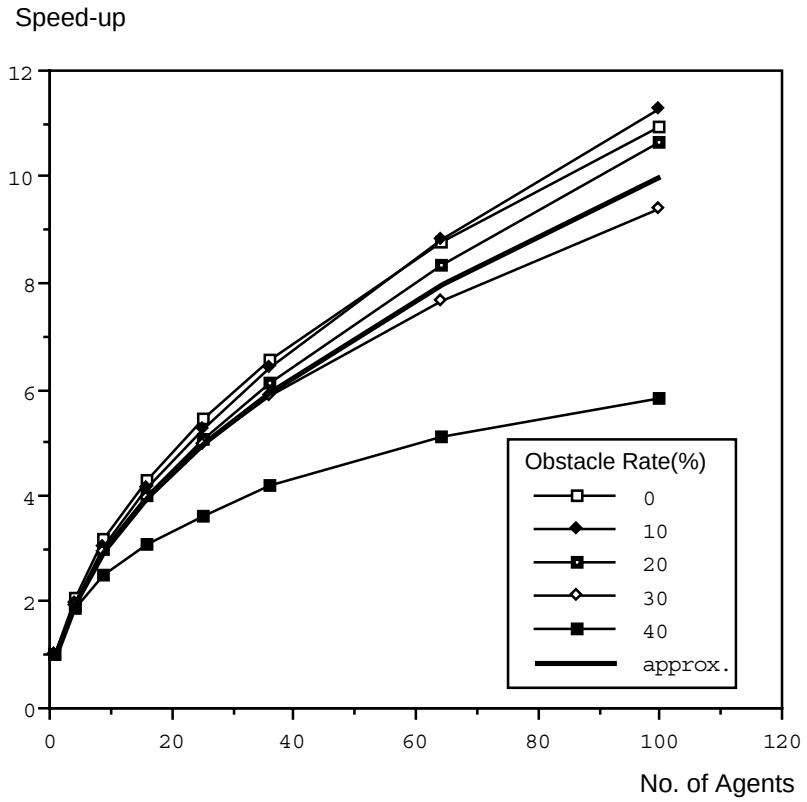
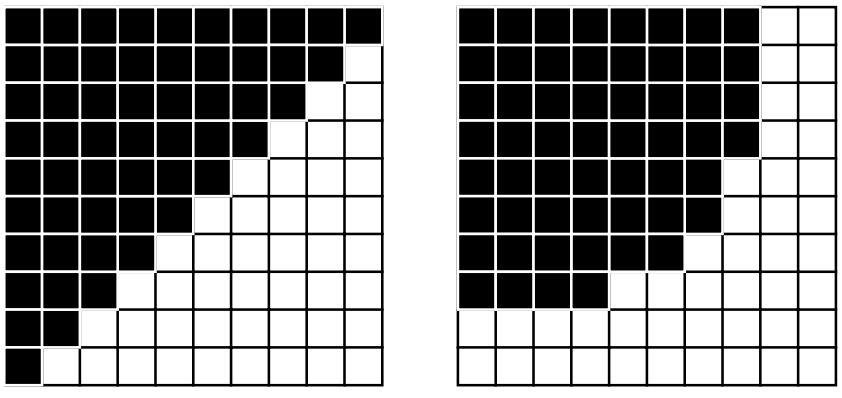



図 8: 最適解アルゴリズムにおける速度向上
Speed-up of optimal solution algorithm.



(a) BF　　(b) A*

図 9: 波及型探索の過程
Processes of diffusing search.

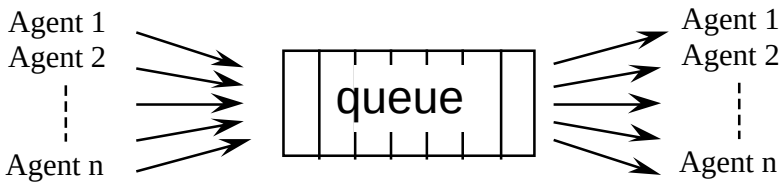



図 10: 通信路のモデル
Model of communiation channel.

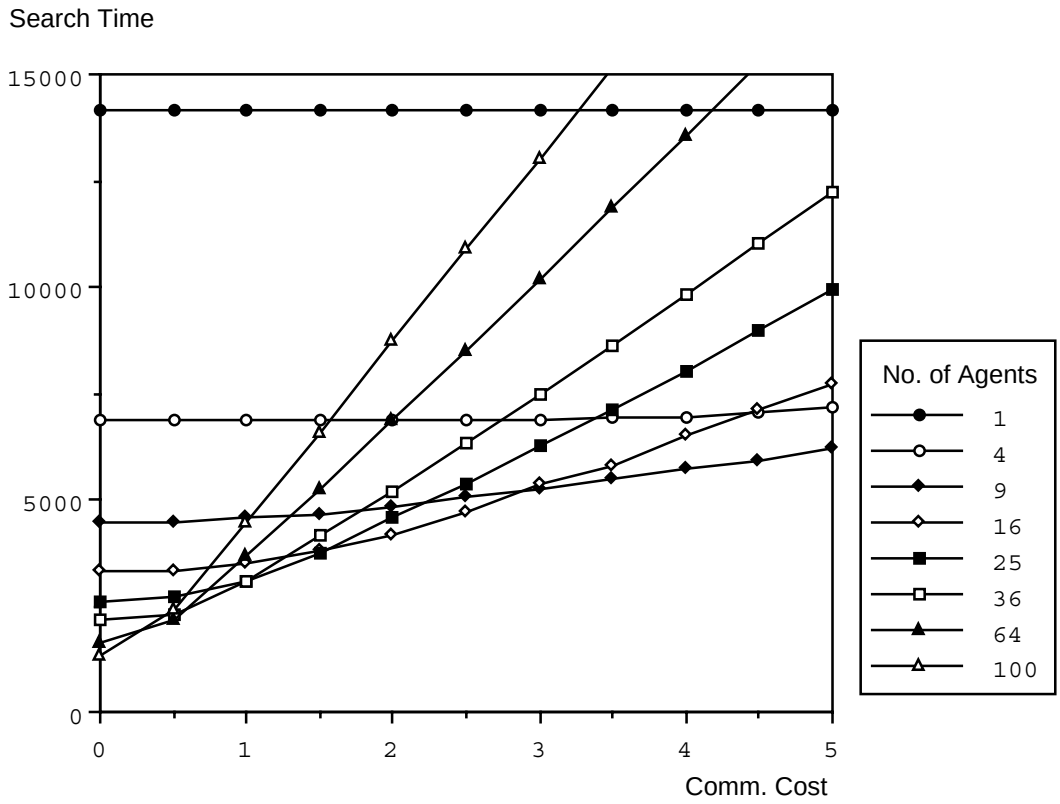



図 11: 通信コストを含めた最適解探索時間
Search time of optimal solution algorithm including communication cost.