モバイルエージェント系における分散制約充足

沼澤 政信　栗原 正仁　能登 正人

本論文では，モバイルエージェント系における分散制約充足について，いつ・どのような目的でエージェントの移動が考えられるのか基本的な考察をする．その中から，実ネットワーク中を流れる総通信量を抑制する目的で移動する場合に焦点を当て，簡単な計算モデルを構成し，計算機シミュレーションにより，このモデルに基づく移動が総通信量に与える効果について定量的な評価をする．分散制約充足アルゴリズムとしては分散 breakout アルゴリズムを仮定し，シミュレーションにより総通信量を測定した．その結果，以下のことがわかった．

1. 総通信量は，一般にリンク数の増加とともに急激に増加し，ピークに達した後急激に減少する．その後，緩やかに増加した後に緩やかに減少する．
2. エージェントサイズが，上限として定義した s_n よりも十分小さいときには，移動したほうが総通信量が概ね削減できる．逆に，エージェントサイズが十分大きいときは，移動により総通信量はかえって増加する．
3. 総通信量がピークとなる困難な問題に対しては，エージェントサイズがある程度大きくても総通信量を抑制できる．

1　はじめに

近年のネットワーク技術の急速な発達により，コンピュータの利用形態は，単体で使用するというよりも，インターネットや LAN などのネットワークを介して使用する環境になった．そのような分散環境に対応した分散型ソフトウェアを柔軟に構築するためにエージェント技術が注目され，さらに最近ではそれを発展させたモバイルエージェントと呼ばれるホスト間を移動するエージェント技術が有効であると考えられている [1][2][3]．モバイルエージェントは，途中のタスクの内容を保持したまま別のホストに移動してタスクを継続することができるため，ネットワーク上のコンピュータ資源を有効利用することが可能である．よく知られているモバイルエージェントシステムとしては，General Magic 社による Telescript [4]，IBM による Aglets [5]，東芝による Plangent [6] などがある．

このような状況に対応して人工知能の分野においても，知的エージェントの実現基盤の一つとして知られている制約充足技術が，分散制約充足技術の研究に移行してきている [7]．そこでさらに，分散型ソフトウェアの構築を考慮し，分散制約充足の計算モデルもモバイルエージェントに対応するように拡張することは自然な研究の流れである．しかし，そのような研究例はまだみられない．エージェントが移動しようがしまいが，いわゆる位置透過性があるシステムにおい

Distributed Constraint Satisfaction for Mobile Agents

NUMAZAWA Masanobu，小樽商科大学商学部社会情報学科，Department of Information and Management Science, Faculty of Commerce, Otaru University of Commerce

KURIHARA Masahito，北海道工業大学工学部情報デザイン学科，Department of Information Design, Faculty of Engineering, Hokkaido Institute of Technology

NOTO Masato，神奈川大学工学部電気電子情報工学科，Department of Electrical, Electronics and Information Engineering, Faculty of Engineering, Kanagawa University

ては，エージェント間のメッセージ交換の媒体が異なるだけで，メッセージの内容や交換のプロトコルが異なるわけではないので，抽象的あるいは論理的な意味では両者に差異はない．しかし，実装レベルでは，実ネットワーク中を流れる総通信量，延いては，実行効率の面で大きな違いとなりうる．

本論文ではそのような研究の第一歩として，まず，いつ・どのような目的でエージェントの移動が考えられるのか基本的な考察をし，その中から，実ネットワーク中を流れる総通信量を抑制する目的で移動するケースに焦点を当て，簡単な計算モデルを構成する．そして，計算機シミュレーションにより，このモデルに基づく移動が総通信量に与える効果について定量的な評価をする．

2 章では，モバイルエージェントと分散制約充足について概説し，その特徴を述べる．3 章では，考えうる種々の移動方式を簡単に考察した後，通信量を移動のトリガとする計算モデルを導入し，その評価のためのシミュレーション実験について述べる．4 章では，実験結果に基づき，この計算モデルの特徴と有効性を議論する．5 章では，まとめと今後の課題を述べる．

2　モバイルエージェントと分散制約充足

2.1　モバイルエージェントシステム

モバイルエージェントは，コンピュータ間の自律的移動能力を持つプログラムであり，エージェント自体が利用者の代理人としてネットワーク上を自律的に移動しながら特定のタスクを遂行する．そのため，利用者はネットワークに逐一接続して作業する必要がない．通常のエージェントでも，それぞれのホスト間で遠隔通信機能を使ってホスト間でのメッセージ通信が可能であるが，ネットワークが不安定で間欠的であったり，低速もしくは費用の高い通信路を使用しなければいけないといったケースが考えられるなど問題点も多い．なお，モバイルエージェントは AI 技術の関連で議論されている他のエージェント技術とは異なり，必ずしもインテリジェンスを持たないことが多いが，本研究で扱うモバイルエージェントは，分散制約充足という限定ながらも AI の基礎的な計算処理を能動的に実行し，移動先もそれ自身により選択し，移動することができるように設計されている．

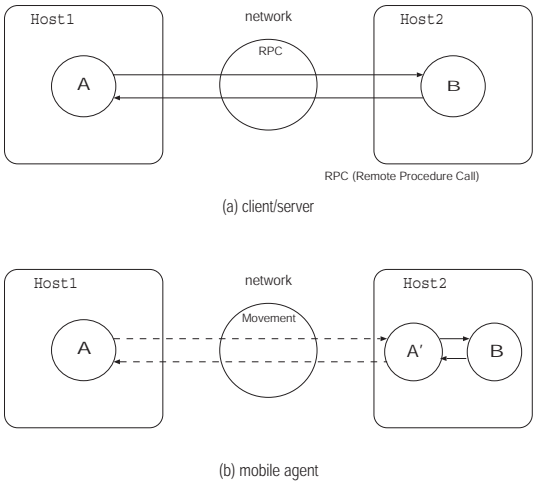

図 1　処理形態の違い

モバイルエージェントを用いてコード転送し，それぞれのホストでその転送されたコードを実行するためには，セキュリティ保護，障害対策，トラフィック増加，管理の手間の増加といった問題が複雑化する．しかしながら，不安定で間欠的な通信路を常時使用する必要がなく，通信コストの削減が可能となるなどメリットが大きい．その理由は，従来のクライアント／サーバ方式とモバイルエージェント方式の二つの処理形態を比べることにより，容易に理解することができる．例えば，二つの異なるホストにいるエージェント間で 1000 回の要求を発行し，それに対する応答があるとき，クライアント／サーバ方式では，1000 回の要求と応答が Host1 と Host2 間のネットワーク上を往復することになる（図 1(a)）．一方，モバイルエージェント方式（図 1(b)）では，エージェントが 1 往復するだけで 1000 回の要求と応答は，Host2 上でローカルに行うだけで済む．この 1000 回が，非常に費用の高い通信路であったり，要求の回数が 1 万回，10 万回と増えていった場合を想定すると，移動して処理を行う効果は高いと想像できる．

また，この処理形態の違いにより，もう一つ重要な結果が生じる．クライアント／サーバ方式では，毎回の要求と応答がネットワーク上を行き来するため，ホスト間のコネクションは，常時維持しなければならな

いのに対して，モバイルエージェント方式では，エージェントが移動するときだけコネクションを確立すればよい．これは，電話回線でモデムを利用する場合の通信や，無線を利用しているためコネクションが不安定な携帯電話，移動体通信などの通信形態にとっては非常に有利である．

2.2 分散制約充足

制約充足問題（Constraint Satisfaction Problem: CSP）[8] は，離散値をとるいくつかの変数に割り当てられる値の組合せのうち，与えられた制約をすべて満たす組合せを発見する探索問題である．分散制約充足問題[9][10] は，制約充足問題の変数と制約が複数のエージェントに分散された問題とみなすことができ，さまざまな問題が分散制約充足問題として定式化されている[11][12][13][14]．制約充足問題では，初期状態において一つのアルゴリズムが全体の問題を見渡すことができるのに対し，分散制約充足問題の特徴は，初期状態はもちろんのこと，任意の時点においてそれぞれのエージェントは全体の状態を知らないことである．このため，各エージェントがどのような手順で，どのような情報を交換し合えば，全体として制約を満たす解が発見できるかが問題となる．

制約充足問題を解くアルゴリズムは，厳密解法である木探索アルゴリズムおよび近似解法である反復改善型アルゴリズムの二つに大きく分類される[15]．前者は整合性をとりながら部分解を拡張して探索木をたどり完全な解を求める．後者は変数に制約を満足しない値を割り当てた状態から制約違反を局所的に改善していくものであり，横尾，平山[16] によって提案された反復改善型分散制約充足アルゴリズム（分散 breakout）が効果的な状態空間探索法の一つとして注目されている．

3 モバイルエージェント間の分散制約充足

本章では，モバイルエージェントと分散制約充足の二つの技術を結びつけてモバイルエージェント間での制約充足方式を提案する．はじめに，考えうるいくつかの移動方式を列挙し，次に，そのうちの一つである通信量に基づく移動方式を詳細化した計算モデルを構築した後，それを評価するためのシミュレーション実験の条件を設定する．

3.1 種々の移動方式

現在，モバイルエージェントを分散制約充足に導入した例はまだないので，研究の第一歩として，いつ・どのような目的で移動したらよいのかを検討する必要がある．以下に，タイミングと移動の目的として考えられる基本的な移動方式を述べる．

1. 時間による移動

   一定時間，他のエージェントから何のメッセージ応答もないときに移動する．現在エージェントが存在しているホストが過負荷である，または何らかの理由でプラットフォームの状態が不具合であるかもしれないと考えられるときに移動するため，本移動方式は，ネットワークで問題を解くときのロバスト性を高める効果が期待できると考えられる．
2. 準局所最適時の移動

   エージェントが準局所最適状態になったときに移動する．準局所最適の状態を抜け出すには多くのメッセージ交換が必要になる場合がある．本移動方式は，そのような場合，その負荷を軽減し得る可能性を持つ．
3. メタレベルからの移動要求による移動

   アプリケーションに依存して，自分が求めるエージェントに連絡をとり移動する．本移動方式は，タスクとして分散制約充足だけを行う研究レベルのエージェントシステムには付加されるものではない．しかし，このタスク機能を一部として包含する現実のアプリケーションにおいては「メタレベル」での要求によって移動する必要が生ずることもあり得ると考えられる．
4. 計算負荷による移動

   自分が存在しているホストマシンの負荷が所定の値を超えたときに移動する．本移動方式では，資源の効率的な利用が期待できると考えられる．
5. 通信量による移動

   メッセージの累積量が所定の量を超えたときに，一方のエージェントが他方のエージェントの

いるホストマシンへ移動する．本移動方式は，非常に汎用的な単純な方法であり，通信コストの削減が期待できると考えられる．

以下では,「通信量による移動」に基づく考え方で具体的に計算モデルを構築し，基盤となる分散制約充足アルゴリズムとして分散 breakout アルゴリズム [16] を採用してシミュレーションを実施し，移動が通信量に与える効果について評価する．

3.2　通信量をしきい値とする移動方式

任意の分散制約充足アルゴリズムが与えられたとする．そのアルゴリズムは，必ずエージェントが他のエージェントとメッセージ交換しながら制約充足のための計算を実行しているようにモデル化されていると仮定する．我々は，各エージェント i が現在自分のいるホストと異なるホストにいるエージェント j とメッセージ交換（送信，受信）する部分のプログラムコードに手を入れ，エージェントにこれまでの通信量の累積を把握させる．そして，それがあるしきい値を超えれば，エージェント i が他のホストに移動して，その後本来の計算を続行するような計算モデルを構築したい．そのような制限の下でもさらにいくつかのバリエーションが考えられるが，本論文ではこれから述べるような最も単純なものの一つを考察の対象としたい．

エージェント間で交換するメッセージ 1 つあたりのサイズ（メッセージサイズ）は簡単のため固定されていると仮定し，それを 1 と正規化して通信量の単位とする．エージェント i について，

q_{ij}：　エージェント j とのメッセージ累積量
Q_i：　総メッセージ累積量
C_i：　総通信量
t_i：　しきい値

とそれぞれ表し，初期値の設定は，$q_{ij} \leftarrow 0$，$Q_i \leftarrow 0$，$C_i \leftarrow 0$，$t_i \leftarrow$ エージェントサイズ とする．エージェントサイズは，エージェントの大きさ（すなわち，プログラムコードおよびエージェントの状態を表すデータの量）であり，エージェントが一回移動するのにかかる通信量でもある．簡単のために，これも固定されていると仮定する．移動条件の初期しきい値は，エージェントが移動するための初期値である．直観的に，メッセージ通信量がエージェントサイズより相対的に小さいうちは移動することによってかえって大きな通信量の増加をまねくので，しきい値の考えうる最小の値という意味で初期値をエージェントサイズと等しくした．以上の準備のもとで，エージェント i がエージェント j とメッセージ交換したときに実行すべき処理は以下の手順に従う．

1.　$q_{ij} \leftarrow q_{ij}+$ メッセージサイズ;
2.　$Q_i \leftarrow Q_i+$ メッセージサイズ;
3.　$C_i \leftarrow C_i+$ メッセージサイズ;
4.　もし，$Q_i > t_i$ ならば以下の処理（**4.1**〜**4.4**）をする;
　4.1　$j \leftarrow \max_j\{q_{ij}\}$ を与える j;
　4.2　エージェント j のいるホストへ移動する;
　4.3　$C_i \leftarrow C_i+$ エージェントサイズ;
　4.4　$t_i \leftarrow 2t_i$;
5.　分散制約充足アルゴリズムの処理;

最初に，しきい値を満たす前は，各エージェント間でメッセージ交換を行いながら，メッセージサイズを q_{ij}，Q_i，C_i に累積していく．その後，メッセージ累積量がしきい値を超えたときに，エージェントは，メッセージ交換したすべてのエージェントの中で個々のメッセージ量が最も多いエージェントのいるホストに移動する．このとき，総通信量にエージェントサイズを累積する．なお，メッセージ量の最大値をもつエージェントが複数の場合は，その中から非決定的に選択し移動先を決める．しきい値は移動するたびに 2 倍にしていく．これは，データ構造の分野で配列やハッシュ表のサイズを動的に拡張する際にも用いられる考え方の一つとほぼ同様で，単純ながらも自然な考えである．

本研究では，基盤となる分散制約充足アルゴリズムとして分散 breakout アルゴリズムを採用するが，システムを実ネットワーク上に実装するのではなく，性能の事前評価を目的とし，1 台のマシン上でのシミュ

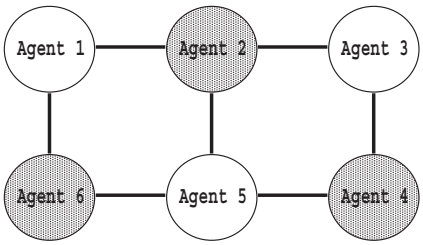

図 2　分散グラフ色塗り問題の例

レーション・システムとして実現している．

3.3　シミュレーション実験

分散制約充足問題の代表的な例題として分散グラフ色塗り問題がある．グラフ色塗り問題とは，任意のグラフと色数が与えられ，グラフ上の隣り合うノードを異なる色で塗るものである．このグラフのノードをエージェントとみなすことで，分散グラフ色塗り問題のエージェントネットワークが得られる．色数 2，エージェント数 6，リンク数 7 の分散グラフ色塗り問題の例および解を図 2 に示す．

3 色の分散グラフ色塗り問題をランダムに発生させ，メッセージサイズを 1 として表 1 に示す条件で，通信量に関する移動のシミュレーション実験を行う．

表 1　実験内容

	エージェント数	リンク数	エージェントサイズ
実験 1	30	30〜 270	10，70, 300, ∞
実験 2	60	60〜1100	50, 300, 500, ∞
実験 3	90	90〜2600	100, 300, 500, ∞
実験 4	120	150〜4500	100, 400, 600, ∞
実験 5	150	150〜7000	100, 500, 900, ∞

実験では，リンク密度 0〜100%の範囲でリンクの本数をパラメータとして設定し，各エージェント数（30〜150）に対してリンク数を変化させていき，各エージェントサイズでの通信量をそれぞれ求める．リンク数ごとに 100 回ずつ問題を解き，そのときの平均値をとる．なお，この問題では，リンク密度 25%前後付近の値のところで最も難しい問題が存在することが知られている [17] [18]．便宜上，エージェントを移動させない（固定した）ケースをエージェントサイズ ∞ で表す．

4　実験結果と考察

シミュレーション実験の結果として実験 1，実験 3 および実験 5（エージェント数が 30，90 および 150）のときの総通信量のグラフをそれぞれ図 3 〜 図 5 に示す．

はじめに，エージェント数が十分多い実験（図 4，図 5 ）において，比較の基準となるエージェントサイズが ∞（すなわち移動しないケース）のグラフの特徴を確認する．リンク数が増加していくと，ある狭い範囲で急激に総通信量が増加し，そのピークを超えると急激に減少している．これはいわゆる「相転移」などと呼ばれる現象としてよく知られている [19]．その後，総通信量は緩やかに増加に転じた後，再度，緩やかに減少する．

次に，このグラフを他の（移動する場合の）エージェントサイズのケースと比べてみる．エージェントサイズが十分小さい場合には，リンク数にかかわらず，移動しない（固定した）ときの総通信量よりも移動したときの総通信量の方が概ね削減できていることがわかる．逆に，直観的にも明らかなことだが，エージェントサイズがあまりにも大きい場合は，移動することによりその移動に必要な大きな通信量が加算されるため，総通信量は移動しないときよりもかえって大きくなってしまう．さらに，重要なこととして，リンク数が 250（図 4 ）付近（図 5 では 400 付近）の最も難しい問題による通信量のピークが，いずれのエージェントサイズの場合も，∞ のケースより低く抑えられていることがわかる．これは，今回設定した移動基準が，過大な通信量を抑制するように働くことによるものと考えられる．

エージェントサイズがある程度小さければ移動の効果があり，逆にある程度を超えて大きければ移動は逆効果となることがわかった．そこで，移動が効果的でありうるようなエージェントサイズの上限を求めてみる．エージェント数に加えてリンク数が既知として与えられたときは，エージェントサイズをパラメータとして今回のような実験をして，移動しないケースと比較すれば上限は自明な方法で実験的に求めら

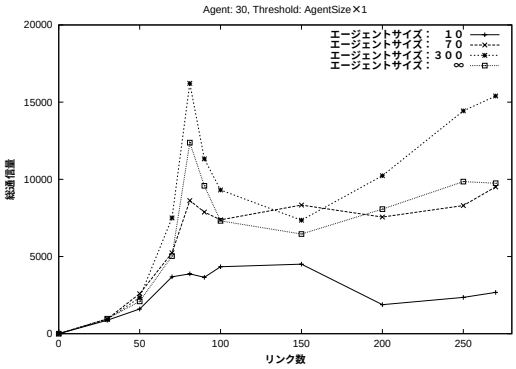

図 3　実験 1（エージェント数：30）

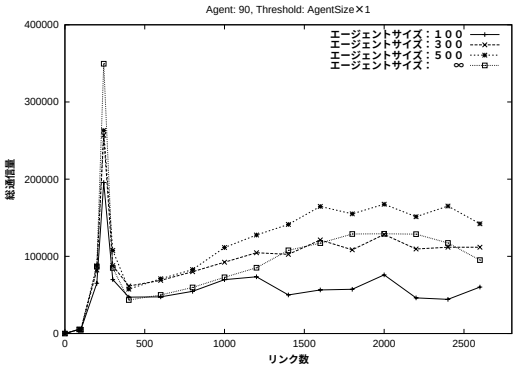

図 4　実験 3（エージェント数：90）

れる．しかし，分散環境においては一般にエージェントは問題の全体を知らないので，エージェント数のみは知っているとして，リンク数を知らない場合の上限を定義してその概数を試算しておくことが興味深い．そのようなヒューリスティクスはあまり厳密に論じても意味が薄いので，本論文では非常に単純なものだけを以下に述べる．

はじめに，エージェント数が n，エージェントサイズが s のときの総通信量をリンク数 ℓ の関数として $C_{n,s}(\ell)$ で表す（図 3 〜 図 5 の一本一本の折れ線がそのような関数の実例を表している）．そこで，$s=\infty$ の関数との平均的な差を以下の式で定める．

$$D_{n,s}=\frac{1}{L}\sum_{\ell=1}^{L}[C_{n,s}(\ell)-C_{n,\infty}(\ell)]$$

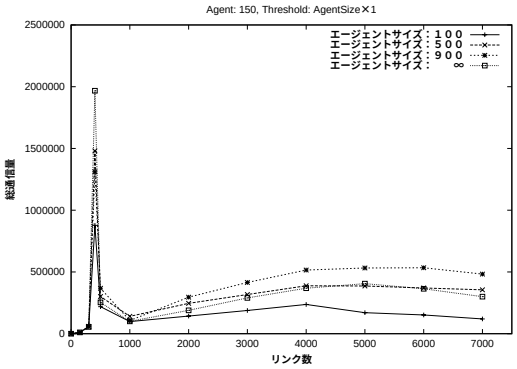

図 5　実験 5（エージェント数：150）

ただし，$L=n(n-1)/2$ はリンク数の上限値である．今回の実験から求めた $D_{n,s}$ の概算値の一部を表 2 に示す．

表 2　$D_{n,s}$ の計算結果

s	エージェント数 n				
	30	60	90	120	150
10	−4220	−21800	−61500	——	——
50	−1520	−14500	−51400	−124000	——
100	1180	−7270	−38600	−107000	——
200	3230	5640	−16700	−71900	−94200
300	——	14000	2730	−35100	−59400
400	——	18500	18400	−5380	−28900
500	——	——	29500	26200	17300

表 2 からわかるように，エージェント数を固定したとき，エージェントサイズ s が十分小さいときは $D_{n,s}<0$ であることから，移動が効果的であるといえる．逆に，s が十分大きいときは $D_{n,s}>0$ であることから，移動は効果的でないといえる．そこで，先ほど論じた上限として，$D_{n,s}<0$ を満たす最大の s の値を採用する．その値は n 毎に定まることから，s_n と表すことにする．表 2 から補間によって求めた s_n の近似値を表 3 および図 6 に示す．

表 3　s_n の近似値

n	30	60	90	120	150
s_n	75	156	286	417	463

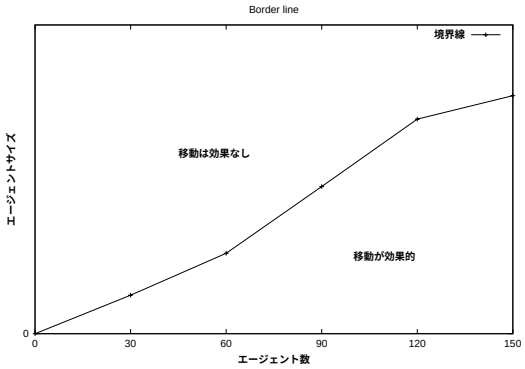

図 6　移動の境界線

きわめて大まかな指針として，リンク数が未知の場合，メッセージサイズを 1 としたときのエージェントサイズが，エージェント数のおよそ 3 倍程度以内のときは，移動による通信量の削減が期待できる．エージェントを実装するクラスファイルを各ホストに事前に置いておくなど，実装上の工夫でこの条件を満たせば，通信量の削減という理由だけで，エージェントを移動させる正当な理由となり得る．

ただし，エージェントサイズがこの指針より大きいときに本方式が役に立たないというわけではない．リンク数が既知で，特に，計算量的に「難しい問題」を発生させるようなリンク数のときには，多少エージェントサイズが大きくても，それを大きく上回るメッセージ量を削減できるので効果が大きい．また，3.1 節で述べたような種々の理由で移動そのものが効果を発揮することもあり得る．

5　おわりに

本研究では，モバイルエージェント系における分散制約充足について，いつ・どのような目的でエージェントの移動が考えられるのか基本的な考察をし，その中から，総通信量を抑制する目的で移動するケースに焦点を当て，簡単な計算モデルを構成した．次に，分散制約充足アルゴリズムとして分散 breakout アルゴリズムを仮定し，シミュレーションにより総通信量を測定した．その結果，以下のことが確認できた．

1. 総通信量は，一般にリンク数の増加とともに急激に増加し，ピークに達した後急激に減少する．その後，緩やかに増加した後に緩やかに減少する．
2. エージェントサイズが，上限として定義した s_n よりも十分小さいときには，移動したほうが総通信量が概ね削減できる．逆にエージェントサイズが十分大きいときは，移動により総通信量はかえって増加する．
3. 総通信量がピークとなる困難な問題に対しては，エージェントサイズがある程度大きくても総通信量を抑制できる．

今後の課題は以下のとおりである．

- 分散 breakout アルゴリズム以外の分散アルゴリズムの場合について検討する．
- 3.1 節で示したような通信量以外の移動方法について検討する．
- 1 つのホストに多数のエージェントが集中すると，CPU タイムが各エージェントに分配されて 1 エージェント当りの計算スピードが減少する問題があるので，それを解決するモデルを考えたい．つまり，上記の 2 つの検討事項を踏まえた上で，複合的要因により移動を決定する方法の検討を行う．
- 実際のモバイルエージェント系に実装を行い，シミュレーションとの比較を行う．

謝辞　本研究の一部は，文部科学省科学研究費（課題番号 12780246）の補助によって行われた．

参考文献

[1] 本位田真一, 飯島正, 大須賀昭彦: エージェント技術, 共立出版 (1999).
[2] 長尾確編著: エージェントテクノロジー最前線, 共立出版 (2000).
[3] 新谷虎松, 大囿忠親, 福田直樹: モバイルエージェントの応用–マルチエージェントシステムのためのモビリティの利用–, 人工知能学会誌, Vol. 16, No. 4, pp. 488–493 (2001).
[4] White, J. E.: Mobile Agents, *Software Agents* (Bradshaw, J. M.(ed.)), AAAI Press/The MIT Press, chapter 19, pp. 437–472 (1997).
[5] Lange, D. B. and Ohshima, M.: *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley (1998).

[6] Ohsuga, A., Nagai, Y. and Irie, Y.: Plangent: An Approach to Making Mobile Agents Intelligent, *IEEE Internet Computing*, Vol. 1, No. 4, pp. 50–57 (1997).

[7] 横尾真, 平山勝敏: CSP の新しい展開：分散/動的/不完全 CSP, 人工知能学会誌, Vol. 12, No. 3, pp. 33–41 (1997).

[8] Mackworth, A. K.: Constraint Satisfaction, *Encyclopedia of Artificial Intelligence* (Shapiro, S. C.(ed.)), Wiley-Interscience Publication, New York, pp. 285–293 (1992).

[9] 平山勝敏, 横尾真: 分散不完全制約充足問題, 人工知能学会誌, Vol. 14, No. 4, pp. 60–69 (1999).

[10] Yokoo, M. and Hirayama, K.: Algorithms for Distributed Constraint Satisfaction: A Review, *Autonomous Agents and Multi-Agent Systems*, Vol. 3, No. 2, pp. 198–212 (2000).

[11] Conry, S. E., Kuwabara, K., Lesser, V. R. and Meyer, R. A.: Multistage Negotiation for Distributed Constraint Satisfaction, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 6, pp. 1462–1477 (1991).

[12] Huhns, M. N. and Bridgeland, D. M.: Multiagent Truth Maintenance, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 6, pp. 1437–1445 (1991).

[13] Lesser, V. R. and Corkill, D. D.: The Distributed Vehicle Monitoring Testbed: A Tool for Investigating Distributed Problem Solving Networks, *AI Magazine*, Vol. 4, No. 3, pp. 15–33 (1983).

[14] Sycara, K. P., Roth, S., Sandeh, N. and Fox, M.: Distributed Constrained Heuristic Search, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 6, pp. 1446–1461 (1991).

[15] Morris, P.: The Breakout Method for Escaping from Local Minimum, *Proc. of the Eleventh National Conference on Artificial Intelligence*, pp. 40–45 (1993).

[16] 横尾真, 平山勝敏: 分散 breakout：反復改善型分散制約充足アルゴリズム, 情報処理学会論文誌, Vol. 39, No. 6, pp. 1889–1897 (1998).

[17] Cheeseman, P., Kanefsky, B. and Taylor, W.: Where the Really Hard Problems Are, *Proc. of 12th International Joint Conference on Artificial Intelligence*, pp. 331–337 (1991).

[18] Minton, S., Johnston, M. D., Philips, A. B. and Laird, P.: Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems, *Artificial Intelligence*, Vol. 58, pp. 161–205 (1992).

[19] Hogg, T., Huberman, B. A. and Williams, C. P.: Phase Transitions and the Search Problem, *Artificial Intelligence*, Vol. 81, pp. 1–15 (1996).