ソフトウェア工学 119－1
（1998．5．15）

マルチエージェント協調プロトコルのコンカレントモデル

加藤 尚玄　　上田 佳寛　　安藤 津芳

沖電気工業株式会社

〒261-7109 千葉市美浜区中瀬 2-6 W.B.G
E-mail: kato@wbg.telcom.oki.co.jp

あらまし　　本稿は複数の移動型エージェントが協調して動作するためのプロトコルをペトリネットでモデル化する方法について記述する。エージェントの定義とプロトコルの定義は静止型エージェントの既知の研究結果を参照し、複数の移動型エージェントがインターネット上を移動する際のモデルの特徴を本稿では考察した。更に移動型エージェントがインターネット上を移動する際のコスト見積り問題を提案しその問題をペトリネット問題であるTインバリアント問題の応用であることを説明する。

Concurrent Modeling for Multi Agent Cooperation Protocol

Naomoto KATO　　Yoshihiro UEDA　　Tsuyoshi ANDO

Oki Electric Industry Co., Ltd.

W.B.G. Bldg., 2-6 Nakase, Mihama-ku, Chiba-city 261-71, Japan
E-mail: kato@wbg.telcom.oki.co.jp

Abstract　This paper describes the methods for modeling of multi agent cooperation protocol by means of Petri nets. This paper refers a static agents cooperation protocol and proposes a cooperation protocol for multi moving agents on Internets. This paper discusses a cost problem of multi moving agents on Internets and propose this problem can be solved if we utilize T-invariants Petri nets problem.

1 まえがき

近年、マルチメディアや分散処理等の通信技術を利用した組織における協同作業支援環境（グループウェア）の研究開発が盛んになっている。この背景にはコンピュータハードウェアの高性能化、低価格化及びネットワーク技術の進歩により、多くのコンピュータをネットワーク接続した分散環境が出現したことがある。組織においての分散環境の代表がイントラネットであり、一般での分散環境の代表がインターネットである。このような分散環境は、ニーズに逢った信頼性の高いサービスを効率よく低コストで提供できるだけの滞在能力を持つ。ところが、実際に このような能力を十分に活かす応用システムを開発することは、主に複雑さに起因する事情により非常に難しい。同様にシステムの運用・管理や利用の局面でも困難を伴うことが多い。

ここで浮かび上がってきた概念が、システムを細かく分割してそれぞれを独立したエージェントとして考えて、それらの関係を考えることによってシステム全体の秩序を示すという自律分散システムの考えである。この「エージェント技術」に基づく自律分散システムにおいては提供されるサービスは自律性を持ったエージェントと呼ぶ計算主体を複数組み合わせることにより実現される。エージェントは分散環境の中に規定された論理的な「場：フィールド」をにおいて互いに協調しながら処理を進めることによりユーザが欲するサービスを提供する。エージェントはフィールドに動的に出入りできるため、ユーザニーズやシステム構成の変更にも柔軟に適応することができる。

本稿では、非同期並列に動作する大規模離散事象システムの数学的、図形的モデルとして広く用いられているペトリネットを用いて、自律分散システムを実現するエージェントをモデル化することについて議論する。モデル化にペトリネットを利用することでペトリネット分野で研究されてきた解析手法がエージェントを用いた自律分散の解析に利用できる（文献[1]）。ペトリネットモデルを用いることは現在のエージェント研究では殆んどその解析手法などが提示されていない自律分散システムでのマルチエージェントの動作プロトコルの記述内容の一貫性、無矛盾などの解析手法に寄与する。

2 エージェントの定義

本章ではエージェントの定義を行う。定義は実装言語として実現されている Telescript（商標を含む全権利 Genral Magic 社保有）、Java（商標を含む全権利 Sun Micro Systems 社保有）の実装アーキテクチャを想定して行う。エージェントはコンピュータ上で動作する計算主体である。コンピュータのアーキテクチャを図1に示し、エージェントの動作方法を示す。

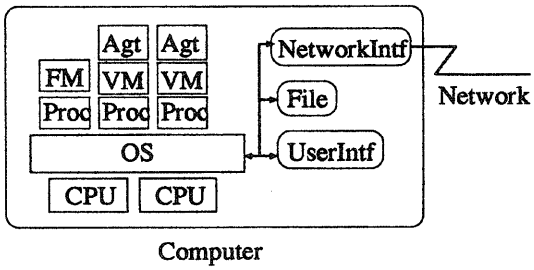

図 1: コンピュータアーキテクチャ

コンピュータはハードウェアとして複数の中央処理装置（図 1 での記号 CPU、以下 CPU）から構成される。コンピュータはソフトウェアとして1つのオペレーティングシステムオペレーティングシステム（図 1 での記号 OS、以下 OS）が CPU 上で動作する。OS は図 1 で Proc として記述されているプロセスを複数生成、消滅することができる。プロセスが全てのソフトウェアの動作主体である。OS はプロセスの動作を開始及び停止できる。また OS はプロセス間の通信を提供し、更にファイルシステム（図での記号は File）、ユーザインタフェース（記号は UserIntf）及びネットワークインタフェース（記号は NetworkIntf）の通信をプロセスに提供する。プロセス上で動作するソフトウェアはネットワークインタフェース通信を行うことにより他のコンピュータで動作するソフトウェアと通信を行うことがきる。このネットワークインタフェースを介して他のコンピュータに通信されるデータをメッセージと呼ぶ。

エージェントの動作は多様なハードウェアのマシンや多様な OS での実行が望まれることから、仮想マシンによって実現される。仮想マシンはソフトウェアでありプロセスで動作する。仮想マシンは図 1 で VM として記述されている。仮想マシンは仮想マシン上で動作するエージェント（図1での記号 Agt）に同一のソフトウェアインタフェースを提供する。言い替えれば仮想マシンは全てのコンピュータにおいて動作するエージェントのソフトウェア記述言語を統一するために、各コンピュータで異なるCPU、OS の相違を吸収するソフトウェアである。エージェントは仮想マシン上で動作するソフトウェアである。上述したようにエージェントのソフトウェア記述言語は仮想マシンを持つ全てのコンピュータで同一である。

エージェントはエージェント自身の実行中断命令または OS からのエージェント実行中断命令によりその実行を中断し他のコンピュータに移動することができる。エージェントは実行中断命令により実行を停止さ

れ、エージェントのプログラム、データ及び実行環境を仮想マシンにより圧縮、暗号化され他のコンピュータにネットワークインタフェースを介して送られる。実行環境には仮想マシンのプログラムカウンタ及びCPUレジスタ情報も含まれるためエージェントを受けとったコンピュータで実行を再開することができる。図1で記号Networkで示されているネットワークはTCP/IPプロトコルで接続されるインターネットとする。

図1で記号FMで示されているのはフィールドマネージャと呼ばれるソフトウェアである。フィールドマネージャはエージェントが動作する論理的な場：フィールドを複数提供し、複数のエージェントの協調の相手や手順などを選択する「協調プロトコル」を提供する。また移動型エージェントの移動受け入れ拒否・受諾や資源管理などの複数エージェントの動作コントロールを行う。またフィールドはコンピュータ上で一意な整数値IDで識別できる。

図2にフィールドマネージャとフィールドの関係を示す。図の記号Fieldで示されているのが論理的な場：フィールドである。フィールドはエージェントを複数格納することができる。

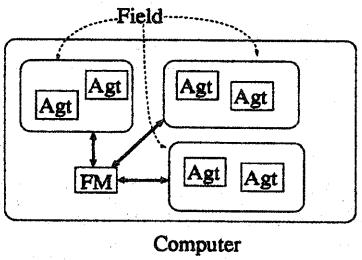

図2: フィールドとフィールドマネージャの関係

次に実装方式の観点からエージェントを以下の2種類に分類して考える。

1：移動型エージェント

移動型エージェントはネットワークを移動するエージェントである。移動型エージェントはネットワークを移動しながらユーザ要求サービスを達成する。

2：静止型エージェント

静止型エージェントは1つのコンピュータから移動はしないエージェントである。ユーザ要求サービスを達成するために他のコンピュータの情報が必要な場合にはメッセージを利用して、通信を行う。

2者の大きな違いはネットワークを移動するか否かである。それゆえネットワーク上の距離、すなわち通信遅延が存在しない場合には2者の区別は無意味である。逆に言えばバンド幅が狭く通信遅延の大きいインターネットにおいては2者の区別と用途の棲み分けは重要である。ネットワークを流れるデータとして移動型エージェントとメッセージを比較した場合、単なるデータであるメッセージとプログラムとデータと実行環境をデータ化した移動型エージェントのサイズはかなり大きなものになる。しかしユーザ要求サービスを達成するために多数のコンピュータと通信する必要がある場合には、移動型エージェントでは多数のコンピュータを巡回してゆく通信量であるが静止型エージェントでは多数のコンピュータと多数のメッセージ通信を行う必要があり、両者の通信コストはトレードオフになる。また不特定の外出先からのモバイルコンピュータからある仕事を行うエージェントをネットワークに送出し仕事の結果を自分のオフィスのコンピュータに報告するような用途で使用するエージェントは本質的に移動型エージェントとなる。

3 マルチエージェントの協調プロトコル

本章ではフィールドソフトウェアが提供すべきマルチエージェントの協調プロトコルについて説明する。協調プロトコルはエージェントが実際に他のエージェントや、エージェントが動作するフィールドと協調して目的を遂行するために必要となる。移動する複数のマルチ移動型エージェントの協調プロトコルについての研究は少ないようであり筆者には適切な参考文献を見つけることができなかった。そのため本稿では移動型エージェントではないがマルチの静止型エージェントの協調プロトコルのメッセージンングについて研究している文献[2]を取り上げ、ここで議論されているプロトコルについて考察する。静止型エージェントと移動型エージェントの違いは通信コストと情報の参照範囲の相違であることを考慮すれば協調の意味的構造は移動型エージェントの協調プロトコルと文献[2]の協調プロトコルではさほど相違ないと考えられる。そこで本稿で考察する自律分散システムエージェントの協調プロトコルとして文献[2]での協調プロトコルを採用する。また文献[2]での協調プロトコルは会議設定システムとして実装、実現されている点でも実績のあるプロトコルとして評価できる。文献[2]では以下の15個のメッセージ（タイプ）が事前に規定されている。request：要求、delegate：委任、result：結果、announce：告知、bid：入札、list：候補、choice：選択、cancel：取消、requests：分解要求、accept：受理、command：内部処理要求、comrslt：内部処理結果、error：エラー、fileout：ファイル出力メッセージ、filein：ファイル入力メッセージ。

更に文献[2]では以下の典型的な5つのプロトコル（文献では「典型的なメッセージパターン」と呼んでいる）が提示されており、フィールドとエージェントはこのプロトコルを認識しているとしている。なお

A ～ D は他のエージェントである。図 3 に文献 [2] で提示されている図を示す。もちろんこれらの 5 つの例は通信エラー回復手順などを含まない正常処理のみであるが、文献ではプロトコルはエージェント及びフィールドの記述言語で書き換え可能としている。

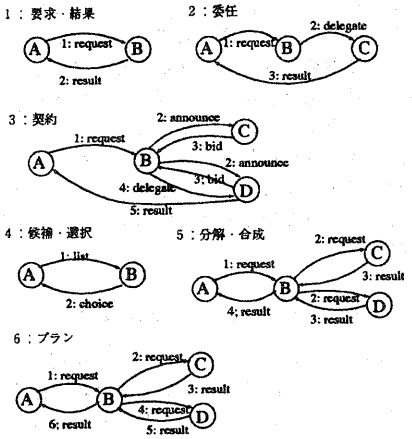

図 3: 協調プロトコル

図 3 に対して、文献 [2] で行われている説明を行なう。

[要求／結果]：要求に対し結果を返す基本的パターン。[委任]：要求を他に委任するパターン。[契約]：要求を委託する対象を要求元が即座に決められない時に利用するパターン。要求を受けたマネージャ B が各メンバエージェント（C、D）に対して告知を行なう。メンバは自己の基準に従い B に対し入札を行う。B は入札を評価し委任するメンバエージェントを決め委任する。[候補／選択]：候補のリストを送り選択結果を返すパターン。[分解／合成]：元の要求を分配し、その結果を合成し要求者に返す。[プラン]：元の要求に対して一連の要求／結果を繰り返し、最終的な結果を元の要求者に返す。

文献 [2] で図 3 に対して想定されている静止型エージェントではなく移動型エージェントを想定した場合の相違点について述べる。移動型エージェントを想定した場合には円で示されるエージェントはエージェントが動作可能なコンピュータに相当することに注意されたい。[要求／結果]：A の移動型エージェントが B のコンピュータに移動し要求を実現し A に再び移動する。[委任]：上述の要求／結果と同様に A、B、C、A と移動する。[契約]：この例では B において入札のアナウンスを行う際に移動型エージェントは自らのコピーを生成し C、D の二手に分かれて情報を探索する。その後 B に戻り依頼先を D に決定しコピーした移動型エージェントを消去する。その後、D で要求を実現し A に移動する。[候補／選択]：A から B に移動し依頼する候補リストを得て A に移動する。[分解／合成]：B で元の要求を分配する際に移動型エージェントは自らのコピーを生成し C、D に移動し要求の仕事を分配して行なう。その結果を B で合成し A に移動し要求者に返す。[プラン]：元の要求に対して一連の要求／結果を繰り返すので 1 つの移動型エージェントが A、B、C、B、D、B、A と移動して最終的な結果を元の要求者 A に返す。

4 ペトリネットによるモデル化

本章では移動型エージェントモデルのモデル化をペトリネットにより行う。ペトリネット理論の基本的定義に関しては参考書 [3] 等を参照して頂きたい。本稿では紙面枚数の都合上ペトリネット理論の基本的定義に関しては省略する。

移動型エージェントモデルからペトリネットへの変換規則について図 4 及び図 5に示す。

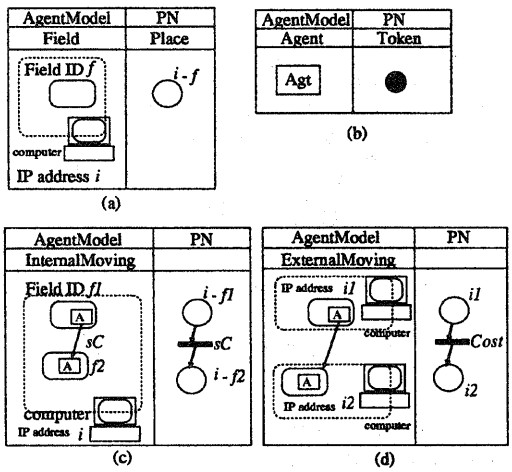

図 4: ペトリネットによるモデル化 1

図 4 及び図 5の説明を以下に行う。図 4 の (a) はあるコンピュータのフィールドをペトリネットのプレースでモデル化することを示す。コンピュータはインターネット全体で一意な IP アドレスという 4 バイトの数値が割り当てられる。またフィールドはコンピュータで一意な整数 ID が割り当てられる。この 2 つの ID をプレースのラベルとする。

図 4 の (b) は移動型エージェントをペトリネットのトークンでモデル化することを示す。現実にはフィールドが格納可能なエージェントの個数には限界があ

る。その意味でプレースにおけるトークンの数には制限があるが、本稿では解析の簡単のため無制限と仮定する。

図 4 の (c) はコンピュータ内のフィールドをエージェントが移動する場合、これをペトリネットのトランジションとアークでモデル化することを示す。*sC* という記号は移動のコストである。一般に内部通信のコスト：*sC* は非常に小さい。

図 4 の (d) はあるコンピュータから別のコンピュータにエージェントがインターネットを通して移動する場合、これをペトリネットのトランジションとアークでモデル化することを示す。この場合、プレースのラベリングに関して IP アドレスを示すラベルは必須であるが、フィールドの ID を示すラベルは必須ではない。理由はエージェントを送信する際に相手コンピュータがエージェントをどのフィールドに割り当てることは指定しない場合が多いからである。*Cost* という記号は移動のコストである。一般にインターネット通信のコストは *sC* と比較して非常に大きい。

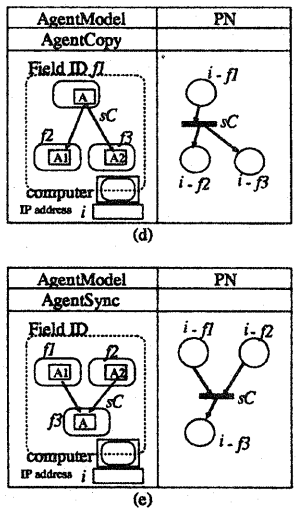

図 5: ペトリネットによるモデル化 2

図 5 のモデル化は通信のためではなく、協調プロトコルをモデル化するための変換規則である。

図 5 の (d) はコンピュータ内でエージェントがコピーされる場合、これをペトリネットのトランジションとアークでモデル化することを示す。(d) ではエージェント 2 倍のコピーを示しているが、トランジションの出力枝を N 本にすることにより N 倍のコピーが可能である。また (d) でのフィールド：*i-f2, i-f3* が *i-f1* と同じでも構わない。*sC* という記号はコピーのコストである一般に内部コピーのコスト：*sC* は非常に小さい。

図 5 の (e) はコンピュータ内で複数エージェントが同期を行う場合、これをペトリネットのトランジションとアークでモデル化することを示す。(d) では 2 つのエージェントの同期を示しているが、トランジションの入力枝を N 本にすることにより N 個のエージェントの同期が可能である。また (e) でのフィールド：*i-f3* が *i-f2, i-f3* と同じでも構わない。*sC* という記号は同期のコストである一般に内部同期のコスト：*sC* は非常に小さい。

このモデル化でコンピュータ内のフィールドに相当するプレースが 1 つのコンピュータとして 1 つのプレースに畳み込まれた場合、すなわちインターネット上のコンピュータのみを見たモデル化の場合にはトランジションの入出力枝が常に 1 本であるという単純な構造になる。

5 ペトリネットによる協調プロトコルモデル

次に図 3 をペトリネットでモデル化した図を図 6、図 7 図 8 、で示す。

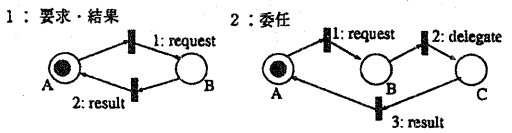

図 6: 協調プロトコルモデル 1

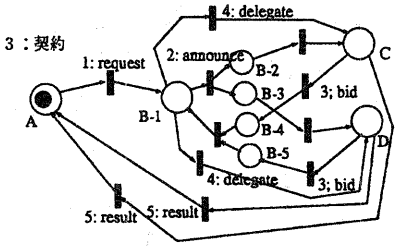

図 7: 協調プロトコルモデル 2

図 6 ではトークンとその発火規則がプロトコルメッセージ動作を（部分的に）規定する。注意しなくてはならないのは図 3 と説明文で示されたプロトコルを完全にペトリネットで表現している訳ではないということである。例えば図 7 で A から B にトークンが移動したとして意味的には 2 ： announce でコピーされ

てC、Dに移動しなくてはならない。しかしペトリネットモデルでは4：delegateでCまたはDに移動することも可能になっている。勿論、ペトリネットでは制御トークンを用いて2：announce、4：delegateの順序を制御することも可能である。

注意すべき点は協調プロトコルはフィールドマネージャで提供されるプロトコルと各移動型エージェントが持つプロトコルの解釈の両者が協力して遂行されるということである。ペトリネットの構造はエージェントモデルではフィールドマネージャで実装されるため、制御トークンなどの状態データを移動型エージェント毎にフィールドマネージャに実装することは現実的ではない。それよりも各移動型エージェントで自分自身が何を行なってきたかという状態データから判断して次に何をすべきかを判断するソフトウェアを移動型エージェントに実装するのが現実的である。

重要なことは移動型エージェントの協調プロトコルの設計においてフィールドマネージャで実装する部分と移動型エージェントの内部ソフトウェアで実装する部分を明確にするということである。逆にいえばペトリネットのモデル化能力を超えている部分はソフトウェア言語を用いて移動型エージェント内部で実装すべきとの見通しを得る。フィールドマネージャで扱う部分は大量のエージェントを扱うことになるためフィールドマネージャで複雑な処理を行なうことは現実的ではない。

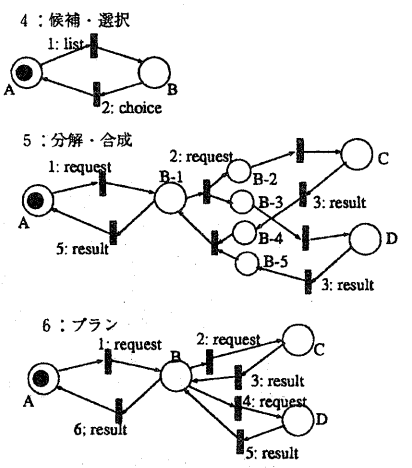

図 8: 協調プロトコルモデル3

6 移動コスト問題

本章ではインターネット中を移動する移動型エージェントのコストを求める問題について考察する。コストは通信遅延があるがゆえの考慮すべき通信時間である。この問題を移動コスト問題と呼び以下にその定義を示す。

Instance: 過去 n 回に渡り複数の移動型エージェントをある目的Aを持たせてインターネット上に出発させ、目的Aを達成し戻ってきた実績があり、それぞれの移動型エージェントは自分が移動したインターネット上のルートとコストのデータを持つ。

Question: 次に n+1 回めとして目的Aを持たせて移動型エージェントを送り出す。この時、過去のデータから予測される最も小さいコスト（最も早く戻ってくる）及び最も大きいコスト（最も遅く戻ってくる）を求めよ。◇

問題を考察するにあたりインターネットでの特徴を2点述べる。1点目は移動経路は動的に決まりコストは一意でないということである。移動コスト問題において移動型エージェントがコンピュータAからコンピュータBに移動したとしよう。この場合の移動に使用されるのはTCP/IPプロトコルでありAからBの間にはエージェントを動作させないような、ペトリネット上には陽に現れない、コンピュータが存在する。そのようなコンピュータは通信経路を通信回線の混み具合により動的に経路選択するコンピュータも存在する。このようなコンピュータをルータと呼ぶ。動的に経路選択するプロトコルにはRIP（Xerox社）、OSPF（RFC1311）がある。ある1つの経路に関してのコストも通信回線の混み具合により常に一定とは限らないが、ある中央値からの偏差をもったコストとして観測される。

2点目はある時点のインターネットの構造の全体は知ることができないということである。インターネットは自律的なシステムでありシステム全体の管理はなされていない。システム全体のネットワーク構造を知るためにエージェントやメッセージを用いて調べたとしても遅延がある場所の情報は過去の情報であり観測者は全ての情報を同時に知ることはできない。このインターネットの構造的な特徴は本稿のモデル化で使用するペトリネットの一般的な構造の特徴としてはあまり見られない。ペトリネットによるモデル化ではそのペトリネットのネットワーク構造は一意に静的に与えられることが多い。しかし移動コスト問題で与えられるネットワーク構造は、ある時点の移動型エージェントの移動経路であり問題を考える上では動的に変化するデータである。

7 移動コスト問題のモデル化

移動コスト問題をペトリネットを用いてモデル化しその解法の指針について述べる。モデル化の方法は4章の方法に従う。但しコンピュータ内部の通信コス

トはインターネット上の通信コストと比較して非常に小さいためフィールドに相当するプレースは全て1つのコンピュータとして1つのプレースに畳み込んでまとめる。このようにすれば前述したようにトランジションの入出力枝が常に1本であるという単純な構造になる。

移動コスト問題をペトリネットでモデル化した場合の特徴は2点ある。1点目はペトリネットにおける初期マーキングと最終マーキングが同じになるということである。これは最初の状態では自分のコンピュータにエージェントがあり、それが様々な移動系列を経てインターネット上の様々なコンピュータを移動し、最後に再び自分のコンピュータに帰ってくることと、モデル化では移動型エージェントはトークンに相当することから分かる。

2点目は移動型エージェントが移動したインターネット上のルートとコストのデータからペトリネットを合成するという点である。以下ではその合成方法について説明する。N個の移動型エージェント移動データの合成は2個の移動データの合成ができれば可能である。また移動データは4章で説明したペトリネットモデルでモデル化可能である。ここで移動データAに相当するペトリネット:Aに対して移動データBに相当するペトリネット:Bを合成することを考えよう。

最初にプレースに対する処理である。プレースはそのラベルで完全に識別できる。ペトリネット:Bのプレースを1つずつペトリネット:Aに同じラベルプレースが存在するか調べる。存在すれば何もしない。存在しなければそのプレースをペトリネット:Aに加える。

次に2つのプレース間のアークとトランジションに関する処理である。前述したようにトランジションの入出力枝が常に1本であるから2つのプレース間を指定すれば2本のアークと1つのトランジションが一意に定まる。これをルートセットと呼ぶ。ペトリネット:Bのルートセットを1つ取りだしペトリネット:Aに同じルートセットが存在するか調べる。もし存在しなければペトリネット:Aに加える。存在した場合には両者の移動データからインターネットの経路を調べる。ペトリネットでは陽には現れないがインターネットの経路は動的に変化するため常に同一とは限らない。経路が違う場合にはペトリネット:Aに加える。経路が同じ場合には両者のトランジションのコストの平均をとりペトリネット:Aのほうのトランジションのコストとする。このルートセットに関する合成を図9に示す。

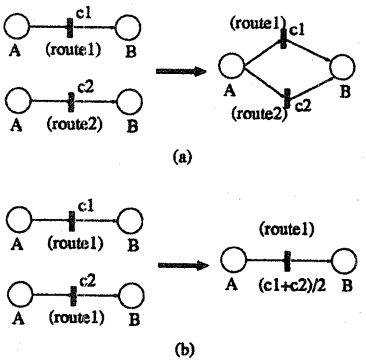

図 9: ルートセットの合成

7.1 Tインバリアントによる解法

移動コスト問題の解を与える方法はTインバリアント問題の解法と深い関連がある。以下に解法の方針を示す。移動コスト問題の過去 n 回に渡った移動型エージェント移動データは前章のペトリネット合成を用いることにより、1つのペトリネット構造になる。そのペトリネットの構造を、自己ループがないと仮定してP／T行列：Aで表されるとする。

移動コスト問題をペトリネットでモデル化した場合の特徴である初期マーキングと最終マーキングが同じであることから、式：

$$Ax = 0$$

を満たすような0でなくベクトルの各要素が非負整数であるxは、初期マーキングから最終マーキングに至るトランジションの発火系列の中に現れるトランジションの個数、すなわち発火回数ベクトルであることが知られている。このようなxをTインバリアントと呼びこれを求める問題をTインバリアント問題と呼ぶ。このようなxは複数存在し、問題を解くアルゴリズムが存在する。そのアルゴリズムが提示するxは互いに線形独立な最小基底ベクトル群を含む、互いに素な極小基底ベクトル群であることが知られている。注意すべき点は求まった解はxの発火を保証するものではない点である。しかしながら移動コスト問題ではあくまでコストの予測であるのでxの必要十分性については厳密に追求しない。xを求めるアルゴリズムは既知の研究も多く最近の報告では文献 [4] を参照されたい。アルゴリズムについては本稿では言及しない。

本稿ではそのようなxが複数求まった仮定する。移動コスト問題では各トランジションのコストは既知であり、このベクトルをcとすれば

$$x_i \bullet c = C_i \quad (\bullet \text{は内積})$$

で示されるスカラーC_iが、ある移動系列iで必要なコストC_iである。アルゴリズムで求まった全てのxを検査することにより、最も小さいコストと最も大きいコストを求めることができる。注意すべき点はある

コストの移動系列を実現するにはエージェントの移動を制御する必要があることと、その移動系列が実現可能である保証はないということである。

7.2 移動コスト問題の例

本章では移動コスト問題をペトリネットで解く例を示す。まず仮定として移動型エージェントを2回ほど送出したデータがあるとしよう。また簡単のためエージェントのコピーと同期を示すトランジション分岐と収束を通信経路上に配置している。図 10 の場合1と2に関するデータ：すなわちインターネット構造と、コストがペトリネットとトランジションに付与された整数としてモデル化されている。2つのデータを見ると自分 MyHost と他のホスト A が共有されていることが分かる。2つのデータを合成すると3：のように

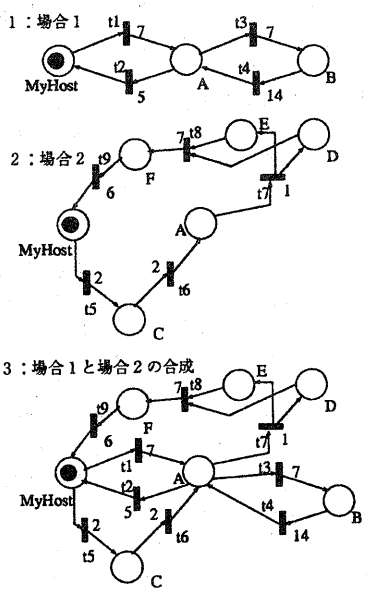

図 10: 移動コスト問題の例

なる。問題を解くためには式： $Ax = 0$ の A が必要である。この例での A を示そう。A は 行をプレース、列をトランジションに対応させ、プレースからトランジションへのアークがある場合には -1、逆は 1 を成分とした行列である。A は以下のようになる。

$$\begin{pmatrix} -1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 1 \\ 1 & -1 & -1 & 1 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}$$

次にコストベクトル c を以下に示す。

$$c = [7 \;\; 5 \;\; 7 \;\; 14 \;\; 2 \;\; 2 \;\; 1 \;\; 7 \;\; 6]^t$$

式 $Ax = 0$ を満たすある1つの x を x_i として例えば

$$x_i = [1 \;\; 1 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0 \;\; 0]^t$$

とすれば、コストは $x_i \bullet c = C_i$ で示されるスカラー C_i であり、計算すれば 1 2 となる。このようにして複数の x_i を求め、各コストを計算すれば最も戻り時間の早い場合の予想値、最も戻り時間の遅い場合の予想値を求めることができる。

8 結論

本稿ではエージェントとプロトコルについて静止型エージェントの既知の研究結果を参照し、目的である複数の移動エージェントがインターネット上を移動する際のエージェントの特徴を明らかにし定義を行なったまたエージェントをペトリネットでモデル化する方法を提案し移動エージェントの協調プロトコルのペトリネットモデルを示した。更に移動エージェントがインターネット上を移動する際のコスト見積り問題を提案しその問題をペトリネット問題である T インバリアント問題の応用として解ける指針を示した。

参考文献

[1] 野畑 智樹 他, “自律分散システムのエージェントネットモデルについて”, Technical Report of IEICE. CST95-32(1996-01), pp.23-29.

[2] 宮崎 一哉, “協調構造の動的な変更を可能とするマルチエージェント協調機構の開発”, 情報処理研究報告, 96-DPS-76, 1996.

[3] J. L. ピータースン 著, 市川 惇信, 小林 重信 訳, “ペトリネット入門”, 共立出版, 1984.

[4] 上田 佳寛, “インバリアントおよびサイフォンの効率的な検出アルゴリズムについて”, Technical Report of IEICE. CST97-31(1997-11), pp.1-6.