人　工　知　能 95－1
（1994．7．21）

分散プランニングのためのモデル獲得ルールの学習について

菅原 俊治
NTT 基礎研究所
〒 243-01 厚木市森の里若宮 3–1
sugawara@ntt-20.ntt.jp

概要: 分散協調問題解決では、各エージェントがそれぞれの観点から自己も含めたエージェントネットワーク全体のために行為を選択し実行するために分散プランニングを行う。しかしエージェントは客観的な情報を持たないため、各エージェントから観測できる情報と、その時点までに他のエージェントからえた情報からプランニングを行わなくてはならない。本研究では、プランニングに必要な局面のモデルを獲得できるようなルールを生成する学習方式について述べる。

Learning Rules to Create Non-Local Models for Distributed Planning

Toshiharu Sugawara
NTT Basic Research Laboratories
3–1 Morinosato, Wakamiya
Atsugi-shi, Kanawaga 243-01, Japan
sugawara@ntt-20.ntt.jp

Abstract: In cooperative distributed problem solving, an agent has to create a plan based on its local viewpoint in a distributed manner, in order to achieve coordinated actions appropriate to global agent coherence. From lack of non-local information, the created plan often contains unnecessary tasks or does not contain important tasks thus results in inefficient coordinated inferences and inappropriate resource usages. This paper presents a learning method to cooperatively identify situation-specific rules that enable a system to have the important part of the non-local model involved in the situation and the agents' states for creating an appropriate plan.

1 はじめに

分散配置された知的で自律したエージェントがある問題を協力して解決する分散協調問題解決 (Cooperative Distributed Problem Solving, CDPS) においては、如何に協調を実現するかが重要な課題である。全体として効果的な行為を実現するには、それぞれのエージェントが他のエージェントから必要な情報を獲得し、プランニングを行わなくてはならない。CDPS で一般的なプランニングは、マルチエージェントプランニングと分散プランニングがある [4]。マルチエージェントプランニングでは一つのプランナが複数のエージェントのためのプランを生成し、一方分散プランニングでは、分散した複数のプランナ (これらは普通アクションを行うエージェントでもある) が個々の観点から最適と思われるプランを生成する。CDPS では、分散プランニングが適当であると言われている。実際、ある特定のプランナを持つ場合は、それが全体のプランを生成しなくてはならないため、プランのための計算の複雑さが増大しプランナに負担がかかってしまう。ここでは分散プランニングに焦点をあてる。

個々の観点からプランを生成するという分散プランニングの特徴は、同時に適切なプラン生成の難しさをも意味する。つまりすべての情報を収集し、エージェント全体の観点から見たプランの生成ができないので、プランナは判断に必要な情報を十分持っていなかったり、通信の遅延や他のエージェントの変化を把握できずにもはや古くなったデータを参照することとなるかもしれない。一方、プランナがエージェント全体の情報を持っていたとしても、各判断のポイントで参照すべきものはその一部分であることが多い。

各エージェントが判断に必要な非局所的な情報が何んであるかを認識し、それらを獲得したうえで判断をくだすことが重要である。各場面でのドメインデータ、エージェントの状態、環境など全体のモデルのうち、真に重要な部分モデルを作らなくてはならない。しかし、何が重要であるかは、エージェントが設置された環境、他のエージェントとの関係なども加味して判断しなくてはならない。たとえば推論の中間結果が納められている変数の値があるプランの生成に重要であるとしても、特異的な環境を除いてはその変数の値が常に一定であるかも知れない。すべての場合をあらかじめ予測してシステムに記述することも考えられるが、システム作成のコストが高くなるばかりか、不要かも知れない値までも (通信により) 獲得し検証するので効率が低下する。また通信ネットワークのように多くの変形と時とともに進化する領域への応用では、すべての場合を予測することすら非現実的である。

本発表では、エージェントが、その置かれた環境において適切なプランを生成する上で、必要な情報を獲得し、エージェントネットワーク全体のモデルのうち重要な部分のみを認識するためのルールを学ぶ枠組みについて述べる。本学習では、学習以前または学習の途中段階では、協調活動に失敗するか、少なくとも適切でない行為が紛れ込むことを想定している。このような問題の発生をトリガとして、推論が終了したのちに、それぞれのエージェントが問題発生前後の状況を再生し持ち寄る。集められた情報をもとに、プラン生成に欠けていた情報を見い出し、同様な状況でその情報を早期に取得でき、かつ適切なエージェントに伝えるようにルールを生成する。必要であれば情報を早期に獲得するためにタスクの順序などに制限を加える。生成されたルールを実行するエージェントに提示して受け入れられるかを問い、可能ならそれをルールとして保持させ、不可能なら再度別のルールを抽出する。

本稿の構成は以下の通りである。第二節では、分散プランニングの処理過程のモデルについて述べ、どの部分で協調上の問題が発生する可能性があり、どのような問題を引き起こすかについて述べる。第三節では、提案する学習のモデルと、学習のために仮定するシステムの機能 (トレースの保存と問題を検知するモニタの存在) について説明する。つぎに、学習の詳細なプロセスを述べ、最後に簡単な例題 (実際の問題をモデル化したもの) をとりあげて、学習によってルールが生成される仕組みを説明する。

2 CDPS におけるプランニングのモデル

2.1 ゴール、タスク、オペレーション

エージェントは獲得された情報から、当面の解決すべき問題のゴール (目標) を設定する (もし、ゴールを作るための情報が十分なければ、情報を補うというゴールを生成する)。各プランナは、

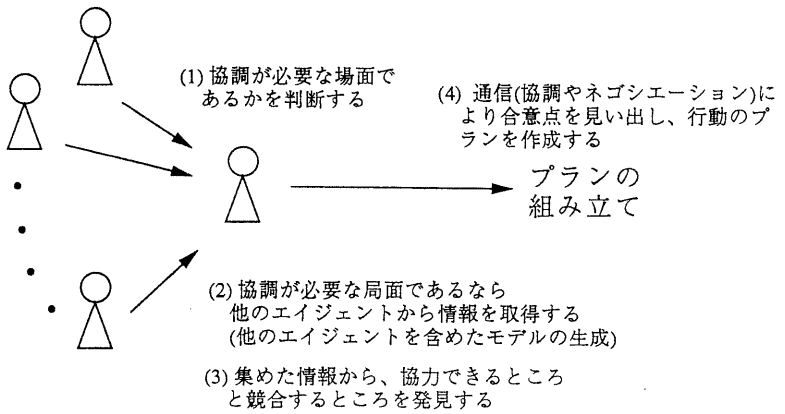

図 1: 分散プランニング処理過程のモデル

そのゴールの達成のためのタスクを生成する。タスクとは、適当に半順序が与えられたより細分化されたタスク (部分タスク) の集合で、これらのタスクを順序にしたがって実行することでゴールが達成される。これ以上分割できないタスクをオペレーションと呼ぶ。各部分タスクにもゴールが設定されており、これらをサブゴールと呼ぶ。あるゴールやタスクをエージェントが達成しようと持ち続けている時間を duration という。あるゴールを達成するためのタスクは、通常一通りではないから、プランナは協調しながらその場面で最適と思われるものを選択する。この選択されたタスクの列をプランと呼ぶ。

タスクは、それ以前にどこかのエージェントで実行されたタスクの影響をうけることがある。これらをタスク間の関係という。たとえば、Enable 関係はタスク T_1 の実行がタスク T_2 の実行を可能にし、Facilitate 関係はタスク T_1 の実行が T_2 の duration を短くする。Subtask 関係は、あるタスクがその上位のタスクの一部分になっていることを意味する。Support 関係は、T_1 の結果が T_2 のゴールのプライオリティを上げ、間接的に T_2 の優先度をあげる。この他 disable、hinder などの負の関係もある。CDPS では、タスク間の関係は複数のエージェントにまたがることもある [1, 2]。

2.2 情報の獲得から実行まで

本稿では、分散プランニングは、以下のように実行されると考える (図 1参照)。第一に、協調すべき場面であるかを判断する。この判断のために追加情報が必要であればそれも収集する。協調が必要であると判断したときには、周囲のエージェントから参照すべき情報を収集する。通常、各場面でどのドメインの情報が必要かは、知識として記述されている。その他に他のエージェントのプランやゴールなども必要に応じて収集する。収集された情報から、自己のプランの妥当性、他のエージェントのプランとの関係などを解析して、全体のタスクのモデルを作る。最後に、冗長な行為、追加すべき協調行為、実行のタイミングの設定などを考慮して、自己のみならずエージェント全体にとっても適切なタスクを選びだしてプランを組み立てる (または修正する)。この際に、他のエージェントへのプランの提案や、競合を防ぐためのネゴシエーションも行われる。たとえば disable 関係があればその影響を受けないか与えないように別のタスクを選択したり (または選択を依頼する)、enable 関係では続く他のタスクの実行を遅らせないように早目に実行することなどが考慮される。

2.3 問題の発生原因

前節のモデルからそれぞれのステップで、適切な協調が実行されなかった原因が考えられる。

(1) 協調が必要でない場面で協調を行おうとする。逆に協調が必要なのに省いてしまう。

(2) 適切な情報が収集されていない。不要な情報に惑わされる。原因としては、収集すべき情

報を獲得するための知識が無い場合や、時間制限で情報を要求前あるいは到達前にプランの生成をしなければならないことがある。

(3) 情報は取得しているが、(枝刈りや時間の制限などのために) それらを十分解析できず、重要なプラン間の関係が考慮されていない。

(4) 十分な情報と解析はなされたが、そこから適切なタスクを選択できていない。これはタスクを選択するルールがないか、あってもルールに不適切なプライオリティが設定されていたなどが考えられる。

本発表では、(2)(3) について論じる。なお (4) については [7] で触れられている ((1) はすべての場合に関連してくる。ここでは特に協調が必要なのに不十分であった場面の認識が関係する)。

(2)(3) の問題は、プランニングのために生成される環境、問題の状況、他のエージェントの能力や推論状態などに関するモデルを不完全とする。一般に各エージェントはすべての情報を獲得できないので、各々の観点と立場から主観的なモデルに基づいてプランを作らざるをえない。不完全なまたは片寄った主観的なモデルからは適切な協調は生まれない。一方、すべての情報を含む理想的な客観的立場からのモデルに基づいたとしても、必要とするのはその一部であることが多いから、各局面で必要な情報のみを含むモデルを各エージェントが生成できればよい。しかし、何が必要な情報であるかは問題の種類、他のエージェントの種類、環境などにも依存する。したがって、あらかじめ何が必要であるかをすべて羅列できないし、出来たとしてもある環境ではまったく必要ない事項まで参照することになるかも知れない。

不完全なモデルは多くの協調上の問題点を発生させる。たとえば、

Long Duration: プランの duration を長くさせる。たとえば、他のエージェントが適切なモデルを保持せず、こちらで真に必要なデータやタスクの要求が後回しにされてしまう (実行順序の問題)。

Redundancy: 同じタスクを複数のエージェントが実行する。これは不要なタスクであるだけでなく、必要以上にリソースを消費することにもなる。

Executions of Unnecessary Tasks: 実際には問題解決に貢献しないタスクを実行する。これは自己の推論を遅らせるばかりではなく、通信をともなえば他のエージェントの推論も惑わせる結果となる。

本稿では、各場面で必要な情報を学習する枠組みについて提案する。なお、問題 (2)(3) のみを対象しているので、必要な情報を含むモデルが生成されれば、適切なタスクを選択するためのルールはシステムが保持しているものと仮定する。

3 モデル獲得のための学習

プランニングに必要なモデルを生成するための情報は、他のエージェントが持っているものの、それを要求していないか送られていないために不完全さが発生するものとする。

3.1 推論トレースとその役割

各エージェントが推論中に履歴を残すことを仮定する。ここで履歴とは、それぞれの判断のポイントで選択したゴール / プラン / タスク、それらを選択することになった理由、通信の時刻 / 内容 / 送り先、そのときに獲得していた情報 (ドメインと相手のプランやゴールに関するものなど) とそのうち実際に参照していたものなどの情報を含む推論トレースである。各エージェントは、推論トレースをもとに各場面の状態を再現できなくてはならない。

推論トレースは以下のように構成される (図2を参照)。各エージェントは、それぞれの判断ポイントで何かを選択するが、その根拠となったデータをモニタに送る。この中には、ドメインのデータ、自他のエージェントのゴールやプランの内容、選択されなかったものの候補となっているたのプランやタスクなどが含まれる。図2は、CDPS システムの典型的なシステム構成に推論モニタを付加したものである (たとえば [1]) [1]。ここ

[1]Coordinator は、他のエージェントがら送られてきたタスクやデータの内容を解析し、要求に対応するタスクやタスクの列をスケジューラに渡したり、必要であれば、(Local) Planner に知らせて競合や共通項目を割り出す。逆にタスクやデータの要求も行う。Planner は、エージェントが保持する情報の範囲内で最適と思われるプランを生成する。したがって外部から新たな情報、とくに他のエージェントのプラ

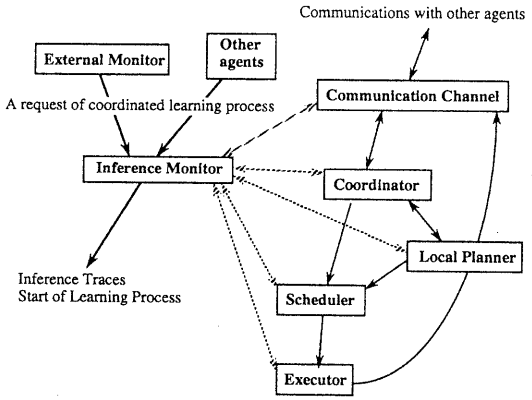

図 2: エージェント内のモジュールと推論モニタ

ではそれぞれのコンポーネントがその活動に応じてモニタにイベントとそれに関係する内容を通知するか、モニタが割り込みをかけて必要な情報を吸い上げるかをして、それらを時系列に並べて記録する。

推論トレースの役割は以下の通りである。第一に、モニタから次節で述べるような警告があったときに、トレースを使って協調上の問題が起こった局面を同定する。問題が観測された時刻、その前後のイベント (主に通信や特殊なリソースへのアクセスなど) から、それを発生させたタスクを見つける。第二に、トレースにより問題発生の直前の段階を再現する。これによりエージェントがプランやタスクを選択したときの内部状態と、その局面で各エージェントが持つ情報が何かを知ることができる。この情報を交換すれば、より大局的なモデルを作ることができる。

3.2 問題の検知

推論モニタは協調上の問題が発生した (もしくはその可能性) ことを監視できるものとする。これらは、たとえば以下のようなイベントを全部もしくは一部検出する。

(i) 解を求められないか、解が間違っていた。

(ii) 要求された時間内に解が求まらなかった

(iii) あるプランの duration が予期されたよりも十分長かった

(iv) 応答の遅延等による長い idle 時間があった

(v) リソースの寡占状態や同時利用要求があった

(vi) 新しいプランやオペレーションが選択された

これらの監視は、予期される実行時間や応答時間に関するデータ、リソースの状況などをもとに、コントローラーやメタコントローラにより検出できる。なお、(vi) は、新しいプランやオペレーションが新たな問題を引き出していないかを検証するためにある。この他に図 2のように外部環境のモニタを仮定して、自己の推論だけでなく、環境に与えた影響のモニタをしてもよい[2]。

学習のプロセスは、推論モニタからイベント発生の知らせがあったとき、または他のエージェントから学習プロセス起動の要求があったときにスタートする。協調上の問題が、イベントとして顕著に現れないこともあるので、定期的な起動も仮定する。

(i) についてもう少し議論を加える。(i) の場合には何が正しい解なのかを指定する必要がある。またこの場合の問題は協調コントロールとは関係なく発生した可能性も高い (たとえば推論ルールが欠如していた[3])。ここではルールの実行順序などの変更はできるが、新たなドメインに関するルールは生成できない。したがって、本研究では、本来必要なルールはあるもののコントロールの結果、枝刈りなどにより重要なルールが削除されてしまったとする。外部から指定する情報は、タスクが刈られてしまった場面で本来重要であるはずのタスクである。学習によって同様な局面でそのタスクを選択できるようなルールを追加する。この場合は最終的な結論を導くまでたどってはいないので、一回では不十分かも知れない (次節参照)。

ンに関する情報が届いた場合には、必要に応じて協調行為を含むプラン (もちろん他のエージェントと合意して) に仕上げる。Scheduler は、選択されたプランに基づいてタスクの割り当て、実行順序を決定し、そのスケジュールにしたがって Executor に渡す。Executor は、ルールを実際に起動させ、観測やテストとそれらの結果の解析を行う。テストの際に、通信を伴うこともある。

[2]たとえば、我々が開発したインターネット監視 / 診断システム [5] では、ネットワークを管理するコンポーネント自身が、自己が送出したパケットをも監視し、環境に与える影響を把握している。

[3]本学習で獲得するルールはコントロールのルールである

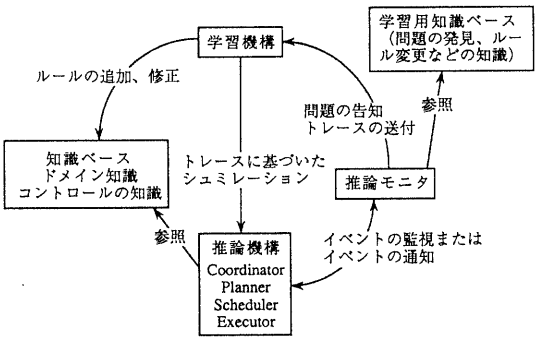

図 3: 学習を含めたエージェントの動き

3.3 学習のモデル

本学習は協調上の問題発生をトリガに開始され、将来の同じ状況で、問題を起こしたタスクの選択を防ぐため、コントロール上必要な非局所的なモデルを獲得できるように情報収集または情報送付のためのルールを作る。情報が単に不足しているだけの場合には、以下で述べる学習方法で必要なルールが獲得できる。しかし、学習結果をもとに推論した結果がまた別の問題を引き起こす場合には、学習プロセスが繰り返し適用され、徐々に必要なルールを獲得する。つまり繰り返しにより、徐々にさらなる情報を集めるようにするか、プランやタスクの候補の中から別のものを適当に選択して、試行錯誤的に問題が起こらないものを見つけ出す。ただし、繰り返しているうちに、第2.3節の(4)のタイプの問題に遭遇するかもしれない。このときには、[7]で提案した学習を適用することになる。

本研究の学習のモデルで、問題発生に依存して学習が起動されること、学習を繰り返しながらルールを求めることの2点が、実際の分散環境で重要であることを述べておく。たとえば、我々はインターネットワーク監視システムを構築している[5]。ネットワークの環境は多種多様な計算機、プロトコル、サービスがあり、それぞれのネットワークが個性的でしかも絶え間なく変化を続ける。ある環境でのネットワーク上の問題が、他では問題とならなかったり、症状の出方も異なる。これは協調上の問題に関しても同様で、たとえば協調のための通信量増大が問題になるところとならないネットワークがある。特にエージェントが自己の通信活動を監視できる本システムでは、それ自身が環境に対する影響のモニタとなる。このように個性的で変化を考慮しなくてはならないシステムの研究は、分散処理を実際に適用するには必須である。これはネットワーク(環境)の変化、交信相手の変化、並びに予期しない相手からの応答などがあるからである。

エージェントがどのような環境で動作するのか予測できないか、環境が変化する可能性がある場合には、最適な協調のためのルールも変わる可能性がある。すべての場合を尽くすのは非現実的であり、仮にできたとしても、状況や環境を調べるためのプロセスが多くなり非効率的である。これは、ある環境では絶対に起こりえないような状況まで調べることになるからである。たとえばネットワークの例を取ると、診断によって通信のためのパケットが多く送出されても通常の環境では問題ない。しかし、特殊な環境として、非常にスループットの低いネットワークや従量課金のネットワークでは、しばしば望ましくない状況を産み出す。そのためのチェックは本当に必要な環境では仕方がないものの、そのようなネットワークが存在しないコミュニティでは意味がない。

本学習モデルでは、問題発生ごとに同じ問題を繰り返さないように(少なくとも問題を引き起こしたタスクは同様の局面では使わないように)しているため、実際に必要のない局面や環境ではルールを生成しない。変化が生じ、それが新たな問題を引き起こしたとしても、その問題をトリガとして再び学習プロセスが実行され新たなルールを作るか既存のルールを書き換える。上記の通信ネットワークの例のように、通常の環境では問題は発生しないのでルールも生成されないが、特殊な場合には推論モニタか環境モニタから警告が送られ、それを回避するためのルールが獲得されるのである。

3.4 学習の方法

推論モニタや環境モニタまたは他のエージェントから協調(通信上)の問題があったと報告をうけたとする。以下のような順序で学習が進められる

推論に貢献したタスク / 通信のマーキング

最終的な結論を導くのに貢献したエージェントの活動を解析するフェーズである。これは推論ト

レースを最終結果からタスク間の関係を逆にたどるマーキングのプロセスにより同定できる。

M1: 最終結果を直接導いたタスクにマークをつける。

M2: マークをつけたタスクを選択する根拠となったデータを導いたタスクにマークをつける。またこのデータが他のエージェントから送られてきていた場合にはその通信にマークをつける (送り手、受け手の両側で)。またその送られたデータが、予めデータを要求していたために発生したのであれば、そのデータ要求の通信にマークをつける。

M3: マークをつけたタスクに正の影響を与えたタスク (enable 関係や facilitate 関係) にマークをつける。

M4: マークのついた通信を発生させたタスクにマークをつける。

M5: マークのついたタスクで発生したタスクの要求またはデータの要求を行うための通信にマークをつける。

M6: マークのついた要求を伴う通信によって、その要求を実行するために発生したタスク、またはタスクにマークをつける。

M7: マークのついたタスクで実行された部分タスクにマークをつける。

このフェーズは通信をたどることで他のエージェントにも伝播する。通信についたマーク以外は各エージェント内で局所的に保持される。

注意として、facilitate 関係のように定量的に影響するものも考慮した。実際 facilitate 関係は関係相手のプランの実行を容易にするが絶対に必要なものではないが、その場面で推論効率上重要な役割を果たしている可能性があるので無視できない。負の関係 (disable など) も考慮すべきであるが、それは将来の課題とする。

問題発生局面の同定

第 3.2 節で見つけたイベントの種類に応じて問題の発生場所をトレース上で見つける。ここで三つの場合に分ける。第一の場合は、第 3.2 節 (ii) のように特定はできないが全体的に協調活動の向上が求められる場合である。第二は (iii) ～ (vi) のように、ある程度問題発生の場所が絞り込めている場合である。いずれの場合にも以下のような場面を、(ii) のときはトレース全体から、(iii) ～ (vi) のときはイベントの発生した近隣から見つけ出す。

- 結果が参照されていないタスク
- マーク付きのタスクで予期されたよりも実行時間が長かったもの。
- マークが付かないタスクで、マーク付きのタスクの実行を遅らせたもの
- マークが付かないタスクで、コストが大きいかユーザとのインタラクションがあるもの
- マークが付いた冗長なタスク (たとえば変数が 2 回定義されたか既知の内容が送られてきたなど)
- 同じリソースをほぼ同時にアクセスするタスク、または一つのタスクでもリソース寡占のイベントが報告された時刻付近でそのリソースを使用したタスク

なお、(iii) ～ (vi) でイベントの発生した場所は、イベントの時刻、前後の通信やリソースへのアクセス状況などふ含めて同定する。もう一つの場合は (i) のように外部からなんらかの指定をしないと問題発生場所が分からない場合がある。たとえば、解を求めることができなかったのなら、どこで適切なプランが切り捨てられたのかなどを指定する必要がある (第 3.2 節の議論を参照)。この場合には指定された正しいタスクまたはプランを選択できなかった局面を問題発生場所として、外部から指定する。

問題発生時の解析

問題が発生した場面で (あるいは発生したと思われる場所で)、その問題を起こしたエージェント (達) は、問題を引き起こしたタスク (プラン) を選択したときの (ドメイン、エージェントネットワーク、環境などの) モデルを作り上げる。その中で、実行されたタスクを選択する根拠となったデータをトレースと比較しながら見つける。また他のエージェントは同時期のモデルを生成する。他のエージェントが同時期と判断するのは、その前後の通信からの相対時間を利用する。

つぎに、生成したモデルをエージェント間で交換する。これによってその場面での全体的なデータが収集できたことになる。問題を起こしたエージェントは、その全体のモデルをもとに再度プランならびにタスクを選択する。ここで3つの場合に分ける。第一に、もし唯一同じタスク(つまり問題を起こす)ものだけしか選択できない状況になった場合には、その局面でのルールがそもそも不足しているので、本論文の範囲外となる。第二に、もし異なるプランのみが選択できる状況となったときには、それが同じ問題を引き起こさない(可能性がある)と考えられるので、以降の同様な局面ではそのプランの中から選択できるように次のフェーズに移る。第三に、新たなデータを供給しなくても、問題を起こしたプランの他にもプランが提案されておりそれらが同程度のプライオリティを持つものの、たまたまその一方が選ばれた場合である。この場合には、他方のプランを選択すれば問題を起こさない可能性がある。最後に、データを集めた結果、問題を引き起こしたプランも提案されているものの、他のプランも提案されている場合には、やはり他方のプランを選択するようにしたい。

ルールの生成フェーズ

このフェーズでは、適切なプラン生成や選択に必要だが前回の推論では獲得されていなかったか参照されていなかったものを獲得し参照するためのルールを作る。ここでは前フェーズで異なるプランを生成し選択するために参照したデータと問題を起こしたルールを排除するために貢献したデータがあれば、それらを獲得するアクションを生成し、ルールの実行部とする。また、問題を発生させたプランをも含んでいるが、他のプランも同様に選択されている場合には、他のプランを選択するように実行部に書き加える。ルールの条件部は、本データを獲得するべき局面を同定する部分でなければならない。現状では、そのルールが発火すべき局面で、エージェントが保持するデータがまったく同じときのみに条件部が満たされたこととする。ただし、照合するデータは、データ間の依然関係などから不要なものは除かれる。得られたルールは、そのルールを実行するエージェントに渡され、そのルールが実際に受理できるかどうかを判断し、可能であれば受理する。受理できなければ、再度可能な別のルールを生成する。

注意として、このようにして作られたルールはover-specificationとなり、同じではないが類似した局面で同様なルールが必要な場合には発火しない。この場合には、ルールの条件部を広げるためにcomparative analysis[3, 6, 7]を採用するが、詳細に付いては本論文の範囲を越えるので省略する。第二の注意として、学習のモデルの節でも議論したように、ルールは失敗と試行を繰り返しながら獲得されていく。そのために必要なデータ(トレース、学習解析結果など)は記憶されなくてななならない。たとえば上記のルール生成の部分で、「直前の推論で問題を起こしたプラン」を選択しないだけでなく、「過去に同様な場面で問題を起こしたプラン」も選択してはならない。この繰り返しの結果、すべてのプランが選択できなかった時には、そもそもプラン生成のためのルールが欠如していたわけであり、別の学習アルゴリズムが必要となる[6, 7]。

4 協調診断の例

本学習はインターネットワーク監視/診断システムで評価中である。紙面の関係で、ネットワークの例題は触れられないが、代わりに簡単な協調活動のモデルをもとに如何なるルールが獲得されるかを説明する(このモデルは実際のインターネットワークの診断例をモデル化し、一部改訂したものである[4])。なお本例題では、予期された以上に時間のかかったプランがあり、学習プロセスが起動されたとする。

第一のフェーズは、最終結果をもたらしたタスク(プラン)から逆にマーキングプロセスを適用する。図4の網かけのタスクとそれらを結ぶ通信にマークがつく。第二のフェーズでは、問題の

[4] より具体的には、スタティックルーティングを採用しているこちら側のホストに、あるネットワークへのルーティング情報が(デフォールトルートも)欠如していたとする。その相手側ネットワークからこちらの対象ホストへあるサービスの要求を出すが、応答がないと報告されたとする。こちら側の診断エージェントがAgent1、相手側のネットワークの診断エージェントがAgent2に対応する。Agent1では、下位レベル(ネットワークレイヤのlayer4以下)に関しては問題なく見えるので、サービスに関わる問題と判断する(T_2側)。一方、Agent2では、基本的なテストパケットに対してにもさえ応答しないことが観測される。この情報がAgent1に伝わり解析されると、agent1が作った対象ホストに関するモデルと矛盾することが見つけられ、その差異から新たなタスク(T_5)を生成し、結果をえる。

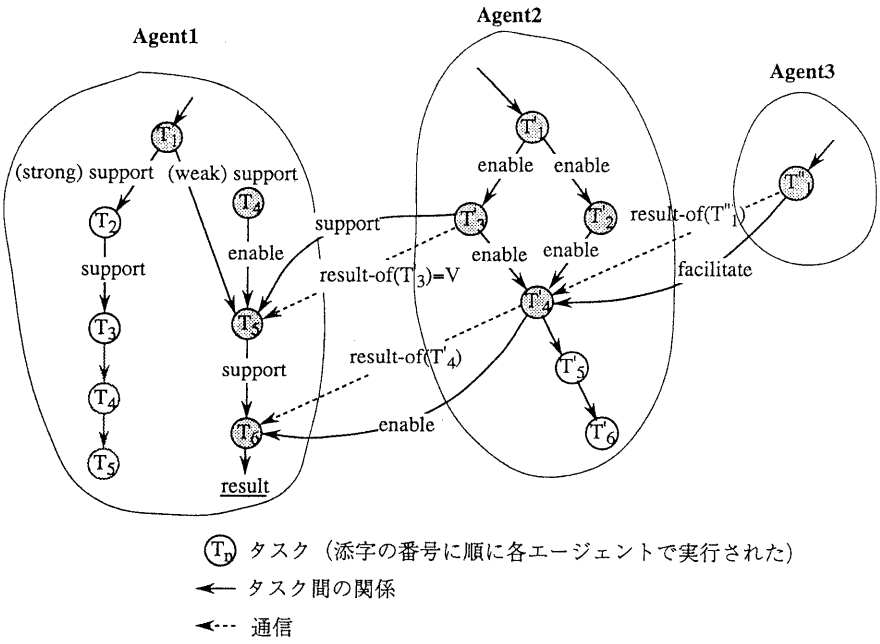

図 4: 協調活動の一例

発生局面を同定する。問題が発生したプランで図4のようなタスクが実行されていたとすると、Agent1 で T_5 ではなく T_2 が先に選択されたが、その後 T'_3 の結果が送られてマークのついたタスク T_5 が選択されたことが分かる（「マークの付いていないタスクがマーク付きのタスクの実行を遅らせた」という問題発生局面を認識するルールによって発見される)。なお注意として、Agent1 で T_2 は常に不要な分けではない。T_2 が先に選択された理由としては、たとえば、(1) T_1 の結果から可能性の高いゴールが提案されそれを検証するために T_2 が選択された、(2) T_1 の結果から影響度の大きい重大な問題の可能性が示唆され、それを検証するために T_2 が選択されたなどが考えられる。したがって、T_2 を選択するのは一般的には正しい。ただし、この局面ではなるべく早く T_5 にスイッチしたい。

次のフェーズでは、各エージェントが他のエージェントが持つデータをも獲得し、そのタスクを選択した局面でのより大域的なモデルを作る。このモデルが図4全体である。ここで、Agent2 のタスク T'_3 の結果が V という値であったことが Agent1 に伝えられてタスク T_5 が選択できたことが分かる。しかし、T'_3 は T'_2 の後に実行され(この二つは Agent2 の観点からはどちらが先でもよい)、そこでこの結果の転送が遅れたことが分かる。ここでは、

```
if (同じ局面であるとき)
  prior(T'3, T'2)
  if (Result-of(T'3) = V)                    (R1)
    then send-immediately(result-of(T'3),Agent1)
```

というルールが生成される。これは Agent2 において、同じような局面 (つまり同じ問題の種類と同じ観測結果であったら)T'_2 よりも T'_3 を先行させ、その値が V であったならその値を直ちに Agent1 に (相当する) 転送することを意味する。このルールは Agent2 に提案され、受理可能である可能かを問う。ここでは、T_2 と T_3 の順序が本当に交換可能であるかどうかを調べ、問題なければこれを受理する。この確認は必須である。なぜなら、ここで作られたモデルは、問題発生にかかわりありそうな部分を切り出してきた一部のモデルなので、このモデルには記述されていなくても、間接的にタスクの順序に影響を及ぼす要因があるかも知れないからである。また Agent2 がこのルールを持つことを拒否した場合には、たとえば、Agent1 が本ルールを持ち、同じ局面になっ

たら、上記のルールに相当する内容を要請として相手側に送ることも考えられる。この場合 prior の部分はタスクを要求するのではなく、もし共に Agent2 で提案されたら、そこに選択の順序をつけて欲しいことを意味する。

一回のルール生成で十分な場合もあるが、本例題では再び予期された以上に時間を消費する可能性がある。たとえば、T_5 が選択、実行されても T_6 が実行できずに、T'_4 の結果を待ってしまうかもしれない。この場合に、第二回目のルール生成時には、T'_4 の実行を速めるために Agent3 に facilitate 関係にある T''_1 に実行を要請することもできる。この場合にえられた新たなルールは、

if (同じ局面であるとき)
  prior(T'_3, T'_2)
  if (Result-of(T'_3) = V) (R2)
    then send-immediately(result-of(T'_3),Agent1)
    and Request(T''_1,Agent3)

となり、Agent2 に生成される。仮にこの後にも消費時間が長い場合には、協調の部分のみでは高速化は望めず、基本的に何らかのルールが欠如しているか、システム上の問題の存在を意味する。

ルールの生成以外に、新たな協調すべきタイミングが見つかったことも重要である。つまり、Agent1 において T_1 が実行され、この問題と同様な結果となり Agent2 にも同じタスクが実行されようとしているときには、Agent2 の情報に大きく影響されることが分かった。Agent1 は T_2 は実行するものの、Agent2 に学習したルールの適用を促したり (もちろん、Agent2 はまったく異なる種類の問題であると判断しているかもしれない。そのときには、このルールは適用されないし、それが正しい)、相手からの通信に迅速に応答することができる。

5 おわりに

本稿では、分散プランニングで各局面ごとに必要な非局所的な情報を失敗から同定し、将来の同様な状況で情報が獲得できるようにルールとして保持する枠組みを提案した。提案した学習モデルでは、エージェントははじめに効率的な協調ができないか失敗するかもしれない。しかし、この失敗の後に各エージェントがその状況を再現したモデルを作り、それらを持ち寄ってお互いの手の内をみせる。そこで問題が発生した原因の解析され、回避の方法 (少なくとも問題を起こしたタスク以外のものが選択できるように) がルールとして生成される。

参考文献

[1] K. S. Decker and V. R. Lesser, "Generalizing the Partial Global Planning Algorithm," *Int. Jour. of Intelligent Cooperative Information Systems*, Vol. 1, No. 2, pp. 319-346, 1992.

[2] Decker, K. and Lesser, V. R. "Quantitative Modeling of Complex Environments," International Journal of Intelligent Systems in Accounting, Finance and Management, special issue on Mathematical and Computational Models of Organizations: Models and Characteristics of Agent Behavior, 1993.

[3] E. Hudlická and V. R. Lesser, "Modeling and Diagnosing Problem-Solving System Behavior," *IEEE Trans. on System, Man, and Cybernetics*, Vol. 17, No. 3, pp. 407-419, 1987.

[4] F. von Martial, *Coordinating Plans of Autonomous Agents*, Lecture Notes in AI 610, Springer-Verlag, Berlin, 1992.

[5] T. Sugawara, "A Cooperative LAN Diagnostic and Observation Expert System," *Proc. of IEEE Phoenix Conf. on Comp. and Comm.*, pp. 667-674, 1990.

[6] T. Sugawara and V. Lesser, "On-Line Learning of Coordination Plans," *Proc. of the 12th Int. AAAI Workshop on Distributed AI*, 1993.

[7] 菅原 俊治, "協調のためのルールの学習について," 日本ソフトウエア科学会 マルチエージェントと協調計算研究会 1993 (同タイトルで マルチエージェントと協調計算 III, MACC'93, 近代科学社にも掲載予定).