人 工 知 能 89−19
（1993. 8. 19）

実時間応答を目的とした推論方法に関する研究

今成 文明† 小川 均† 高玉 圭樹‡

† 立命館大学理工学部情報工学科
‡ 京都大学大学院工学研究科応用システム科学専攻

Abstract

プロダクションシステムにおける問題解決で一般に用いられている推論方法では本質的に推論が最終的に終了するまで解を得ることができない．しかしながら処理に時間のかかる計画・スケジューリング問題などでは完全な解でなくても最終的な解の近似解を実時間で得たい場合がある．

そこで本論文では，このような問題に対処するため，実時間で近似的な解を導出する推論方法の提案を行なう．問題の制約という観点に注目し，まず必ず解が満足しなければならない制約（絶対的制約）とできるだけ満たした方が良い制約（希望的制約）に分ける．ます絶対的制約のみを考慮して実時間で近似的な解を獲得し，その後希望的制約を考慮し近似的な解を修正することで最終的な解を得る．解答の要求があったときに，その時点での解を提示すればよい．

本論文ではこの推論方法の推論メカニズムの提案を行ない，また関連研究の考察を示した．

An Inference Method for the Purpose of the Real-Time Response

Fumiaki Imanari† Hitoshi Ogawa† Keiki Takadama‡

† Department of Computer Science and Systems Engineering,
Faculity of Science and Engineering, Ritsumeikan University
‡ Division of Applied Systems Science,Graduate School of Engineering,
Kyoto University

Abstract

As for the planning or scheduling problem, there are case where approximate results are expected to be obtained within the real time.

Previous inference methods, however, cannot cope with these problems because results of the inference are basically obtained after terminating the inference comletely.

This paper proposes an inference method with which an approximate result is obtained within the real time by separating constraints into absolutely-satisfied and desirable-satisfied ones.

In this method, the inference is excuted by considering only absolutely-satisfied constraints, and the final result is obtained by modifying an approximate result by desirable-satisfied constraints.

1 はじめに

実用的問題解決では扱うデータの量が大きくなり，また問題に関連する制約が複雑なものとなってきており，解を得るまでに長い時間を必要とする．既存の推論方法は「推論が完全に終了するまでは解答を得ることができない」という本質的な性質があるからである．リートアルゴリズムなどによる推論の高速化 [1] や競合解消戦略による推論の効率化などにより解をより短時間で獲得しようとするアプローチも研究されているが，上述の本質的な性質は変わらないため根本的な解決にはなっていない．しかし，災害時の電話通信網，電力送信網の変更などのように緊急時に実時間での解を必要とする問題が存在する．この場合，最適解である必要はない．これらの問題では環境がどのような場合においても適用可能な明確な推論規則が整理されていない場合が多い．そこでこれらの問題を制約充足問題（constraint satisfaction problem: ＣＳＰ）として捉えると問題を適切に表現できる場合が多い．ＣＳＰには二つのクラスがあり，一つはBoole 制約充足問題（boolean constraint satisfaction problem）であり，もう一つは制約最適化問題（constraint optimization problem）である．後者はその解が大量の局所的な拘束条件を満たす度合いが最大になるようにシステムを設定するときに生じる数値的最適化問題であり [2]，本論文で議論するＣＳＰはこのクラスに属するものである．ＣＳＰの特徴は，知識を使ってどのように問題を解くかを考えるのではなく，知識として解が満足すべき条件を考えることと解が一つではなく複数存在する場合があることである．設計やスケジューリングなどの合成型問題は，一定の拘束条件のもとで，与えられた要求を満足する最適（または妥当）なシステム構成を生成する問題であり [3]，ＣＳＰの一つとして捉えることができる．先ほどの緊急時の問題では問題に関する制約をすべて満足する解を長い推論時間をかけて得るよりも解答として許容できる範囲内での最終的な解の近似解を実時間で得た方が有効である．しかし，ＣＳＰに対して従来の推論方法ではこの目的を実現するものがあまり存在しない．

本論文では上記の問題に対応するため，ＣＳＰに対して実時間で解として許容できる近似解を導出する推論機構の提案を行なう．ここで用いる「実時間」という言葉は，時々刻々と変化する環境下で，予め設定された制限時間内で解を導出するという意味ではなく，最終的な解が導出されるまでの長い時間に対して近似解が導出されるまでの相対的に短い時間という意味で用いている．本論文ではこの推論機構を用いた推論を実時間推論と呼ぶこととする．

ＣＳＰで与えられる制約については，解が必ず満足しなければならない制約と，必ずしも満足しなければ解として許容できないわけではないが満足した方がよい制約の二つに分けて考えることができる [4][5]．本論文では前者を絶対的制約（absolutely-satisfied constraint:ＡＳＣ），後者を希望的制約（desirably-satisfied constraint:ＤＳＣ）と呼ぶこととする．本論文ではＡＳＣとＤＳＣのすべての制約を満足する解を最適解，ＡＳＣのみまたはＡＳＣとＤＳＣの一部を満足する解を準最適解と呼ぶこととする．ユーザは実時間で最適解を得ることが理想ではある．それが不可能である場合は実時間で準最適解を得ることができれば，準最適解はＡＳＣを満足しているのでユーザはそれを利用し，とりあえず問題を解決することができる．また準最適解を得た後，その解をもとにした実際の実行までに時間がある場合には，準最適解が修正されより最適解に近似された準最適解を得ることも可能である．本論文では以下の章でまず実時間推論の推論方法の説明とその定性的性質の考察を示し，さらにその分散並列化による発展について述べる．最後に関連研究の考察とまとめを行なっている．

2 推論方法

2.1 制約充足実時間推論

実時間推論では問題解決のための制約に注目し，まず実時間で近似解を導出するために解答が必ず満足しなければならない制約だけを考慮して推論を行ない，その後その他の制約を考慮しながら近似解の修正を行ない，最終的な解

を導出する．実時間推論の推論機構について説明する．その処理手順のフローチャートを図1に示す．問題の分析を行ない，解答に関する制約がＡＳＣとそれぞれ異なる観点から得られるＤＳＣに分離されていることが前提となる．最初にＡＳＣだけの満足を考慮し推論を行ない，準最適解を求める．その後ＤＳＣのうち一つを選び，ＡＳＣと今選択したＤＳＣを満足する解を求め，これを新たな準最適解とする．そしてまた残っている考慮されていないＤＳＣのうちから一つを選び，ＡＳＣとその時点で考慮しているＤＳＣと今新たに選んだＤＳＣのすべてを満足する解を求め，新たな準最適解とする．このようにして一つづつ考慮するＤＳＣを増やしてゆき，ＡＳＣとすべてのＤＳＣを満足する解が求まった時点で推論が終了する．なお，準最適解を修正して最適解を推論する過程において選択された複数のＤＳＣとＡＳＣを満足する解を存在しなかった場合，そのＤＳＣの中から一つを選びその充足をあきらめる．そのため最終的な解はＡＳＣを必ず満足しているので準最適解であることは保証できるが．しかしＡＳＣとＤＳＣの関係により最適解が存在しない場合もあり，最終的解が必ず最適解であることは保証できない．またその満足をあきらめるＤＳＣを一つを選択する際に何をその判断基準にするかの議論については本論文では言及しない．これはＤＳＣの優先度など他の議論とともに扱われるべきであろう．

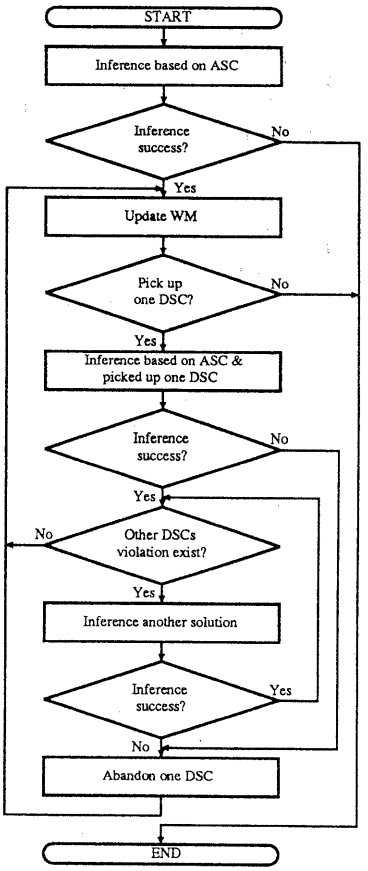

図 1: 実時間推論の処理手順フローチャート

2.2 例題

実時間の適用の例題について説明する．ＣＳＰの一つとして研究室におけるゼミナールのスケジューリング問題を取り上げる．ここでは実時間推論の動作説明のため，問題の設定を簡単なものとしている．

【ゼミ割り当て問題】

図2に示すように各ゼミの開催日時について制約を満足するように時間割上の候補場所に割り当てる問題を考える．予め問題に関する情報としてつぎの情報が与えられる．

- 時間割上の候補コマ（割り当て可能なコマ）
- どのゼミも割り当てができないコマ
- 割り当てを行なうゼミとその各ゼミの参加構成員
- 各参加構成員のゼミに参加できないコマ
- 各参加構成員ができればゼミに参加したくないコマ

予め与えられる解に関する制約にはつぎのものがある．

【ＡＳＣ】

- 時間割上の一つの候補コマには最大一つのゼミしか割り当てることができない．
- どのゼミにも割り当てができないコマにはゼミを割り当てない．
- 割り当てを行なうゼミはすべて割り当てを行なわなければならない．
- 割り当てられたゼミにはその構成員全員が出

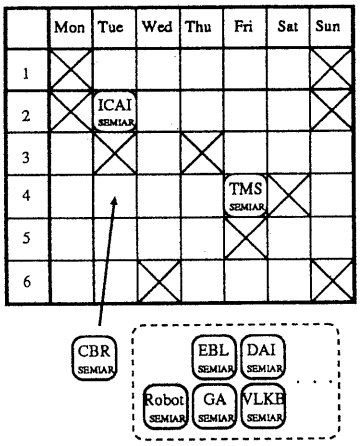

図 2: ゼミ割り当て問題

席できなければならない．

【ＤＳＣ】[例]

- 各参加構成員ができれば参加したくないコマにはその参加構成員が所属するゼミを割り当てたくない．
- 各参加構成員一人が三コマ連続してゼミに参加したくない．
- 各参加構成員一人が一コマめのゼミに一週間に二つ以上参加したくない．
- 各参加構成員一人が六コマめのゼミに参加し，かつつぎの日の一コマめのゼミに参加することを避けたい．
- 各参加構成員一人が一つのゼミだけに参加する日を作りたくない．

解については全解を求めるのではなく，制約を満足する解を一つ見つければ推論を終了することとする．コマ数，ゼミ数，構成員数を増減させることにより扱うデータ数を調節し，またより複雑な制約を付加することにより，より複雑な制約を扱えば十分に複雑な問題となる．

この問題の解決に実時間推論を適用する．まず，ＡＳＣのみを考慮して割り当てるゼミをすべて時間割上に割り当てる．この解は四つのＡＳＣを満足する準最適解であり，参加構成員全員が参加できるようにゼミが割り当てられているので，ユーザはこの時点で準最適解をとりあえず使用できる．つぎにＤＳＣを一つづつを考慮するＤＳＣを増やしながら，それぞれのＤＳＣをできるだけ多く満足するように準最適解の修正を繰り返す．この過程で解が修正される度に例えば三コマに連続して参加する人や１つのゼミだけに参加する日を持つ人が減少してゆく．推論が成功し，すべてのＡＳＣ，ＤＳＣを満足する解が最適解として獲得され推論が終了する．勿論，場合によってはすべてのＤＳＣを同時に満足する解が存在せず，推論を終了することもある．

2.3 定性的考察

ここでは，実時間推論と従来の一般的な推論との比較を行ないながら実時間推論の定性的考察を行い，さらにその有効性を向上させるための論点を整理する．ここでいう一般的な推論とはＡＳＣとＤＳＣの分離などを行なわず両方の制約を同時に考慮して推論を行なう方法を指す．図3では実時間推論と一般的な推論を解とその導出時間との関係から比較を行なっているものである．図3のグラフにおいて縦軸は解が最適解に対しその時点で導出された解がどの程度制約を満足しているかの割合を示し，横軸はその解が導出されるまでの時間を示している．なお，ここでは最適解が存在し，最適解を一つ推論するまでの時間を考えている．一般的な推論では点 (c) で最適解が導出されるまでユーザにはまったく解が与えられず，その過程において準最適解を得ることができない．一方実時間推論では点 (b) において t_{ASC} の時間で最適解の制約と比較し p_{ASC} の割合で制約を満足する準最適解がユーザに与えられ，その後ＤＳＣを考慮しながら準最適解の修正を繰り返し，最終的には $t_{ASC}+t_{DSC}$ の時間で最適解を導出する．図3はある条件での比較結果であり，実際には様々な要因によりグラフの実時間推論の軌跡は変化する．その要因と影響を考察すると以下のものが考えられる．

１． 点 (b) が前後する．

t_{ASC} が短くなるの場合はＡＳＣが緩く簡単な問題やＤＳＣがきつく複雑な問題であると考えられる．

２． 点 (a) が上下する．

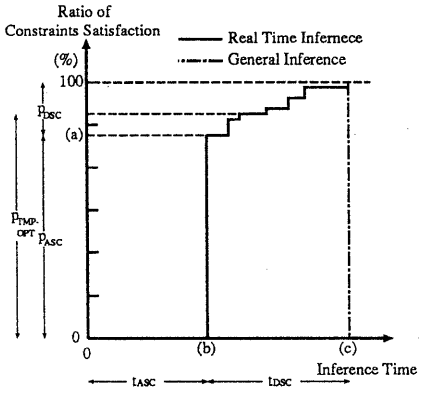

図 3: 実時間推論と一般的な推論の比較図

p_{ASC}が小さくなる場合はＤＳＣの数に比べＡＳＣの数が少ない問題であると考えられる．

3． $t_{ASC}+t_{DSC}$の時間が点 (c) に対し前後する．

どのような要因により$t_{ASC}+t_{DSC}$が変化するかについては分析できていない．しかし実時間推論を使用する目的を考えると$t_{ASC}+t_{DSC}$が点 (c) よりも長くなる場合でも実時間推論を使用する意義はあるであろう．

4． 点 (b) 以降，準最適解を修正すると$p_{TMP-OPT}$が前の状態よりも低下する．

点 (b) で準最適解が得られた後の段階においてはあるＤＳＣを満足するためその時点まで制約を満足してきたＤＳＣの一つの満足を放あきらめることがあるので，そのＤＳＣの違反の発生により修正後$p_{TMP-OPT}$がの修正前よりも低下する．修正ごとに$p_{TMP-OPT}$が必ず上昇するという意味で必ず修正により解が最適解に近づくという保証はない．ただし，どんな修正を行なってもＡＳＣは必ず満足されているので，$p_{TMP-OPT}$がp_{ASC}未満になることはない．

1．と3．の分析をもとに推論時間の短縮を図ることを考える．t_{ASC}の短縮については従来の効率化手法によりそれを実現すればよいと考えられるので，ここでの議論は行なわない．一方t_{DSC}の短縮については従来の手法では困難である．2． で点 (a) が低い，つまりＤＳＣの数が多い場合への対応を考慮し，この推論過程を分散・並列化してt_{DSC}の短縮を図るものとする．

3 実時間推論の分散・並列化

3.1 分散協調問題解決手法の適用

実時間推論では制約をＡＳＣとＤＳＣに分離し，それぞれを用いて問題解決を行なっているので分散型と言える．ＡＳＣを満足する準最適解を得た後，ＤＳＣを満足しながら最適解に近づいていく必要がある．制約の分離という点に注目して考察を行なうと，それぞれのＤＳＣを考慮して推論を行なう部分については必ずしもシーケンスに処理を行なう必要はなく，またＤＳＣ間の関係を分析すれば他のＤＳＣに影響を与えないものも存在する．つまり実時間推論において準最適解を実時間で求め，さらにできるだけ短い時間で最適解を求めることが可能となればさらによい．そこで，本章ではＤＳＣの充足に分散協調問題解決の手法を適用し，準最適解から最適解を求める推論時間の短縮を図ることとする．実時間推論を分散・並列化するためには，協調問題解決における黒板モデルを用いた結果共有の考え方を導入する．結果共有は複数のエージェントが，独立の見地から問題の部分的な解を持ち寄る協調の形式であり，問題解決の過程はデータ駆動である．各エージェントの処理は他のエージェントによって供給されたデータに基づいて行なわれる [6]．本研究ではこの考え方を以下に説明するように実時間推論に適用し，その分散・並列化を実現する．

3.2 分散・並列化方法

3.2.1 概要

エージェントは後に説明する管理エージェントとＤＳＣ推論エージェントを用いる．詳細は後で述べる．その推論手順の概要は管理エージェントがＡＳＣをもとに準最適解を推論し，その解をもとにそれぞれの観点から得たＤＳＣの情報を持つ並列に動作する複数のＤＳＣ推論エージェントがその解をもとに制約チェックと

推論を行ない，自分が担当するＤＳＣを満足する解を管理エージェントに返す．管理エージェントはそれぞれのＤＳＣ推論エージェントが返してくる解をもとに整合性を保ちながら作業記憶の書き換えを行ない，その繰り返しにより最終的に最適解を導く．ＤＳＣ推論エージェントはその動作のトリガを基本的に作業記憶の書き換えとするデータ駆動で動作する．黒板モデルに当てはめると管理エージェントの作業記憶が黒板に対応する．黒板の書き換えは管理エージェントのみが行なう．ＤＳＣ推論エージェントは管理エージェントからのメッセージを通してのみ黒板の情報を獲得でき，かつ黒板の情報の書き換えはすべて管理エージェントへの依頼として行ない，直接黒板にアクセスは行なわない．

3.2.2 エージェント間のメッセージ通信

エージェントについてはつぎの二種類のエージェントを設定する．

• **管理エージェント**（manage agent）
システムに一つだけ存在する．ＡＳＣを考慮した準最適解の推論，ＤＳＣ推論エージェントへのタスクの依頼と解答の受けとり，ＤＳＣ推論エージェント間の競合の解決，作業記憶のデータ更新などを行なう．

• **ＤＳＣ推論エージェント**（DSC inference agent）
ＤＳＣそれぞれ一つについて一つ存在する．管理エージェントから送られる作業記憶のデータをもとに自分が担当するＤＳＣについて違反がないかのチェックとＡＳＣを満足し担当のＤＳＣの違反を解消する解を推論し，管理エージェントに報告する．

エージェント間の通信は管理エージェントとＤＳＣ推論エージェント間で非同期に行なわれる．通信を非同期とする理由は，普通複数のＤＳＣ推論エージェントが並列に解のチェック，推論を行なうので与えられるデータの設定，ＤＳＣのきつさ，分散させたマシンのCPUパワーなどにより管理エージェントがＤＳＣ推論エージェントから結果を受けとる期間を確定することが困難であり，また推論時間の短縮を図るという目的からである．二者のエージェント間で通信されるメッセージにはつぎのものがある．

【管理エージェントからＤＳＣ推論エージェントに送られるメッセージ】

• **初期チェックリクエスト**（initial check request）
準最適解を推論したのち，その解についてＤＳＣの違反チェックと担当ＤＳＣを満足する解の推論を依頼する．同時に準最適解の情報が送られる．

• **修正受容**（modify accepted）
ＤＳＣ推論エージェントから受けとった「修正リクエスト」の返答としてその修正を受容したことを知らせる．

• **修正拒否**（modify rejected）
ＤＳＣ推論エージェントから受けとった「修正リクエスト」の返答としてその修正を拒否して他の解の推論を依頼する．

• **アップデイトチェックリクエスト**（update check request）
他のＤＳＣ推論エージェントの「修正リクエスト」により書き換えられた作業記憶の情報を伝え，そのチェックと修正の推論を依頼する．

• **推論終了メッセージ**（inference terminate message）
すべての推論が終了したことを知らせ，ＤＳＣ推論エージェントの処理の終了を指示する．

【ＤＳＣ推論エージェントから管理エージェントに送られるメッセージ】

• **ＤＳＣ違反なし**（no DSC violation）
「初期チェックリクエスト」や「アップデイトチェックリクエスト」によりチェックを行なった結果，制約違反がなかったことを知らせる．

• **ＤＳＣ満足不可**（DSC satisfy impossible）
「初期チェックリクエスト」によりチェックを行なった結果，制約違反が発見されその違反を解消する解の推論を行なった結果推論に失敗したことを知らせる．同時に違反の箇所と違反の理由（原因）が送られる．

• **修正リクエスト**（modify request）
「初期チェックリクエスト」や「アップデイトチェックリクエスト」によりチェックを行なった結果，制約違反が発見されその違反を解消するための推論を行なった結果推論に成功しその

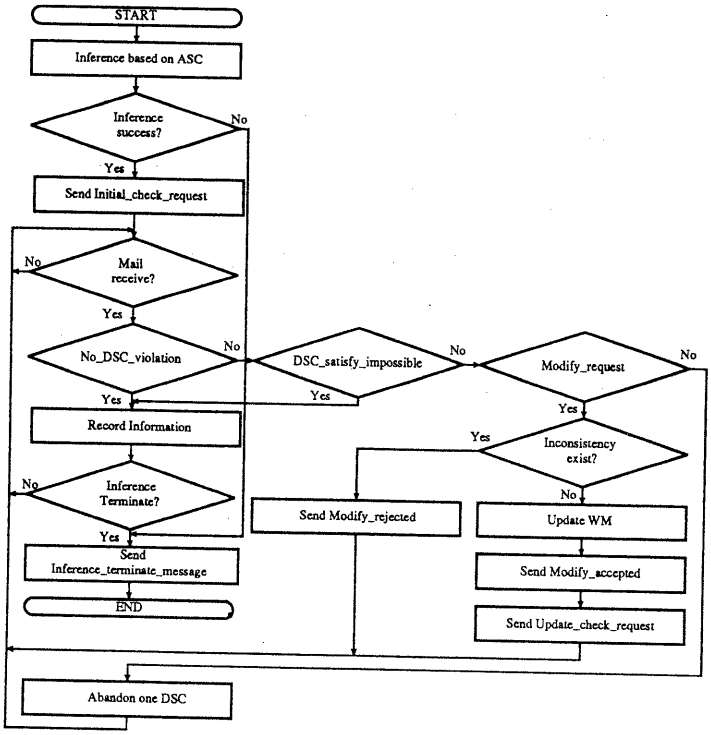

図 4: 管理エージェントの処理手順フローチャート

解を知らせる．同時に違反していた箇所と違反だった理由（原因）が送られる．

• **修正不可**（modify impossible）

「修正リクエスト」を送った後,「修正拒否」をうけとり制約違反を解消する他の解を推論した結果，もう他に解がなく推論に失敗したことを知らせる．

3.2.3 管理エージェント

管理エージェントの具体的な処理手順について説明する．図4にそのフローチャートを示す．まず，ＡＳＣのみを考慮して推論を行ない準最適解を導出する．つぎに各ＤＳＣ推論エージェントに同時にそれぞれ「初期チェックリクエスト」を送り，メッセージが届くのを待つ．以後届いたメッセージの種類により処理を進める．「ＤＳＣ違反なし」の場合は単にその記録をとる．「ＤＳＣ満足不可」の場合はその記録をとり，そのＤＳＣを最適解の推論の考慮から除外する．この二種のメッセージが届いた後には推論の終了判定を行なう．この判定はすべてのＤＳＣ推論エージェントからメッセージが送られており，それらが「ＤＳＣ違反なし」または「ＤＳＣ満足不可」または「修正不可」のいずれかであるならば各ＤＳＣ推論エージェントに「推論終了メッセージ」を送り，推論を終了する．「修正リクエスト」の場合はまずその整合性のチェックを行ない，整合性がない場合には「その修正リクエスト」を送ってきたＤＳＣ推論エージェントに「修正拒否」を送る．整合性がある場合にはその修正を採用し，作業記憶の状態の書き換えを行なう．そしてその「修正リクエスト」を送ってきたＤＳＣ推論エージェントに「修正受容」を送り，その時点で「修正リクエスト」を送ってきている他のＤＳＣ推論エージェントには「修正拒否」と「アップデイトチェックリクエスト」の二つのメッセージ

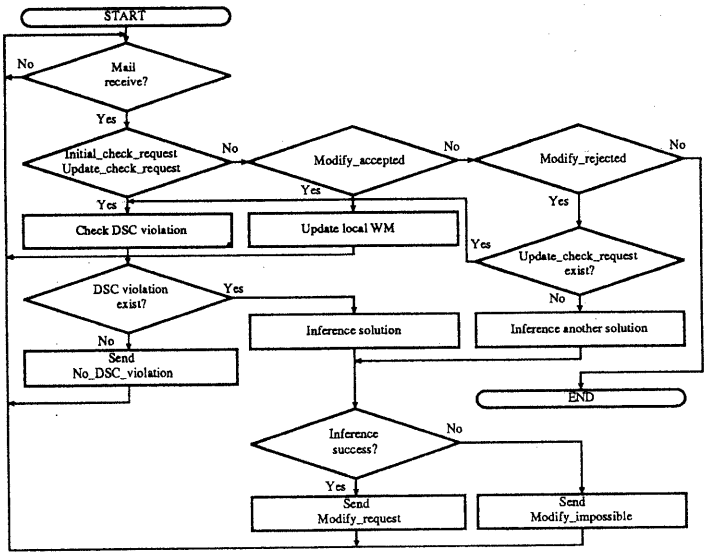

図 5: ＤＳＣ推論エージェントの処理手順フローチャート

を送り，その他のＤＳＣ推論エージェントには「アップデイトチェックリクエスト」を送る.「修正不可」だった場合はそのメッセージを送ってきたＤＳＣ推論エージェントまたはそのＤＳＣ推論エージェントと制約を満足するうえで競合をおこしている他のＤＳＣ推論エージェントの中の一つについて，その満足をあきらめる．これら，一連の処理が終るとＤＳＣ推論エージェントからのメッセージを待つループに入り，推論が最終的に終了するまで繰り返される．

3.2.4 ＤＳＣ推論エージェント

ＤＳＣ推論エージェントの処理手順について説明する．図5にそのフローチャートを示す．それぞれのＤＳＣ推論エージェントはシステムが起動された時点で管理エージェントと同時に起動される．この時点ですでにＤＳＣ推論エージェントはＡＳＣとそれぞれが担当するＤＳＣの知識は与えられている.「初期チェックリクエスト」が送られることを待つ．そのメッセージが届いたら，まずＤＳＣに関する違反がないかのチェックを行なう．違反がなかった場合は「ＤＳＣ違反なし」を送る．違反があった場合にはその違反を解消する解を推論する．推論が成功すれば「修正リクエスト」を修正箇所とともに送り，その返答を待つ．以後その返答の種類により処理を進める.「修正受容」の場合はローカルの作業記憶の情報の書き換えを行なう.「修正拒否」の場合はまずその時点で「アップデイトチェックリクエスト」が届いていれば自分が送った「修正リクエスト」の変更内容を放棄し，そのチェックを行なう.「修正拒否」が届いていなければ「修正リクエスト」で送った解以外の解を推論する．別解の推論に成功すれば「修正リクエスト」を送る．推論に失敗すれば「修正不可」を送る.「アップデイトチェックリクエスト」の場合は「初期チェックリクエスト」の場合と同様の処理を行なう.「推論終了メッセージ」がの場合はＤＳＣ推論エージェントの処理を終了する．なお，これらの処理中に複数の「アップデイトチェックリクエスト」が届いた場合には各処理が終了し，管理エージェントからのメッセージを待つ過程まで最新のものを保存しておくこととする．

3.3 並列・分散化した場合の例題

問題の設定は2．2章で説明したものと同様とする．エージェントは一つの管理エージェントとＤＳＣをそれぞれ担当する四つのＤＳＣ推論エージェントが存在する．この時点でＤＳＣ推論エージェントはＡＳＣとそれぞれが担当するＤＳＣについての情報を持っている．実時間時間推論を並列・分散化した場合でもそれを行なわない従来の方法と同様に管理エージェントがＡＳＣのみを考慮して準最適解を導出する．つぎに従来の方法ではＤＳＣの満足をシーケンスに行なっていたが，ここではその部分をＤＳＣ推論エージェントに並列的にチェック，推論させる．ここで，各ＤＳＣ推論エージェントから例えば，ＡＳＣを満足し一コマめのゼミに一週間に二回以上参加する人がいない解やＡＳＣを満足し，六コマめのゼミに参加した人がつぎの日の一コマめに参加することがない解などが「修正リクエスト」として管理エージェントに届く．これらの「修正リクエスト」をもとに管理エージェントが準最適解の修正を繰り返し，最適解を推論して終了する．

3.4 問題点とその解決法

ここでは実時間推論の分散・並列化を行なう上での，考えるべき問題点とその解決方法について説明する．

• **無限ループのチェックと動作停止の保証**
管理エージェントは準最適解をもとに修正を繰り返し最適解を求める過程において作業記憶の履歴を記録しその情報をもとに解の再現性のチェックを行ない，無限ループを防止する．ＤＳＣ推論エージェントは「修正リクエスト」を送り受理された解の履歴を記録しその履歴と同じ「修正リクエスト」を出すことを防止する．動作停止の保証についてはＤＳＣ推論エージェントの終了条件によりその動作停止が保証できる．

• **作業記憶の整合性の維持と修正要求の処理**
非同期通信を行なうことからＤＳＣ推論エージェントがチェック，推論を行なっている間に作業記憶の書き換えが発生する．管理エージェントは「修正リクエスト」の処理を行なう際に必ずその整合性のチェックを行ない，また同時に一つの「修正リクエスト」の処理しか行なわないことにより，整合性の維持を実現している．

• **ＤＳＣ推論エージェントへの割り込み処理**
上の問題点に関連してＤＳＣ推論エージェントはその処理に長い時間がかかるとその間につぎつぎと「アップデイトチェックリクエスト」が届く事態となる．この場合割り込み処理によりチェックや推論の途中でも新たな「アップデイトチェックリクエスト」が届くたびにそのデータをもとに最初からチェック，推論をやり直す処理が考えられるが，そのような処理を行なうと極端な場合そのＤＳＣ推論エージェントからの情報がまったく管理エージェントに届かなくなる．そのような事態を避けるため，割り込み処理を行なわず，現在行なっている処理が終了した時点で，その終了結果と届いているメッセージの内容を検討することで対応している．具体的には，終了結果が「修正リクエスト」の場合，その時点で届いている最新の「アップデイトチェックリクエスト」との整合性をチェックし，整合性がない場合にはその状態の違いを新たな「アップデイトチェックリクエスト」として処理を行なう．この処理により管理エージェントはＤＳＣ推論エージェントの処理結果を比較的早く獲得できる．

4 関連研究

本論文では実時間推論の発展として部分的な分散・並列化の方法を述べた．これに関連する研究に複数のエージェントが関連するＣＳＰである分散制約充足問題（distributed constraint satisfaction problem:ＤＣＳＰ）の定式化とその解法に関するものがある [7]．文献 [7] によれば，ＤＣＳＰはＣＳＰの変数が複数のエージェントに分散された問題である．本論文でも３章において分散・並列化によるＣＳＰの解決法の一方法を提案したが，必ずしも今回提案した方法が効率的に問題を解決できるがどうかの分析についてはまだ十分ではない．この部分に関しては他の有効な解決法も含めさらなる検討

が必要である．例えば，今回取り上げたゼミ割り当て問題の例で考えればＤＳＣをそれぞれのエージェントに対応させ分散を行なうのではなく，割り当てを行なう各ゼミをそれぞれのエージェントに対応させ，それぞれのゼミ間の制約に注目して問題を解決するという方法も考えられる．文献 [7] と本研究の違いについては文献 [7] が対象としているＣＳＰのクラスは１章で述べた Boole 制約充足問題であるが本研究は制約最適化問題のクラスを対象としいるという点がある．しかしＣＳＰを分散環境に拡張するという点においては共通の議論点を多く見い出すことができる．

また近似解を獲得する方法として実時間推論をとらえる立場からシミュレーテッド・アニーリング（ＳＡ）法 [8] がある．ＳＡ法では最適解の近似解を求めることができるが，その解の制約についてＡＳＣ，ＤＳＣという考え方でとらえると必ずしもＡＳＣが満足されているという保証はなく，ユーザの側でチェックを行なわなければならない．また複数のＤＳＣ間の優先度の指定については比較し評価関数の重み付けという形で行なわなければならず，実時間推論と詳細な調整が難しいと考えられる．

5　おわりに

本論文では制約充足問題において実時間で準最適解を求める推論方法について提案を行なった．さらに最適解を推論する時間の短縮を図るため，実時間推論の分散・並列化手法を提案を行なった．この実時間推論の適用には絶対的制約と希望的制約の明確な表現と分離が行なえる問題であることが前提ではあるが，この前提が成立する問題については大変有効である．なお，現在実時間推論を実装したシステムを用いて定量的な考察を行なうための実験をスケジューリング問題を対象として行なっている．この結果については後の機会に報告を行ないたいと考えている．

今後の課題としては管理エージェントとＤＳＣ推論エージェント間の大きな通信コストの削減方法，システムが希望的制約間の競合依存関係の分析を行ない，最適解を効率的に推論する方法や実時間推論を実現する有用なメカニズムの提案などが考えられる．

参考文献

[1] Forgy,C.L.: Rete: A Fast Algorithm for Many Pattern/Many Object Pattern Match Problem, *Artif. Intell.*, Vol. 19, pp.17-37 (1982).

[2] A. K. Mackworth: “制約充足”, 人工知能大辞典 Ed. S. C. Shapiro (大須賀節雄 監訳), pp.685-690, 丸善 (1991).

[3] 上野晴樹：エキスパート・システム概論，情報処理，Vol.28，No.2，pp.147-157 (1987).

[4] 戸沢義夫, 沼尾雅之, 森下真一: 制約論理プログラミング（淵一博 監修），共立出版, 第１０章 (1989).

[5] 佐藤健: 解釈の順序による柔らかい制約の定式化，情報処理学会論文誌，Vol.31，No.6, pp.772-782 (1990).

[6] 石田亨: 協調問題解決，第１９回人工知能セミナー講演テキスト，人工知能学会，pp.11-30 (1989).

[7] 横尾真，エドモンド Ｈ．ダーフィ，石田亨，桑原和宏: 分散制約充足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).

[8] S.Kirkpatrick, C.D.Gelatt and M.P.Vecci: “Optimization by simulated annealing”, *Science*, Vol. 220, No. 4598, pp.671-683 (1983).