原著論文

Original Paper

限量記号付き分散制約最適化問題のための分散探索手法

Distributed Search Methods for Quantified Distributed Constraint Optimization Problem

松井 俊浩
Toshihiro Matsui
名古屋工業大学
Nagoya Institute of Technology
matsui.t@nitech.ac.jp

Marius C. Silaghi
Marius C. Silaghi
フロリダ工科大学
Florida Institute of Technology
msilaghi@fit.edu

平山 勝敏
Katsutoshi Hirayama
神戸大学大学院海事科学研究科
Graduate School of Maritime Sciences, Kobe University
hirayama@maritime.kobe-u.ac.jp

横尾 真
Makoto Yokoo
九州大学大学院システム情報科学研究院
Graduate School of Information Science and Electrical Engineering, Kyushu University
yokoo@is.kyushu-u.ac.jp

松尾 啓志
Hiroshi Matsuo
名古屋工業大学大学院工学研究科
Graduate School of Engineering, Nagoya Institute of Technology
matsuo@nitech.ac.jp

keywords: quantified, Distributed Constraint Optimization, multi-agent, cooperation

Summary

Distributed Constraint Optimization problems (DCOPs) have been studied as a fundamental model of multi-agent cooperation. In traditional DCOPs, all agents cooperate to optimize the sum of their cost functions. However, in practical systems some agents may desire to select the value of their variables without cooperation. In special cases, such agents may take the values with the worst impact on the quality of the result reachable by the optimization process. Similar classes of problems have been studied as Quantified (Distributed) Constraint Problems, where the variables of the CSP have existential/universal quantifiers. All constraints should be satisfied independently of the value taken by universal variables. In this paper, a Quantified Distributed Constraint Optimization problem (QDCOP) that extends the framework of DCOPs is presented. We apply existential/universal quantifiers to distinct uncooperative variables. A universally quantified variable is left unassigned by the optimization as the result has to hold when it takes any value from its domain, while an existentially quantified variable takes exactly one of its values for each context. We consider that the QDCOP applies the concept of game tree search to DCOP. If the original problem is a minimization problem, agents that own universally quantified variables may intend to maximize the cost value in the worst case. Other agents normally intend to optimize the minimizing problems. Therefore, only the bounds, especially the upper bounds, of the optimal value are guaranteed. The purpose of the new class of problems is to compute such bounds, as well as to compute sub-optimal solutions. For the QDCOP, we propose solution methods that are based on min-max/alpha-beta and ADOPT algorithms.

1. は じ め に

分散制約最適化問題 (DCOP) はマルチエージェントの協調問題解決の基礎的なモデルとして研究されている [Mailler 04, Modi 05, Petcu 05, Silaghi 09, Yeoh 08, Zivan 08]．DCOP では，マルチエージェントシステムにおける協調問題解決が，エージェント間に分散して配置された離散的最適化問題として表現される．エージェントの状態や意思決定は，変数により表される．エージェント間の関係は，制約およびその評価値を表すコスト関数により表される．コスト関数値の合計を大域的に最適化する最適解を分散探索アルゴリズムにより計算する．分散ミーティングスケジューリング，電源施設や分散センサ網の資源割り当て問題が DCOP として形式化されている [Farinelli 08, Kumar 09, Maheswaran 04].

従来研究の DCOP では，全てのエージェントが協調してコスト関数の合計を最小化する．しかし，実際的なシステムをモデル化するとき，コスト関数の最小化には協調せずに自変数値を決定するエージェントが存在しうる．例えば，電力供給ノードと需要ノードから構成される電

力スマートグリッド網を考える．このようなシステムにおける供給ノードは，節電等に対して一部で非協力的な需要ノードからの要求に応えられる頑健なプランを立てる必要がある．

別の例として，複数の自律的なセンサと複数の観測対象物から構成される観測システムが考えられる．観測対象物の一部が侵入者などのように逃避的にふるまう場合，センサの観測資源はこのような観測対象物の挙動にも対抗しうるようにスケジュールされなければならない．特に，非協力的なエージェントが常に敵対的な場合には，敵対的なエージェントは最適化の過程で解の評価値を可能な限り悪くする変数値を選択する．

そこで本研究では，通常の変数と非協力的な変数を区別するために，存在限量子または全称限量子を適用する．全称限量された変数は非協力的な変数を表し，最適化の過程では，その変数値は値域に含まれるいずれの値もとりうるものとして扱われる．これに対して存在限量された変数は通常の変数を表し，各解において単一の変数値をとる．

近年，(分散) 制約充足問題におけるこのような問題のクラスが，限量記号付き制約充足問題 (QCSP) [Chen 04] および限量記号付き分散制約充足問題 (QDCSP) [馬場 11] として研究されている．これらの問題では，制約充足問題の変数が限量され，すべての制約が全称限量された変数の値によらず充足されることが目的とされる．文献 [馬場 11] では，非同期バックトラッキングを用いる QDCSP の解法が示されている．

DCOP から限量記号付き DCOP(QDCOP) への自然な拡張の一つは，ゲーム木探索の概念を導入することである．たとえば，もしも元の問題が最小化問題であれば，全称限量された変数を持つエージェントは最悪の場合にコスト値を最大化するような変数値をとると考えられる．他のエージェントは従来通り最小化問題を最適化しようとする．そのため最適コストには，従来の解法で得られる下界値および新たに導入される上界値のみが保証される．

このように拡張された問題の一つの目的として，最適コストの上界値およびその場合の準最適解を求めることが挙げられる．これは，事前に模擬的な問題解決を行うことにより，最悪のコストとその場合の解を明らかにすることが必要な場合などに有用であると考えられる．

集中型の問題解決おいて，限量記号付きの制約充足問題や最適化問題は，最悪の場合を想定した解を見積もるためのモデルの基礎として一般的な概念と思われる．分散システム上では，敵対的なエージェントやそのチームが存在する前提は自然であり，そのもとでの最適化問題と解法から協調問題解決のプロトコルを理解し，応用の可能性を広げてゆく試みには意義があると考える．

分散制約最適化問題やその関連研究においては，従来の単純な定式化では不十分な問題を扱うために，動的変化，不確実さ，多目的などの要素を含む拡張された問題と解法が検討されている [Petcu 07, Delle Fave 11, Léauté 11]．限量記号付を伴う分散制約最適化問題は，多目的最適化の特別な例として捉えることができる．

集中型の問題解決に対して，分散協調型の問題解決手法で課題となる点に，個々のエージェントがどの程度まで制御されうるかという点が挙げられる．分散制約充足問題や最適化問題は，少なくとも最適化のためのプロトコルを実行するための最低限の基盤の存在を前提とする．従って，協調のための基盤を構成し得ないエージェントを含む場合には，それを吸収するための手法を用いることが考えられる．例えば，センサ網の例では，実際の侵入者は協調の対象では無いが，その侵入者を観測できる付近のノード上で敵対的な行動をするエージェントとしてシミュレーションされることは自然な方法と考えられる．また，電力網の例に示されるような社会基盤の資源を共有する場面では，参加者のエージェントに一定の通信の規則などを整備することは，本質的に必要と思われる．資源の要求について節約と消費というチームに分かれるという前提は動機づけのためのやや極端な例ではあるが，事前に資源の最悪の要求量を見積もる目的を表現する一例と考えられる．より高度には個々の役回りや要求の程度を表現するような評価関数や制約を導入することも考えられる．

本研究では，min-max 法，alpha-beta 法，および DCOP の解法 ADOPT [Modi 05] に基づく，QDCOP のための解法を示す．疑似木に基づく DCOP の解法が，ゲーム木探索に一般化されることを示し，解法の効果を実験により評価する．

本論文の概要は次のようである．まず，2 章では DCOP, QCSP, QDCSP および QDCOP を含む問題の定義を示す．次に，QDCOP の解法を 3 章に示す．4 章では実験により提案手法を評価する．5 章では関連研究について述べる．6 章で結言を述べる．

2. 問 題 の 定 義

まず，従来手法として DCOP，QCSP および QDCSP について述べ，限量記号付き DCOP の定義を示す．

2·1 DCOP

分散制約最適化問題 (DCOP) は (A, X, D, C, F) により定義される．ここで，A はエージェントの集合．X は変数の集合，D は変数の値域の集合族，C は二項制約の集合，F は二項制約のコストを表す評価関数の集合である．

エージェント i は自身の変数 $x_i \in X$ を持つ．x_i は有限で離散の値域 $D_i \in D$ の値を取る．変数 x_i の値はエージェント i のみが決定できる．C の要素である制約 $c_{i,j}$ は x_i と x_j の関係を表す．変数値の割り当て $\{(x_i, d_i), (x_j, d_j)\}$ のコストは，F の要素である関数 $f_{i,j}$ により，$f_{i,j}(d_i, d_j) : D_i \times D_j \to \mathbb{N}_0$ のように定義される．ただし，$\mathbb{N}_0$ は 0 を

含む整数を表す．大域的なコストの合計

$$\sum_{f_{i,j}\in F,\ \{(x_i,d_i),(x_j,d_j)\}\subseteq\mathcal{A}} f_{i,j}(d_i,d_j) \tag{1}$$

を最小化するような最適解 $\mathcal{A}$ を発見することが目的である．以降では表記を簡単にするために，必要に応じてエージェント i と変数 x_i を厳密に区別せずに用いる．

2·2 QCSP/QDCSP

限量記号付き制約充足問題 (QCSP)[Chen 04] は制約充足問題を拡張した問題である．従来の制約充足問題は (X,D,C) により定義される．X は変数の集合，D は値域の集合族，C は制約の集合である．x_i は値域 $D_i\in D$ の値をとる. CSP の解は C に含まれる制約をすべて充足する変数値の割り当て $\{(x_0,d_0),\cdots,(x_n,d_n)\}$ である．

QCSP では従来の CSP の定義に加えて，限量子の系列が定義される．QCSP は $Q.C = q_0x_0\cdots q_nx_n.C$ の形式により表される. Q は変数の系列であり，q_i は存在限量子 $\exists$ か全称限量子 $\forall$ のいずれかである．

QCSP $Q.C$ の意味は次のように再帰的に定義される．C が空であれば，$Q.C$ は真である．Q が $\exists x_0q_1x_1\cdots q_nx_n$ の形式であれば，$Q.C$ は $q_1x_1\cdots q_nx_n.(C\cup\{x_0=d\})$ であるような $d\in D_0$ が存在しその時にかぎり真である．Q が $\forall x_0q_1x_1\cdots q_nx_n$ の形式であれば，$Q.C$ は $q_1x_1\cdots q_nx_n.(C\cup\{x_0=d\})$ が $d\in D_0$ なるすべての値 d について真でありそのときにかぎり真である．それ以外の場合には，$Q.C$ は偽である．

限量記号付き分散制約充足問題 (QDCSP)[馬場 11] では，変数はエージェントに分散して配置される．関連研究 [馬場 11] では，非同期バックトラッキング探索アルゴリズムを拡張した解法が示されている．

2·3 QDCOP

本研究では DCOP に基づく問題である限量記号付き分散最適化問題 (QDCOP) を導入する．QDCOP では DCOP の定義に加えて，QCSP や QDCSP と同様に，限量された変数の系列が定義される．QDCOP は $Q.(C,F) = q_0x_0\cdots q_nx_n.(C,F)$ の形式により表される．Q は変数の系列を表し，q_i は存在限量子 $\exists$ か全称限量子 $\forall$ のいずれかである．

基本的には，QDCOP の目的は，その問題に対応する DCOP の大域的に最小のコストをとる最適な解を求めることである．しかし，問題に含まれる限量子のために最適化の意味が変更される．存在限量された変数は従来の DCOP の変数と同様である．その一方で，全称限量された変数はいずれの値も取りうる．したがって，その最適解はもとの DCOP とは異なる場合がある．ある DCOP には唯一の最適なコスト値が定義されることに対し，ある QDCOP にはその問題の最適コストの境界が定義される．

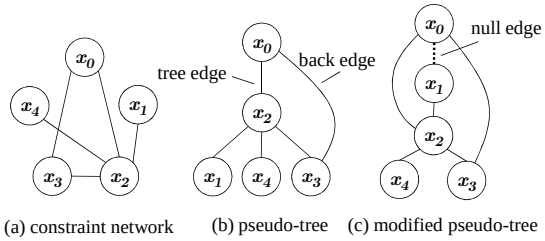


図 1 制約網と疑似木

従来の DCOP の最適コストは，QDCOP では最良の場合のコストであるため，これにより最適コストの下界が定義される．最悪の場合には，全称限量された変数は，コスト値を可能な限り増加する変数値を取る．したがって，最悪の場合の最適コストにより最適コストの上界が定義される．このクラスの問題は，ゲーム木探索 [Knuth 75, Russell 03] における問題と類似する．

本研究では，QDCOP の最悪の場合の最適解に注目する．また，全称限量された変数を持つエージェントは敵対的な行動をとるが，コスト値や解の計算は，いずれのエージェントも協調的に実行するものとする．すなわち，コスト値や解の計算などの分散処理の基盤は保証されているシステムにおいて，非協力的な振舞いをするエージェントが存在する場合を検討の対象とする．

これは，例えば，全エージェントが共有するエネルギーの消費量を最小化することに協力的なエージェントと非協力的なエージェントが存在する際に，非協力的なエージェントの要求を最大限受け入れる各エージェントのプランを決定する処理として捉えられる．また，将来に部分的な故障により協調できなくなる恐れのあるエージェントを仮想的な敵役として振舞わせ，事前に最悪の場合の最適コストおよび解を計算するオフライン処理と捉えることもできる．

3. QDCOP のための分散探索手法

QDCOP の解法を提案する．提案手法は，制約網に対応する疑似木に基づく，DCOP の厳密解法を拡張した手法である．まず，QDCOP のための疑似木およびコストの計算について述べる．そして，3 つの解法，min-max ADOPT, alpha-beta ADOPT, bi-threshold ADOPT を示す．

3·1 ボトムアップに生成される疑似木

疑似木は [Petcu 05] 制約網に含まれる変数に順序を定義するグラフ上の構造である．ここで，変数の順序とは，探索において，変数を展開する順序を意味する．疑似木は制約網の生成木を含む．例えば，図 1 (a) の制約網から同図 (b) の疑似木が生成される．疑似木では，元の制約網の辺が，生成木の辺である木辺と，それ以外の後退辺に分類される．木辺は二変数間の順序関係を表す．すなわち親の変数が子の変数よりも優先される．異なる部分木

表 1 解法の比較

解法	min-max ADOPT	alpha-beta ADOPT	bi-threshold ADOPT
特徴	min-max 法	alpha-beta 法	ADOPT の threshold を拡張
コスト値の上下界	$lb_{d,j}, ub_{d,j}, lb_i(d), ub_i(d), lb_i^*, ub_i^*$	同左	同左
枝刈りの影響を受ける上下界	なし	$lb_{d,j}^{\alpha\beta}, ub_{d,j}^{\alpha\beta}, lb_i^{\alpha\beta}(d), ub_j^{\alpha\beta}(d), lb_i^{\alpha\beta*}, ub_j^{\alpha\beta*}, lb_i^{\alpha\beta*-}, ub_i^{\alpha\beta*-}$	なし
枝刈りの指標	なし	$\alpha_{d,j}, \beta_{d,j}, \alpha_i, \beta_i$	$t_{d,j}, t_{d,j}, t_i, t_i$
枝刈りの方法	なし	$lb_i^{\alpha\beta*-}, ub_i^{\alpha\beta*-}$ を $, \alpha_i, \beta_i$ により制限 (式 (10), (11))	$t_i^{\alpha} < lb_i(d_i) \vee ub_i(d_i) < t_i^{\beta}$ のとき d_i を変更 (図 5 の 36, 38 行)

に含まれる変数には順序は定義されない．エージェント i と関係するエージェントを表すために，次の表記を用いる．i との間に辺が存在するエージェントである近傍エージェントの集合を，Nbr_i により表す．$Chld_i, Dcnd_i$ および $parent_i$ により，疑似木における i の子の集合，子孫の集合および親それぞれを表す．$AnctST_i$ により i の祖先エージェントの部分集合を表す．$AnctST_i$ に含まれるエージェントの変数と，i を根とする部分木に含まれる 1 つ以上のエージェントの変数の間に，制約辺が存在する．疑似木における複数の部分木の間に制約辺は無いため，探索において分割統治法に基づく並列的な処理を部分木ごとに適用できる．

典型的な疑似木は，制約網についての深さ優先探索 (DFS) 木に基づく．DFS 木は根の変数のノードからトップダウンに生成される．しかし，QDCOP の場合は，限量子の順序を考慮する必要があるため，変数の順序を容易に変更することはできない．そこで，限量子の順序を考慮するように変更された疑似木を用いる．すなわち，必要に応じて，親子関係を持つべきノード間に「空」の辺を挿入して疑似木を生成する．このような疑似木を生成する簡単な手法は，次のようなボトムアップな計算により表される．初期状態において，エージェント i は近傍のエージェントの集合 Nbr_i および，それらのエージェントの変数と自変数との順序関係を知る．この順序関係に基づいて，Nbr_i は上位および下位の近傍エージェントの集合 Nbr_i^u および Nbr_i^l に分類される．$Chld_i, Dcnd_i$, $AnctST_i$ および $parent_i$ は次式により再帰的に計算される．

$$Chld_i = \{j | parent_j = i\} \tag{2}$$

$$Dcnd_i = Chld_i \cup \bigcup_{j \in Chld_i} Dcnd_j \tag{3}$$

$$AnctST_i = Nbr_i^u \cup \bigcup_{j \in Chld_i} (AnctST_j \backslash \{i\}) \tag{4}$$

$$parent_i = \begin{cases} \text{lowest } j \in AnctST_i & Nbr_i^l \subseteq Dcnd_i \\ \text{undefined} & \text{otherwise} \end{cases} \tag{5}$$

i が葉ノードの場合は，Nbr_i^l は空集合であるために $AnctST_i$ と $parent_i$ は直ちに計算される．$Chld_i$ はいずれのエージェントの親も決定されていないため空集合であり，したがって，$Dcnd_i$ も空集合である．また，$AnctST_i$ と Nbr_i^u は等しい．ただし，式 (5) における条件 $Nbr_i^l \subseteq Dcnd_i$ はいずれの集合も空集合の場合は真と見なす．この計算はいずれ収束する．i が根ノードの場合は，$AnctST_i$ は空集合であるため，$parent_i$ の最終的な値は「空」であるとみなす．実際の分散アルゴリズムでは，各エージェント i は，$parent_i$ が決定されたときに，$(Chld_i, Dcnd_i, AnctST_i, parent_i)$ を $parent_i$ に送る．このように変更された疑似木を 図 1 (c) に示す．

なお，制約網の辺が，疑似木においてどのように分類されるかという観点からは，次のように考えられる．まず，各エージェント i は，近傍のエージェントの集合 Nbr_i を，上位のエージェントの集合 Nbr_i^u と下位のエージェントの集合 Nbr_i^l に分類する．さらに，上位 Nbr_i^u のエージェントに向かう辺は上位のエージェントに，下位 Nbr_i^l のエージェントに向かう辺は下位のエージェントに，それぞれ分類する．それらのうち，下位への辺のいくつかは木辺となる．また，上位への辺の多くとも一つが木辺となる．他の辺は，後退辺に分類される．さらに，親子関係に対応する辺がない箇所に「空」の辺が挿入される．

3·2 コスト値と解の計算

コスト値は疑似木にもとづいて計算される．この計算は，幾つかのエージェントが最適な目的に対立する選択を行うことを除いて，ADOPT [Modi 05] と同様である．ここでは説明の便宜の為に，各エージェントは計算に必要な他のエージェントの変数値やコスト値を既に受信しているものとする．エージェント i の計算は $AnctST_i$ についての部分解 s_i に基づく．s_i はコンテクストと呼ばれる．$s \approx s'$ により，次式のように定義されるコンテクストの整合性を表す．

$$s \approx s' \triangleq \forall d \, \forall d' \text{ s.t. } (x,d) \in s \wedge (x,d') \in s', d = d' \tag{6}$$

エージェント i における，コスト値の結合の計算は次のように示される．変数 x_i の値 d およびコンテクスト s_i に関する局所コスト $\delta_i(s_i, d)$ は次のように定義される．

$$\delta_i(s_i, d) = \sum_{(x_j, d_j) \in s_i, \, j \in Nbr_i^u} f_{i,j}(d, d_j) \tag{7}$$

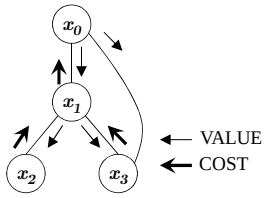


図 2 解法で用いられるメッセージ

変数 x_i の値 d，コンテクスト s_i および x_i を根とする部分木に関する局所最適コストは次のように定義される．

$$g_i^*(s_i) = \begin{cases} \min_{d \in D_i} g_i(s_i, d) & q_i = \exists \\ \max_{d \in D_i} g_i(s_i, d) & q_i = \forall \end{cases} \tag{8}$$

$$g_i(s_i, d) = \delta_i(s_i, d) + \sum_{j \in Chld_i} g_j^*(s_j) \text{ s.t. } (x_i, d) \in s_j, s_j \approx s_i \tag{9}$$

$g_i^*(s_i)$ の計算は変数 x_i の限量子 q_i に依存する．存在限量された変数について，コスト値は従来手法のように最小化される．その一方で，全称限量された変数については，最悪値を計算するために，コスト値は最大化される．

実際の計算では，コスト値の一部は未知である可能性がある．そのような場合は，コストの下限値 0 および上限値 ∞ が既定の値として用いられる．これらの値を用いて計算する場合は，各コスト値は真値の上下界値として表される．疑似木の根の変数 x_r において，大域的最適解 $g_r^*(\varnothing)$ が計算されると，最適解がトップダウンに計算される．

以下では，QDCOP の 3 種類の解法を示す．それぞれの解法の枠組みは類似する一方で，その記述のためにやや多くの異なる表記を要する．そのため，主な要素について，あらかじめ表 1 に対比を示す．これらの詳細については各解法の説明において述べる．

3·3 min-max ADOPT

疑似木に基づく分散探索アルゴリズムは QDCOP に適用するように拡張される．まず，基本的な構成のアルゴリズムとして min-max ADOPT を示す．図 2 に示すように，このアルゴリズムは 2 種類のメッセージを用いる．VALUE メッセージは各エージェントの変数値の割り当てを伝達する．疑似木の木辺からなるトップダウンな配信経路に加えて，後退辺による短絡した経路も用いられる．短絡した経路を用いるメッセージは，送信元のエージェントの変数値の割り当ての情報のみを含む．COST メッセージは木辺に従ってボトムアップにコスト値を伝達する．

解法の概要は次のようである．各エージェントは自身の変数値を VALUE メッセージにより子および下位の近傍エージェントに送信する．VALUE メッセージを受信したエージェントは，コンテクストを更新する．また，各エージェントは自身を根とする部分木以下の局所最適コストを計算し，COST メッセージにより親エージェントに送信する．COST メッセージを受信したエージェントは，現在のコンテクストに対応するコスト値であれば保存し，以降の局所最適コストの計算に用いる．なお，古いコンテクストに対応する子ノードのコスト値は破棄される．局所最適コストは上下界値により表されるが，いずれは，現在の自変数値について収束する．そのとき，エージェントは自変数の値を他の値に変更する．このような処理を反復し，根ノードで大域的最適コストが収束したとき，根ノードは，最適な自変数値を決定し VALUE メッセージにより変数値と終了したことを子ノードに送信する．子ノードは新たな根ノードとして振る舞い，同様に最適な自変数値を決定して終了する．このようにして，いずれは，大域的に最適解が決定する．

各エージェント i は受信した値を一時的に保持する．そのため，以降ではコンテクスト s_i は祖先の変数値の最新の割り当てを表す．ADOPT は memory-bounded なアルゴリズムであり，現在の計算に必要な部分解のみが参照

```
1  main(){
2   initialize();
3   until(forever){
4    until(receive loop is broken){
5     if(¬trm_i){ receive messages; }else{ purge messages; }}
6    if(¬((waiting initial s_i)∨trm_i)){ maintenance(); }}
7  }
8  initialize(){
9   d_i ← d ∈ D_i; s_i ← ∅; trm_i ←false; ptrm_i ←false;
10  foreach d ∈ D_i, j ∈ Chld_i{ (s_{d,j}, lb_{d,j}, ub_{d,j})←(∅, 0, ∞); }
11  if(i is root){ trm_i ←true; Maintenance(); }
12 }
13 when received (VALUE, j, s, ptrm){
14  update s_i using s;
15  ensure child_cost_consistency;
16  if(s is not old){
17   if(j = parent_i){ ptrm_i ← ptrm; }}
18 }
19 when received (COST, j, s, lb, ub){
20  d ← d' s.t. (x_i, d') ∈ s; s ← s\{(x_i, d')};
21  update s_i using s;
22  ensure child_cost_consistency;
23  if(s is not old){
24   update (s_{d,j}, lb_{d,j}, ub_{d,j}) using(s, lb, ub); }
25 }
26 maintenance(){
27  if(ptrm_i ∧ lb_i^* = ub_i^*){
28   if(q_i = ∃){ d_i ← d s.t. ub_i(d) = lb_i^*; }
29   else{ d_i ← d s.t. lb_i(d) = ub_i^*; }
30   trm_i ←true;
31  }else if(lb_i(d_i) = ub_i(d_i)){
32   d_i ← d s.t. lb_i(d) ≠ ub_i(d) if such d exists; }
33  foreach j ∈ Chld_i{
34   send (VALUE, i, s_i ∪ {(x_i, d_i)}, trm_i) to j; }
35  foreach j ∈ Nbr_i^l\Chld_i{
36   send (VALUE, i, {(x_i, d_i)}, null) to j; }
37  send (COST, i, s_i, lb_i^*, ub_i^*) to parent_i;
38 }
```

図 3 min-max ADOPT

される．$lb_{d,j}$ により，$(x_i, d) \in s_j$ であるようなコスト値 $g_j^*(s_j)$ の下界の複製を表す．ここで j はエージェント i の子である．同様に，$ub_{d,j}$ は上界値を表す．また，$lb_i(d)$ と $ub_i(d)$ により，$g_i(s_i, d)$ の上下界を表す．さらに，lb_i^* と ub_i^* により，$g_i^*(s_i)$ の上下界を表す．$lb_{d,j}$ と $ub_{d,j}$ は，コンテクスト $s_{d,j}$ とともにメンテナンスされる．$(s_{d,j}, lb_{d,j}, ub_{d,j})$ それぞれの初期値は，$(\varnothing, 0, \infty)$ である．これらは，受信したコスト値および関連するコンテクストを用いて更新される．$s_i \not\approx s_{d,j}$ であるとき，$(s_{d,j}, lb_{d,j}, ub_{d,j})$ は初期値にリセットされる．

また，ここでは暗黙のうちに，変数値の割り当てに対応する論理時刻ベクトルを用いる．論理時刻は，各変数値の割り当てに関連付られる．割り当てが変更されれば，その論理時刻は増加される．これは，コンテクストを更新する際に最新の変数値を選択するために必要である．変数値の割り当てが変更されれば，そのコスト値は初期値にリセットされる．

このアルゴリズムでは，次の不変条件と操作が用いられる．

child_cost_consistency: $d \in D_i$ および $j \in Chld_i$ それぞれについて，$s_{d,j} \approx s_i$ でなければならない．異なるときは，$(s_{d,j}, lb_{d,j}, ub_{d,j})$ は初期値 $(\varnothing, 0, \infty)$ にリセットされる．

update s_i using s: $(x,d) \in s_i \wedge (x,d') \in s$ なる (x,d) および (x,d') それぞれについて，d' の論理時刻が d よりも新しければ，(x,d) は (x,d') に置き換えられる．$(x,d') \notin s_i \wedge (x,d') \in s$ なる (x,d') それぞれについて，(x,d') は s_i に加えられる．

update ($s_{d,j}$, $lb_{d,j}$, $ub_{d,j}$) using(s, lb, ub): $s_{d,j}$ は s により更新される．$lb > lb_{d,j}$ であれば $lb_{d,j}$ は lb により更新される．$ub < ub_{d,j}$ であれば $ub_{d,j}$ は ub により更新される．

min-max ADOPT の疑似コードを図 3 に示す．ここでは前述の表記に加えて，次の表記が用いられる．d_i は x_i の現在の割り当てを表す．アルゴリズムの終了段階の状態を表すために，$ptrm_i$ と trm_i を用いる．これらは $parent_i$ と i が終了したか否かを表す．

処理の概要は次のようである．処理は根のエージェントからの VALUE メッセージにより開始される (11, 33-36 行)．エージェント i が VALUE メッセージを受信するとそのコンテクスト s_i が更新され (13, 14 行)，$(s_{d,j}, lb_{d,j}, ub_{d,j})$ および $ptrm_i$ がメンテナンスされる (15 行)．これらの値は必要に応じてリセットされる．$(s_{d,j}, lb_{d,j}, ub_{d,j})$ も COST メッセージに従って更新される (19-25 行)．メッセージの受信後に，i は現在の状態の各値を計算する (26-32 行)．$parent_i$ が既に終了していて，$lb_i^* = ub_i^*$ であれば，i は x_i の最適値を選択して終了する (27-30 行)．そうでなければ i は残りの探索空間を探索するために，必要に応じて x_i の値を変更する (31, 32 行)．変更された値は，親および子エージェントへ COST および VALUE メッセージにより送信される (33-37 行)．また，VALUE メッセージは子エージェント以外の下位の近傍エージェントにも送信される (36 行)．ただし，この VALUE メッセージの役割は，コンテクストを短絡して伝達することのみである．したがって，終了状態を表す trm_i は子エージェントのみに伝達するため，他のエージェントには代わりに空の値 (null) を送る．

最終的に，すべてのエージェントが停止しシステムは静止する．なお，全称限量された変数を持つエージェントが解を選択するときには，他の基準も考えられるが，本手法では大域的最小コスト値の上界を与える解を選択する．

このアルゴリズムは従来の ADOPT を簡略したものに基づいている．そのため，より効果的な枝刈りや探索の効率化のための手法を適用する余地がある．たとえば，同一内容のメッセージが続けて繰り返し送信される場合は，後続のメッセージの送信は省略できる．また，特に上述のように min-max 法を用いる場合は，各エージェントは $AnctST_i$ についての割り当てのみに依存する．そのため，VALUE メッセージに含まれるコンテクストを依存する割り当てのみに削減できる．これにより変数値の変更が抑制され，冗長な探索が削減される．後述の評価における実装では，重複する内容の連続したメッセージの送信と，VALE メッセージのコンテクストの一部を削減した．

先述した変数値の割り当てに対応する論理時刻ベクトルは，非同期処理の影響を除くために用いられる．本研究の手法では，あるエージェントの同一の変数値の割り当てに対して，コスト値の上下界が子のエージェントから繰り返し報告される．そのコスト値の境界は初期状態では 0 と ∞ であり，次第に狭くなる．その境界は，祖先または親の変数値が変更されたときに，初期状態に再設定される．

このとき，理想的には，常に現在の変数値についての上下界値が子ノードから報告され，それらは必ず狭くなってゆくべきである．しかし，子孫を含めた複数ノード間の通信の遅延の状況および，探索戦略によっては，以前の変数値の割り当てに対するコストの境界値が中途半端に計算される可能性がある．この上下界値が既に計算されたものよりも広いとき，その値を採用してしまうと境界値の収束が保証されない．

このような可能性がないことを保証するためには，個々の探索手法の性質を考慮して境界値を更新する規則 [松井 05, Yeoh 08, Silaghi 09] が必要である．その一方で，min-max 法や後述の alpha-beta 法に基づく方法では，境界の扱いが従来解法よりも複雑である．そこで，本来の探索の議論に集中するために，文献 [Silaghi 09] でも用いられた汎用的な論理時刻を用いる．ただし，この方法は冗長な探索を増加させる面があると考えられる．論理時刻を用いない安全かつ効率的な規則は今後の課題とし

```
1  initialize(){
2   d_i ← d ∈ D_i; s_i ← ∅; rlvl_i ← 0; trm_i ←false;
3   ptrm_i ←false; (s_i^{αβ}, rlvl_i^{αβ}, α_i, β_i)←(s_i, rlvl_i, 0, ∞);
4   foreach d ∈ D_i, j ∈ Chld_i{
5    (s_{d,j}, rlvl_{d,j}, lb_{d,j}, ub_{d,j}, lb_{d,j}^{αβ}, ub_{d,j}^{αβ}, α_{d,j}, β_{d,j})
6     ←(s_i, rlvl_i, 0, ∞, 0, ∞, 0, ∞); }
7   if(i is root){ trm_i ←true; maintenance(); }
8  }
9  when received (VALUE, j, s, rlvl, α, β, ptrm){
10  update (s_i, rlvl_i) using (s, rlvl);
11  ensure alpha_beta_consistency and child_cost_consistency;
12  ensure child_alpha_beta_invariant;
13  if(s and rlvl are not old){
14   if(j = parent_i){
15    update (s_i^{αβ}, rlvl_i^{αβ}, α_i, β_i) using (s, rlvl, α, β);
16    ensure child_alpha_beta_invariant; ptrm_i ← ptrm; }}
17 }
18 when received (COST, j, s, rlvl, lb, ub, lb^{αβ}, ub^{αβ}){
19  d ← d' s.t. (x_i, d') ∈ s; s ← s\{(x_i, d')};
20  update (s_i, rlvl_i) using (s, rlvl);
21  ensure alpha_beta_consistency and child_cost_consistency;
22  ensure child_alpha_beta_invariant;
23  if(s and rlvl are not old){
24   update (s_{d,j}, rlvl_{d,j}, lb_{d,j}, ub_{d,j}, lb_{d,j}^{αβ}, ub_{d,j}^{αβ}) using
25    (s, rlvl, lb, ub, lb^{αβ}, ub^{αβ}); }
26 }
27 maintenance(){
28  ensure alpha_beta_invariant;
29  ensure child_alpha_beta_invariant;
30  if(ptrm_i ∧ q_i = ∃ ∧ ub_i^* = α_i){
31   d_i ← d s.t. ub_i(d) = α_i; trm_i ←true;
32  }else if(ptrm_i ∧ q_i = ∀ ∧ lb_i^* = β_i){
33   d_i ← d s.t. lb_i(d) = β_i; trm_i ←true;
34  }else if(ptrm_i ∧ rlvl_i < i){
35    α' ← α_i; β' ← β_i; rlvl_i ← i;
36    ensure alpha_beta_consistency;
37    if(q_i = ∃){ α_i ← α'; }else{ β_i ← β'; }
38    ensure child_cost_consistency;
39    ensure alpha_beta_invariant;
40    ensure child_alpha_beta_invariant;
41  }else if(lb_i^{αβ*−} ≠ ub_i^{αβ*−} ∧ lb_i^{αβ}(d_i) = ub_i^{αβ}(d_i)){
42   d_i ← d s.t. lb_i^{αβ}(d) ≠ ub_i^{αβ}(d) if such d exists; }
43  foreach j ∈ Chld_i{
44   send (VALUE, i, s_i ∪ {(x_i, d_i)}, rlvl_i, α_{d_i,j}, β_{d_i,j}, trm_i)
45    to j; }
46  foreach j ∈ Nbr_i^l\Chld_i{
47   send (VALUE, i, {(x_i, d_i)}, rlvl_i, null, null, null) to j; }
48  send (COST, i, s_i, rlvl_i, lb_i^*, ub_i^*, lb_i^{αβ*−}, ub_i^{αβ*−})
49   to parent_i;
50 }
```

図 4 alpha-beta ADOPT

たい.

3·4 alpha-beta ADOPT

alpha-beta 法は，ゲーム木探索における基本的な枝刈り手法である [Knuth 75, Russell 03]．この手法は二種類の評価値の境界の値 alpha，beta を用いる．それらは可能なコスト値の上下界を表す．alpha は最大化プレーヤにより管理される下界値であり，beta は最小化プレーヤにより管理される上界値である．いずれのプレーヤも自身の種類に対応する境界値のみを変更できる．QDCOP では，各エージェントは限量子の種類に応じていずれかの種類のプレーヤとして動作する．

枝刈はトップダウンに行われる．あるエージェントが現在の部分解のコスト値を報告したとき，その親のエージェントは境界値 alpha または beta を狭めようとする．alpha または beta の新たな値は子のエージェントの探索における枝刈りに用いられる．alpha および beta は，ADOPT における *backtracking threshold* [Modi 05] に類似する．このような枝刈を min-max ADOPT に適用し alpha-beta ADOPT を提案する．

解法は min-max ADOPT に次のような処理を加えたものとなる．各エージェントは子エージェントへの VALUE メッセージにより，境界値 alpha,beta を送信する．これらの値を，子エージェントに適切に配分するために child_alpha_beta_invariant と呼ばれる不変条件を維持する．子エージェントはこれらの値を自身のコスト値の境界を制限する値として用いる．各エージェントは局所最適コストを計算するが，それらの値は，alpha,beta 値により制限される．局所最適コスト値は COST メッセージにより親エージェントに送信される．また，エージェントの限量記号に応じて，局所最適コストに基づき，alpha または beta 値を更新する．このとき，alpha_beta_invariant と呼ばれる不変条件を維持する．

alpha-beta ADOPT では, エージェント i は alpha-beta 法のために次のような値を用いる．α_i と β_i により i の alpha および beta 値を表す. これらは枝刈りに用いられる.

alpha-beta 法は効果的な枝刈りが可能である代わりに真の境界を超えるコスト値を計算する可能性がある．例えば，疑似木において根のエージェントが大域的に最適なコスト値を得た時，その値は正しい．しかし，そのエージェントの変数値のうち，大域的な最適解を構成しない変数値について評価されるコスト値も，大域的に最適なコスト値と等しい可能性がある．このような場合は，そのままでは最適解を決定できない.

そこで，コスト値の評価を容易にするために，ここでは真の境界とは別の境界を導入する．$lb_{d,j}^{\alpha\beta}$, $ub_{d,j}^{\alpha\beta}$, $lb_i^{\alpha\beta}(d)$, $ub_j^{\alpha\beta}(d)$, $lb_i^{\alpha\beta*}$ および $ub_j^{\alpha\beta*}$ は，それぞれ $lb_{d,j}$, $ub_{d,j}$, $lb_i(d)$, $ub_i(d)$, lb_i^* および ub_i^* と同様に定義される．さらに，$lb_i^{\alpha\beta*}$ と $ub_i^{\alpha\beta*}$ を制限する値として，$lb_i^{\alpha\beta*-}$ および $ub_i^{\alpha\beta*-}$ を導入する．これらは次のように定義される.

$$lb_i^{\alpha\beta*-} = \min(\max(lb_i^{\alpha\beta*}, \alpha_i), \beta_i) \tag{10}$$

$$ub_i^{\alpha\beta*-} = \min(\max(ub_i^{\alpha\beta*}, \alpha_i), \beta_i) \tag{11}$$

alpha-beta ADOPT では，部分木のコスト値の収束条件に $lb_i^{\alpha\beta*-}$ と $ub_i^{\alpha\beta*-}$ を用いることにより，枝刈りが適用される．alpha および beta 値は i とその子 j に共有される．$\alpha_{d,j}$, $\beta_{d,j}$ により $x_i = d$ の場合に j が知る alpha, beta 値を表す.

各変数値についての暗黙な論理時刻に加えて，大域的な論理時刻 *root level* を導入する．この論理時刻は，最適解を決定する段階において，探索がリセットされたか否

かを判別するために用いる．$rlvl_i$ により i の知る最新の root level を表す．コンテクスト s_i と root level $rlvl_i$ の整合性は，alpha, beta およびコスト値を計算するために必要である．α_i と β_i は関連するコンテクスト $s_i^{\alpha\beta}$ および $rlvl_i^{\alpha\beta}$ とともに維持される．$lb_{d,j}$, $ub_{d,j}$, $lb_{d,j}^{\alpha\beta}$, $ub_{d,j}^{\alpha\beta}$, $\alpha_{d,j}$ および $\beta_{d,j}$ は，関連するコンテクスト $s_{d,j}$ および $rlvl_{d,j}$ とともに維持される．

このアルゴリズムでは，次の不変条件と操作が用いられる

alpha_beta_consistency: $s_i^{\alpha\beta}$ と $rlvl_i^{\alpha\beta}$ はそれぞれ s_i と $rlvl_i$ と等しくなければならない．そうでなければ，($s_i^{\alpha\beta}$, $rlvl_i^{\alpha\beta}$, α_i, β_i) は初期値 (s_i, $rlvl_i$, 0, ∞) にリセットされる．

child_cost_consistency: 各 $d \in D_i$ および $j \in Chld_i$ について，$s_{d,j}$ と $rlvl_{d,j}$ はそれぞれ s_i および $rlvl_i$ に等しくなければならない．そうでなければ，($s_{d,j}$, $rlvl_{d,j}$, $lb_{d,j}$, $ub_{d,j}$, $lb_{d,j}^{\alpha\beta}$, $ub_{d,j}^{\alpha\beta}$, $\alpha_{d,j}$, $\beta_{d,j}$) は初期値 (s_i, $rlvl_i$, 0, ∞, 0, ∞, 0, ∞) にリセットされる．

alpha_beta_invariant: α_i と β_i はコスト値の最良の上下界を超えてはならない．これは次のようになされる．まず，$lb_i^{\alpha\beta*}$ と $ub_i^{\alpha\beta*}$ が，式 (8) と (9) に従って計算される．次に，$lb_i^{\alpha\beta*-}$ と $ub_i^{\alpha\beta*-}$ が式 (10) と (11) に従って計算される．もし，エージェント i が存在限量された変数を持つならば，β_i が $ub_i^{\alpha\beta*-}$ により更新され，そうでなければ，α_i が $lb_i^{\alpha\beta*-}$ により更新される．さらに根のエージェントでは，$lb_i^{\alpha\beta*-} = ub_i^{\alpha\beta*-}$ であるとき，自身が変更する α_i または β_i の逆側の境界を $\alpha_i = \beta_i$ となるように変更する．これにより，根ノードの他の処理を一般のノードと同様に構成できる．

child_alpha_beta_invariant: α_i と β_i はエージェント i とその子と共有される．もし i が 1 つの子 j のみを持つとき，$\alpha_{d,j}$ および $\beta_{d,j}$ は各 $d \in D_i$ について，$\max(0, \alpha_i - \delta_i(s_i, d))$ および $\max(0, \beta_i - \delta_i(s_i, d))$ とする．そうでなければ，$\alpha_{d,j}$ および $\beta_{d,j}$ は各 $d \in D_i$ および $j \in Chld_i$ について 0 および $\max(0, \beta_i - \delta_i(s_i, d))$ とする．ここで 0 は下限値である．このアルゴリズムでは子は与えられた境界値を広げられないため，子ノードが複数の場合には安全策として最も広い境界を用いる．さらに，$lb_{d,j}^{\alpha\beta}$ と $ub_{d,j}^{\alpha\beta}$ は $\min(\max(lb_{d,j}^{\alpha\beta}, \alpha_{d,j}), \beta_{d,j})$ および $\min(\max(ub_{d,j}^{\alpha\beta}, \alpha_{d,j}), \beta_{d,j})$. のように制限される

update (s_i, $rlvl_i$) using (s, $rlvl$): s_i は s を用いて min-max ADOPT と同様に更新される．$rlvl > rlvl_i$ であれば $rlvl_i$ は $rlvl$. により更新される．

update ($s_i^{\alpha\beta}$, $rlvl_i^{\alpha\beta}$, α_i, β_i) using (s, $rlvl$, α, β): $s_i^{\alpha\beta}$ と $rlvl_i^{\alpha\beta}$ はそれぞれ s と $rlvl$ により更新される．エージェント i が存在限量された変数を持つとき，α_i と β_i はそれぞれ α と $\max(\min(\beta_i, \beta), \alpha)$ により更新される．そうでなければ，α_i と β_i はそれぞれ $\min(\max(\alpha_i, \alpha), \beta)$ と β により更新される．

update ($s_{d,j}$, $rlvl_{d,j}$, $lb_{d,j}$, $ub_{d,j}$, $lb_{d,j}^{\alpha\beta}$, $ub_{d,j}^{\alpha\beta}$) using (s, $rlvl$, lb, ub, $lb^{\alpha\beta}$, $ub^{\alpha\beta}$): $s_{d,j}$ と $rlvl_{d,j}$ はそれぞれ s と $rlvl$ により更新される．$lb_{d,j}$ と $ub_{d,j}$ は lb と ub を用いて min-max ADOPT 同様に更新される．$lb_{d,j}^{\alpha\beta}$ と $ub_{d,j}^{\alpha\beta}$ は次のように更新される．まず，$lb^{\alpha\beta}$ と $ub^{\alpha\beta}$ はそれぞれ $\min(\max(lb^{\alpha\beta}, \alpha_{d,j}), \beta_{d,j})$ と $\min(\max(ub^{\alpha\beta}, \alpha_{d,j}), \beta_{d,j})$ のように更新される．これらは child_alpha_beta_invariant のために必要である．そして，$lb^{\alpha\beta} > lb_{d,j}^{\alpha\beta}$ であれば，$lb_{d,j}^{\alpha\beta}$ は $lb^{\alpha\beta}$ により更新される．$ub^{\alpha\beta} < ub_{d,j}^{\alpha\beta}$ であれば，$ub_{d,j}^{\alpha\beta}$ は $ub^{\alpha\beta}$ により更新される．

alpha-beta 法では，各エージェントに与えられる境界値は祖先の部分解に依存する．そのため，alpha_beta_consistency および child_cost_consistency において，s_i, $s_i^{\alpha\beta}$ および $s_{d,j}$ は等しくなるように整合性を強制される．child_alpha_beta_invariant では，エージェント i は元の $lb_{d,j}^{\alpha\beta}$ と $ub_{d,j}^{\alpha\beta}$ を超える $\alpha_{d,j}$ と $\beta_{d,j}$ を割り当てるかもしれない．このような場合は実行不可能であるため，例外的に $lb_{d,j}^{\alpha\beta}$ と $ub_{d,j}^{\alpha\beta}$ は等しい値をとるように強制され，探索は枝刈りされる．

このアルゴリズムの疑似コードを図 4 に示す．主手続きは min-max ADOPT と同様である．また，基本的な処理は枝刈り以外は min-max ADOPT と同様である．

VALUE メッセージは，α および β を含むように拡張される．また，COST メッセージは，alpha-beta 法で集計されるコスト値の境界 $lb^{\alpha\beta}$ および $ub^{\alpha\beta}$ を含む．min-max ADOPT と同様に，VALUE メッセージは，子エージェント以外の下位の近傍エージェントにも送信される (47 行)．ただし，α および β は子のみに伝達するため，他のエージェントには代わりに空の値 (null) を送る．

根ノードのエージェント r では，$\alpha_r = \beta_r \wedge (ub_r^* = \alpha_r \vee lb_r^* = \beta_r)$ がいずれ成立する．このとき，α_r と β_r は alpha_beta_invariant により同じ値となる．このように境界が閉じた後，根ノードは最適もしくは最悪の変数値を選択し，VALUE メッセージにより自身の終了を送信する．この VALUE メッセージは $\alpha = \beta$ なる α と β を含む．i の親が終了しているとき，i は自身におけるコストの境界値が閉じていれば，同様に終了できる．そうでなければ，i における境界を閉じるためにさらに探索が必要である．しかし，このとき既に $\alpha_i = \beta_i$ であり，それ以上の探索が行われないため，探索をリセットする必要がある．

そこで，エージェント i の親が終了しているが i の境界は閉じていないとき，i は *root level* $rlvl_i$ を 1 増加し，探索をリセットする．$rlvl_i$ の値は，メッセージにより i の子孫に伝搬される．このリセットにより i は新たな根ノードとなり，i の祖先についての部分解は固定される．いずれ i におけるコストの境界は収束し，i は終了する．このような探索のリセットは非効率的であるが，探索空間を削減する余地はある．すなわち，完全なリセットの

かわりに，i によって変更されない側の境界値 alpha または beta は i において保持できる．この境界値は i の親によって最適値に設定されている．

この解法は，分枝限定法と動的計画法にもとづく ADOPT 型の解法に，alpha-beta 法の要素を組み込むことを意図している．そのため，本来はコストの境界値の計算に矛盾のない ADOPT に，境界値の非単調性や親子間の不整合を吸収させるように各規則を調整した．これらの規則には，alpha-beta 法と厳密に対応すると解釈できない部分があると考えられる．

特に，alpha-beta カットに対応する操作が明確ではないが，枝刈りに関しては，図 4 の 41 行において，式 (10) と (11) のように α 値と β 値で制限された $lb_i^{\alpha\beta*-}$ と $ub_i^{\alpha\beta*-}$ が収束することを条件としている．また，これらの値が親にコスト値として報告される．そのため，この箇所にカットに相当する計算が含まれると考えられる．また，child_alpha_beta_invaliant においては，子ノードから報告されたコスト値 $lb_{d,j}^{\alpha\beta}$ と $ub_{d,j}^{\alpha\beta}$ が，子ノードに割り当てる α,β 値である $\alpha_{d,j}$,$\beta_{d,j}$ により制限される．これらの個所において，α または β が他方を超える状況における枝刈りが機能している．

min-max ADOPT と異なり，alpha-beta ADOPT のような枝刈りを行う場合は，疑似木に従ってトップダウンに伝搬されるコスト値 α_i および β_i を用いる．このコスト値は，全ての祖先エージェントの変数値に依存する．

局所コスト値の計算と，枝刈りのためのトップダウンなコスト値の計算の双方における，状況の変化を判定するためには，異なる 2 つのコンテクストが必要である．前者は，あるエージェント i においては，$AncstST_i$ の変数値のみに依存する．後者は全ての祖先エージェントの変数値に依存する．従って，前者は後者に含まれるため，後者を共用する．

局所コスト値の計算に用いられる変数の割り当ては，min-max ADOPT でも alpha-beta ADOPT でも同様であることに注意されたい，両者は枝刈りの有無のみが異なる．

alpha-beta ADOPT は，分岐数が複数の場合には，過剰な枝刈りを防ぐために，child_alpha_beta_invariant において，親から子に与える alpha と beta の範囲を最も広く取る必要がある．従って，あるエージェントについて分岐数が増加すると，各子エージェント以下の部分木に対する枝刈の効果が相対的に低下する可能性があると考えられる．

その一方で，エージェント数が同一で分岐数が大きければ，疑似木の深さが小さいことになる．木の深さは探索の反復回数を増加するため，分岐数と木の深さの影響にトレードオフが存在しうると考えられる．

3·5 bi-threshold ADOPT

alpha-beta 法に類似する手法として，ADOPT [Modi 05] における境界値 *backtracking threshold* を 2 つ用いるように拡張し，QDCOP の解法 bi-threshold ADOPT を構成できる．backtracking threshold は alpha-beta 法における alpha に類似する．しかし，alpha-beta 法は最適ではない解について過剰なコスト値を計算することに対し，backtracking threshold はすべての解について真のコスト値を超えない．元の ADOPT では，単一の backtracking threshold が主にコストの下界値により更新される．本節の手法では，さらに主にコストの上界値により更新される backtracking threshold を用いる．各エージェント i は backtracking threshold のために次の値を用いる．t_i^α および t_i^β は i の backtracking threshold を表す．$t_{d,j}^\alpha$ および $t_{d,j}^\beta$ は $j \in Chld_i$ および $d \in D_i$ についての backtracking thresholds を表す．これらは i が j に対して割り当てる．また，コストの計算のために，$lb_{d,j}, ub_{d,j}, lb_i(d), ub_i(d)$, lb_i^* および ub_i^* を用いる. bi-threshold ADOPT の基本的なアイデアは $lb_i^* \leq t_i^\alpha \leq t_i^\beta \leq ub_i^*$ を維持することである．t_i^α および t_i^β は関連するコンテクスト $s_i^{t\alpha\beta}$ および root level $rlvl_i^{t\alpha\beta}$ とともに更新される．また，$lb_{d,j}, ub_{d,j}, t_{d,j}^\alpha$ および $t_{d,j}^\beta$ は関連する $s_{d,j}$ と $rlvl_{d,j}$ とともに更新される．アルゴリズムで用いる不変条件と操作は次のようである．

threshold_consistency: $s_i^{t\alpha\beta}$ と $rlvl_i^{t\alpha\beta}$ はそれぞれ s_i と $rlvl_i$ と等しくなければならない．もし異なる場合は，$(s_i^{t\alpha\beta}, rlvl_i^{t\alpha\beta}, t_i^\alpha, t_i^\beta)$ を初期値 $(s_i, rlvl_i, 0, \infty)$ にリセットする．

child_cost_consistency: $d \in D_i$ と $j \in Chld_i$ それぞれについて，$s_{d,j}$ と $rlvl_{d,j}$ はそれぞれ s_i と $rlvl_i$ に等しくなければならない．もし異なる場合は，$(s_{d,j}, rlvl_{d,j}, lb_{d,j}, ub_{d,j}, t_{d,j}^\alpha, t_{d,j}^\beta)$ を初期値 $(s_i, rlvl_i, 0, \infty, 0, \infty)$ にリセットする．

threshold_invariant: $lb_i^* \leq t_i^\alpha \leq t_i^\beta \leq ub_i^*$ は維持されなければならない．t_i^α と t_i^β はそれぞれ $\min(\max(t_i^\alpha, lb_i^*), ub_i^*)$ と $\min(\max(t_i^\beta, lb_i^*), ub_i^*)$ により更新される．

child_threshold_invariant: $d \in D_i$ と $j \in Chld_i$ それぞれについて，$lb_{d,j} \leq t_{d,j}^\alpha \leq t_{d,j}^\beta \leq ub_{d,j}$ は維持されなければならない．$t_{d,j}^\alpha$ と $t_{d,j}^\beta$ はそれぞれ $\min(\max(t_{d,j}^\alpha, lb_{d,j}), ub_{d,j})$ と $\min(\max(t_{d,j}^\beta, lb_{d,j}), ub_{d,j})$ により更新される．

child_allocation_invariant: t_i^α と t_i^β はエージェント i とその子で共有される．d_i と $j \in Chld_i$ それぞれについて，$t_{d_i,j}^\alpha$ と $t_{d_i,j}^\beta$ は $t_i^\alpha = \delta_i(d_i) + \sum_{j \in Chld_i} t_{d_i,j}^\alpha$ と $t_i^\beta = \delta_i(d_i) + \sum_{j \in Chld_i} t_{d_i,j}^\beta$ を維持しなければならない．この関係を満たさないときは，関係を満たすようにいくつかの $t_{d_i,j}^\alpha$ および $t_{d_i,j}^\beta$ を増減する．このとき，child_threshold_invariant も満たすようにする．

update (s_i, $rlvl_i$) using (s, $rlvl$): alpha-beta ADOPT と同様である．

```
1  initialize(){
2   d_i ← d ∈ D_i; s_i ← ∅; rlvl_i ← 0; trm_i ←false;
3   ptrm_i ←false; (s_i^{tαβ}, rlvl_i^{tαβ}, t_i^α, t_i^β)←(s_i, rlvl_i, 0, ∞);
4   foreach d ∈ D_i, j ∈ Chld_i{
5    (s_{d,j}, rlvl_{d,j}, lb_{d,j}, ub_{d,j}, t_{d,j}^α, t_{d,j}^β)
6     ←(s_i, rlvl_i, 0, ∞, 0, ∞); }
7   if(i is root){ trm_i ←true; maintenance(); }
8  }
9  when received(VALUE, j, s, rlvl, t^α, t^β, ptrm){
10  update (s_i, rlvl_i) using (s, rlvl);
11  ensure threshold_consistency and child_cost_consistency;
12  if(s and rlvl are not old){
13   if(j = parent_i){
14    update (s_i^{tαβ}, rlvl_i^{tαβ}, t_i^α, t_i^β) using (s, rlvl, t^α, t^β);
15    ptrm_i ← ptrm; }}
16 }
17 when received(COST, j, s, rlvl, lb, ub){
18  d ← d' s.t. (x_i, d') ∈ s; s ← s\{(x_i, d')};
19  update (s_i, rlvl_i) using (s, rlvl);
20  ensure threshold_consistency and child_cost_consistency;
21  if(s and rlvl are not old){
22   update (s_{d,j}, rlvl_{d,j}, lb_{d,j}, ub_{d,j}) using (s, rlvl, lb, ub); }
23 }
24 maintenance(){
25  ensure threshold_invariant;
26  if(ptrm_i ∧ q_i = ∃ ∧ ub_i^* = t_i^α){
27   d_i ← d s.t. ub_i(d) = t_i^α; trm_i ←true;
28  }else if(ptrm_i ∧ q_i = ∀ ∧ lb_i^* = t_i^β){
29   d_i ← d s.t. lb_i(d) = t_i^β; trm_i ←true;
30  }else if(ptrm_i ∧ rlvl_i < i){
31    t^{α'} ← t_i^α; t^{β'} ← t_i^β; rlvl_i ← i;
32    ensure threshold_consistency;
33    if(q_i = ∃){ t_i^α ← t^{α'}; }else{ t_i^β ← t^{β'}; }
34    ensure child_cost_consistency;
35    ensure threshold_invariant;
36    if(t_i^α < lb_i(d_i) ∨ ub_i(d_i) < t_i^β){
37     d_i ← d s.t. ¬(t_i^α < lb_i(d) ∨ ub_i(d) < t_i^β); }
38  }else if(t_i^α < lb_i(d_i) ∨ ub_i(d_i) < t_i^β){
39   d_i ← d s.t. ¬(t_i^α < lb_i(d) ∨ ub_i(d) < t_i^β); }
40  ensure child_threshold_invariant and child_allcation_invariant;
41  foreach j ∈ Chld_i{
42   send (VALUE, i, s_i ∪ {(x_i, d_i)}, rlvl_i, t_{d_i,j}^α, t_{d_i,j}^β, trm_i)
43    to j; }
44  foreach j ∈ Nbr_i^l\Chld_i{
45   send (VALUE, i, {(x_i, d_i)}, rlvl_i, null, null, null) to j; }
46  send (COST, i, s_i, rlvl_i, lb_i^*, ub_i^*) to parent_i;
47 }
```

図 5 bi-threshold ADOPT

update ($s_i^{t\alpha\beta}$, $rlvl_i^{t\alpha\beta}$, t_i^α, t_i^β) **using** (s, $rlvl$, t^α, t^β)**:** $s_i^{t\alpha\beta}$ と $rlvl_i^{t\alpha\beta}$ はそれぞれ s と $rlvl$ により更新される．t_i^α と t_i^β はそれぞれ t^α と t^β により更新される．t^α または t^β が lb_i^* または ub_i^* を超えるとき，それらは threshold_invariant により修正される．

update ($s_{d,j}$, $rlvl_{d,j}$, $lb_{d,j}$, $ub_{d,j}$) **using** (s, $rlvl$, lb, ub)**:** $s_{d,j}$ と $rlvl_{d,j}$ はそれぞれ s と $rlvl$ により更新される．$lb_{d,j}$ と $ub_{d,j}$ は lb と ub を用いて min-max ADOPT と同様に更新される．

このアルゴリズムの疑似コードを図 5 に示す．主手続は min-max ADOPT と同様である．基本的な計算は，枝刈りとコストの境界値の計算を除いて，min-max/alpha-beta ADOPT と同様である．枝刈りは，図中 36, 38 行の $t_i^\alpha < lb_i(d_i) \vee ub_i(d_i) < t_i^\beta$ に基づいて d_i の値を変更することにより行われる．

bi-threshold ADOPT においても，終了処理で既存の境界値を用いると，alpha-beta ADOPT 同様に計算が収束しない可能性がある．そのため，終了段階の処理においては alpha-beta ADOPT 同様に探索をリセットする．

3·6 アルゴリズムの正しさ

Min-max/bi-threshold ADOPT は従来の ADOPT の自然な簡略または拡張であり，コスト値の境界の計算が QD-COP に一般化されたことを除いて大きな変更はない．したがって，計算の正当性は従来手法と同様である．そこで，ここでは特に alpha-beta ADOPT 法の正当性について考察する．3·4 節に示したように，alpha-beta 法ではコスト値の境界 alpha および beta は疑似木においてトップダウンに強制される．これらの境界は，各部分木における最適コストの真の境界を超える可能性がある．疑似木の各ノードにおいて，alpha と beta は現在の最良の境界を表すように維持され，子ノードを根とする部分木に適用される．この境界値が部分木において実行不能であれば，トップダウンに強制される形で枝刈りが行われる．その結果として，大域的に最適ではない解のコスト値について，真の境界値は保証されない．

そこで本研究では，alpha-beta ADOPT の alpha，beta 値およびコストの上下界値の計算と並行して，真のコストの境界値も計算することとした．この真の境界値は alpha, beta 値の間隔を狭める要因なので，根のエージェントで最適コストが収束したとき少なくとも一つの真の境界値が alpha または beta 値に一致する．この境界値に対応する変数値が最適解の一部となる．したがって，alpha-beta ADOPT は最適解を決定することができる．

3·7 境界付誤差

提案手法では従来の ADOPT [Modi 05] と同様に，根のエージェントでコスト値の上下界の差が閾値以内となった段階で探索を終了し，準最適解を決定できる．許容される誤差のパラメータを e とするとき，終了条件は $lb_i^* + e \geq ub_i^*$ に置き換えられる．また，存在限量されたエージェントは $ub_i(d) = ub_i^*$，全称限量されたエージェントは $lb_i(d) = lb_i^*$ の条件を満たすように，それぞれ最適な変数値 d を決定する．さらに，存在限量されたエージェントは $\alpha_i = \beta_i = ub_i^*$(または $t_i^\alpha = t_i^\beta = ub_i^*$)，全称限量されたエージェントは $\alpha_i = \beta_i = lb_i^*$(または $t_i^\alpha = t_i^\beta = lb_i^*$)の条件を満たすように，それぞれ，$\alpha_i, \beta_i, t_i^\alpha, t_i^\beta$ の値を決定する．これは，lb_i^* と α_i(または t_i^α)，および ub_i^* と β_i(または t_i^β) の間に最大で e の余裕を挿入することに相当する．根以外のエージェントの処理に変更は無い．

3·8 コスト値が対象な問題への適用

上記で示したアルゴリズムは，コスト値の下限が 0 である問題を解くためのものであった．これを拡張して下限値を $-\infty$ に一般化できる．ただし，負のコスト値を扱うために，alpha-beta ADOPT における，child_alpha_beta_invariant の修正が必要である．この修正により，エージェントは複数の子ノードを持つときに，各子 j の $\beta_{d,j}$ に ∞ を割り当てる．コスト値の合計の計算は非単調となるが，コスト値の下限値が 0 であることに基づく枝刈りが行われなくなるため，各アルゴリズムは正しく動作する．

4. 評　　価

提案手法の効果を実験により評価した．以下では各手法の比較と考察を示す．

4·1 評価用の問題の設定

QDCOP とその解法を評価する上では，従来の DCOP のパラメータに加えて限量子の詳細な配置などの多数のパラメータの影響が考えられるが，本研究では初期の結果として，特に探索の反復処理数に関するいくつかの特徴的な結果を示す．ここでは次の例題に提案手法を適用した．

max-CSP: 最大制約充足問題を意図する例題である. 各コスト関数は変数値の組ごとに 0 か 1 の値をとる．コスト値が 1 となる組の割合 t を，$\frac{1}{3}$ または $\frac{5}{9}$ とした．

COP: 一般的な DCOP の例題である．変数値の組のコスト値は一様分布に従い，0 から 10 の間の整数値をとる．

各問題は n 個の 3 値変数と，コスト値を表す $l \times n$ 個の 2 項関数からなる．ここで l は制約の密度をあらわすパラメータである．本研究では，可能な実験時間でアルゴリズムの特徴を描くために，比較的制約密度の低い場合である $l = 2$ を用いた．全ての変数に対する，全称限量された変数の比率をパラメータ u により設定した．限量子の配置はランダムに決定した．各実験結果は 50 個の例題について平均した．実験結果では，各アルゴリズムは次のように表記される．**min-max:** min-max ADOPT. **al-bt:** alpha-beta ADOPT. **bi-thr:** bi-threshold ADOPT. 実験ではメッセージサイクルを反復するシミュレーションを行った．各メッセージサイクルでは，エージェントは受信キューにある各メッセージを読み込んで処理し，送信キューにメッセージを書き込む．各エージェントの送信キューにあるメッセージは，各メッセージサイクルの最後にエージェント間で交換される．メッセージサイクルは 10^7 を上限とし，上限値を超えたときは実験を強制的に終了した．このとき，その実験について要したメッセージサイクル数として上限値を用いた．

4·2 結　　果

解法の終了までに要したメッセージサイクル数を図 6(a), (b) および (c) に示す．max-CSP, $n = 15$ の場合は，u=1 の場合の各 1 個の例題における al-bt, bi-thr 以外は，メッセージサイクル数の上限以内に終了した．問題の規模を増加した $n = 20$ の場合は，メッセージサイクル数の上限に達した例題の影響をより多く受けているが，グラフの傾向は $n = 15$ の場合と同様である．max CSP の $t = \frac{1}{3}$ では，min-max のメッセージサイクル数が全称限量された変数の割合に従って増加している．これらの問題のコスト関数では，コスト値 0 と 1 の比は均等ではない．さらに，解法ではコスト値の下限値を 0 として枝刈を行うため，コスト値の最小化は最大化よりも容易であると考えられる．min-max 以外の各手法の結果も同様の傾向を示している．なお，min-max の実装においては，3·3 節で述べた冗長なコンテクストを削減するようにした．そのため，枝刈の効果がない場合には，全ての祖先についてのコンテクストを用いる al-bt と bi-thr は，通信範囲と遅延の増加のために min-max よりも多くのメッセージサイクル数を要する場合がある．$u = 1$ の場合は，このような状況と考えられる．$0 \leq u \leq 0.75$ の場合は，al-bt と bi-thr は min-max よりも効果的である．

bi-thr は al-bt よりも若干メッセージサイクル数が少ない．ただしこれらの差異から直ちに解法の優劣を判断することはできないと考えられる．そこで，ここでは差異の原因となりうる各手法の性質についての考察にとどめる．両者の主な違いは探索戦略と，親子エージェント間でのコストの境界値の自由さにあると考えられる．端的にいえば，探索戦略は al-bt は深さ優先であり，bi-thr は楽観的である．また，コストの境界値の自由さは，bi-thr の方が高い．

al-bt の探索戦略は深さ優先であり，図 4 の 41,42 行において，コスト値の境界が未収束の変数値を順に列挙しそのコスト値を収束させてゆく．これに対し，bi-thr は楽観的探索戦略であり，図 5 の 38,39 行において，現在のコスト値の境界が backtracking threshold を超えれば他の変数値を選択する．このとき，その変数値についてのコスト値は必ずしも収束していないので，再探索が起きる可能性がある．一般には充足可能性が高い max CSP のように，最適解が容易に求まる問題では，楽観的手法の bi-thr が有利と考えられる．

また，親子間のコストの境界値の自由さについてはつぎのようである．al-bt では，実行可能なコストの境界はトップダウンに強制される，子ノードは親からの alpha, beta 値に必ず従わなければならない．このとき，過剰な境界を設定することを避けるために，複数の子があっても境界値はそれらに対して分割されず，枝刈りの効果が少なくなる．その一方で，bi-thr では，子ノードは親からの backtracking threshold を修正できる．そのため境界値の投機的な分割が許容される．さらに，al-bt では各エー

ジェントは境界の片側だけを変更するが，bi-thr ではすべてのエージェントが境界の両側を変更できる．これらのことは，境界の収束までの遅延に影響すると考えられる．

max CSP の $t = \frac{5}{9}$ では，min-max のメッセージサイクル数はほぼ下に凸である．これに対し，al-bt と bi-thr は $u = 0$ の場合にメッセージサイクル数がより少ない，これは，コストの下限値 0 による枝刈の効果と考えられる．

COP の cost=$[0, 10]$ では，min-max のメッセージサイクル数は下に凸である．これらの問題のコスト関数は一様なコスト値をとる．さらに，多くの場合はコスト値は非ゼロである．そのため，最大化と最小化を行うエージェントの数がほぼ等しいとき，上下界の収束は比較的早くなると考えられる．

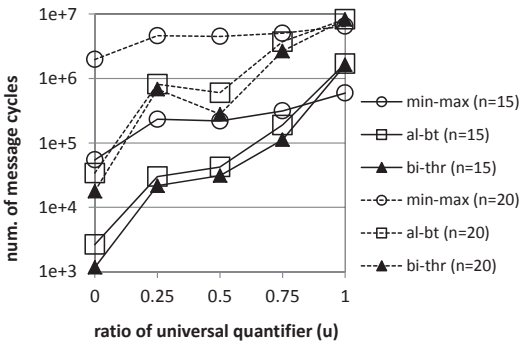
$0 \leq u \leq 0.5$ の場合は，bi-thr は多くのメッセージサイクルを要している．この主な原因は，先述した，bi-thr の楽観的戦略に原因があると考えられる．bi-thr と類似の探索戦略を持つと考えられるオリジナルの ADOPT は，コスト値が広い範囲にわたるとき既に探索された部分解に対する下界値を反復的に改善する傾向が大きくなる [Ali 05, Matsui 09]．このような状況が，存在限量された変数が多く，従来の DCOP に近い問題で発生していると考えられる．その一方で，深さ優先探索およびトップダウンに強制される境界を用いる al-bt は，bi-thr のように過剰に自由な探索を行うことが無いため，より少ないメッセージサイクル数を要すると考えられる．

解法の終了に要するメッセージサイクル数について表 2 に示す．root および global は，それぞれ根およびすべてのエージェントの処理が終了するまでのメッセージサイクル数を表す．また，ratio は，root のメッセージサイクル数に対する global のメッセージサイクル数の比を表す．終了処理では部分的に探索がリセットされる場合があるが，これによる探索処理の増加はいずれの手法でも比較的小さい．

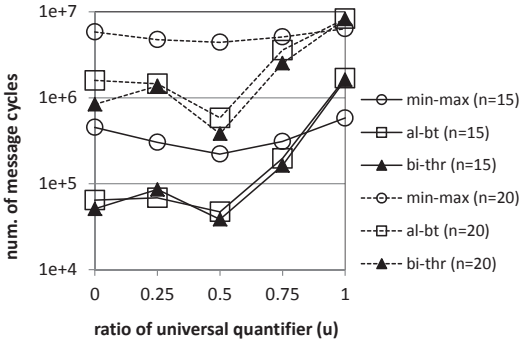
解法を構成する部分的な処理の効果について表 3 に示す．tree-VALUE は疑似木の後退辺に沿う VALUE を用いない．linear-tree は，通常の疑似木の代わりに，葉ノード以外のノードでは子ノードが 1 つであるような疑似木を用いる．提案手法との結果の比較から，これらにおいて省略された処理の有効性が示されていると考えられる．すなわち，QDCOP のために拡張された提案手法においても，後退辺に沿う追加的な VALUE メッセージによる遅延の短縮と，複数の部分木からなる疑似木による計算の並列性は有効であると考えられる．

3·8 節で示したコストの下限値を $-\infty$ に一般化した手法についても評価した．図 6(d) はコスト値が -5 と 5 の間の整数値である場合のメッセージサイクル数を示す．$u = 0$ 付近では効果的な枝刈が行われないため，グラフは下に凸となっている．

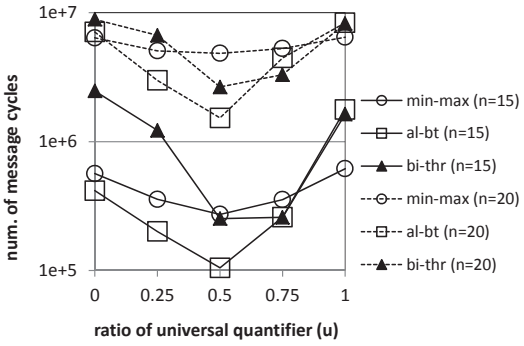
また，3·7 節の境界付誤差を導入した手法の結果を図 7 に示す．これらの結果では，max CSP の $u = 0$ 付近で



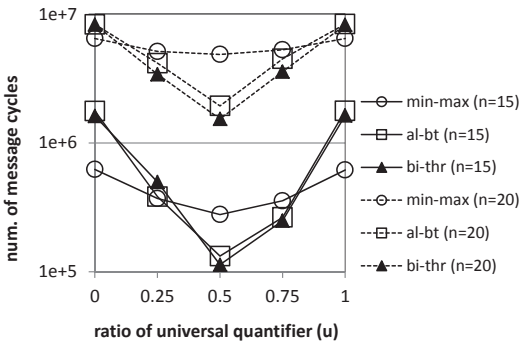
(a) max CSP, $l = 2$, $t = \frac{1}{3}$



(b) max CSP, $l = 2$, $t = \frac{5}{9}$



(c) COP, $l = 2$, cost= $[0, 10]$



(d) COP, $l = 2$, cost= $[-5, 5]$

図 6 メッセージサイクル数

bi-thr のサイクル数が顕著に削減された．境界付誤差は従来の DCOP における ADOPT の探索を削減するためには効果的である一方で，QDCOP とその解法に単純に適用するだけでは十分な効果が得られず，今後の検討の

表 2 解法の終了に要するメッセージサイクル数

(max CSP, $n = 15, l = 2, t = \frac{5}{9}$)

u	0			0.5			1		
algorithm	root	global	ratio	root	global	ratio	root	global	ratio
min-max	360917	452636	1.25	168015	221393	1.32	467045	580420	1.24
al-bt	54622	64442	1.18	33316	47156	1.42	1380234	1688324	1.22
bi-thr	46623	51042	1.09	30616	38594	1.26	1380418	1449839	1.05

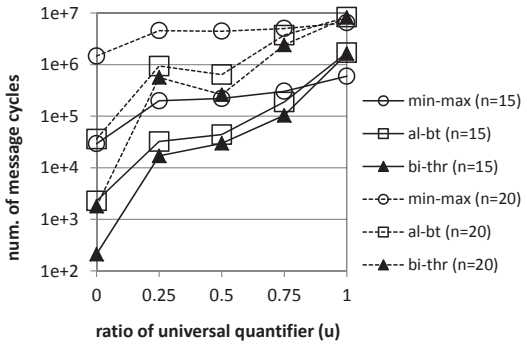
u=1 の al-bt と bi-thr についてはメッセージサイクル数の上限内に終了した 49 例について統計した.

表 3 解法を構成する部分的な処理の効果 (メッセージサイクル数)

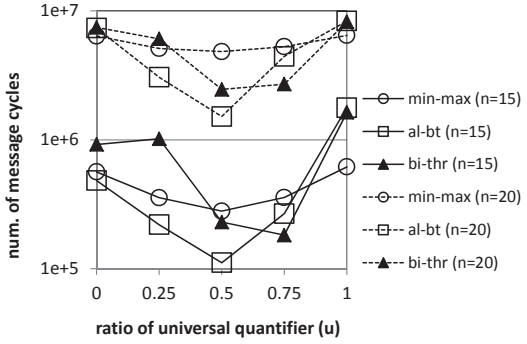
(max CSP, $n = 15, l = 2, t = \frac{5}{9}, u = 0.5$)

algorithms	all	tree-VALUE	linear-tree
min-max	221393	269906	718486
al-bt	47156	53915	81354
bi-thr	38594	41413	101174

(average depth of pseudo-trees is 8.8)



(a') max CSP, $l = 2, t = \frac{1}{3}, e = 3$



(c') COP, $l = 2$, cost= $[0, 10], e = 20$

図 7 メッセージサイクル数 (境界付誤差)

余地があることを示唆していると考えられる.

5. 関 連 研 究

QDCOP の定義は Q(D)CSP [Chen 04, 馬場 11] の定義の拡張と考えられる. 基本的な Q(D)CSP は QDCOP として表現できる. 例えば, hard な制約を用いた $\forall x_0 \exists x_1 . x_0 \neq x_1$ および $\exists x_0 \forall x_1 . x_0 \neq x_1$ はコスト関数 $f : D_0 \times D_1 \rightarrow \{0, 1\}$ を用いて表現できる. ここで, 0 と 1 は真と偽を表す. $D_0 = D_1$ であるとき, 各問題の最適コスト値はそれぞれ 0 および 1 となる.

QCSP の緩和は [Ferguson 07] に述べられている. これに対し本研究の目的は DCOP の一般化である. 拡張された QCSP である QCOP/QCOP+が [Benedetti 08] で提案されている. QCOP/QCOP+では, QCSP に対して追加された目的関数と制約が定義される. この問題の定義は本研究とは異なる. min-max 法を動的計画法に基づく解法 [Petcu 05] として構成することは可能である. しかし, 簡単な動的計画法は, 疑似木の幅が比較的大きいときは, 空間複雑度が指数関数的であるために適用できない. ADOPT のメッセージサイクル数を削減するための手法 [Matsui 09, Silaghi 09] は, 提案手法に適用できる可能性があると考えられる.

alpha-beta 法に基づく手法として, scout [Reinefeld 83] や MTD(f) [Plaat 96] などの効率的な解法が用いられる. これらの手法では, 必ず枝刈りが起きるような alpha,beta 値によって得られる評価値を活用するために null-window search が用いられる. 本研究の alpha-beta ADOPT では, 従来解法の ADOPT に alpha-beta 法の要素を導入することを中心とした. そのため, 解法に用いられる規則には, alpha-beta 法と厳密に対応しない部分がある. 提案手法と本来の alpha-beta 法がどの程度等価であるか, 提案手法においても null-window search と同様の手法を適用できるかの検証が必要である. そのうえで, このような投機的な評価値による探索を繰り返し行う手法を導入して解法を効率化することも課題として挙げられる.

6. お わ り に

本研究では分散制約最適化問題 (DCOP) を拡張し, 限量記号付き分散制約最適化問題 (QDCOP) を提案した. QDCOP では, 各エージェントは存在または全称限量された変数を持つ. コスト値の境界はすべての全称限量された変数のすべての値を考慮して計算される. また, QDCOP のために, min-max, alpha-beta および ADOPT に基づく解法を示し, これらの解法の特性を実験により評価した. QDCOP は従来の DCOP より探索空間が大きいため, 実験では比較的小規模な問題を用いたが, それぞれの解法の特徴を示したことには意義があると考える. 今後の課題として, グラフの構造や限量子の組み合わせの影響についてのより詳細な解析, DCOP 解法の効率化手法を応用することによる QDCOP 解法の改良, 実際的な問題への応用が挙げられる.

謝 辞

本研究の一部は科学研究費補助金 (基盤研究 B, 課題番号 23300060) による.

◇ 参 考 文 献 ◇

[Ali 05] Ali, S. M., Koenig, S., and Tambe, M.: Preprocessing techniques for accelerating the DCOP algorithm ADOPT, in *4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1041–1048 (2005)

[Benedetti 08] Benedetti, M., Lallouet, A., and Vautard, J.: Quantified Constraint Optimization, in *14th International Conference on Principles and Practice of Constraint Programming*, pp. 463–477 (2008)
[Chen 04] Chen, H. M.: *The computational complexity of quantified constraint satisfaction*, PhD thesis, Cornell University (2004)
[Delle Fave 11] Delle Fave, F. M., Stranders, R., Rogers, A., and Jennings, N. R.: Bounded decentralised coordination over multiple objectives, in *10th International Conference on Autonomous Agents and Multiagent Systems*, Vol. 1, pp. 371–378 (2011)
[Farinelli 08] Farinelli, A., Rogers, A., Petcu, A., and Jennings, N. R.: Decentralised coordination of low-power embedded devices using the max-sum algorithm, in *7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 639–646 (2008)
[Ferguson 07] Ferguson, A. and O'Sullivan, B.: Quantified Constraint Satisfaction Problems: from relaxations to explanations, in *20th International Joint Conference on Artificial Intelligence*, pp. 74–79 (2007)
[Knuth 75] Knuth, D. and Moore, R. W.: An analysis of alpha-beta pruning, *Artificial Intelligence*, Vol. 6, pp. 293–326 (1975)
[Kumar 09] Kumar, A., Faltings, B., and Petcu, A.: Distributed constraint optimization with structured resource constraints, in *8th International Conference on Autonomous Agents and Multiagent Systems*, pp. 923–930 (2009)
[Léauté 11] Léauté, T. and Faltings, B.: Distributed Constraint Optimization Under Stochastic Uncertainty, in *25th AAAI Conference on Artificial Intelligence*, pp. 68–73 (2011)
[Maheswaran 04] Maheswaran, R. T., Tambe, M., Bowring, E., Pearce, J. P., and Varakantham, P.: Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling, in *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 310–317 (2004)
[Mailler 04] Mailler, R. and Lesser, V.: Solving distributed constraint optimization problems using cooperative mediation, in *3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 438–445 (2004)
[Matsui 09] Matsui, T., Silaghi, M. C., Hirayama, K., Yokoo, M., and Matsuo, H.: Directed soft arc consistency in pseudo trees, in *8th International Conference on Autonomous Agents and Multiagent Systems*, pp. 1065–1072 (2009)
[Modi 05] Modi, P. J., Shen, W., Tambe, M., and Yokoo, M.: Adopt: Asynchronous distributed constraint optimization with quality guarantees, *Artificial Intelligence*, Vol. 161, No. 1-2, pp. 149–180 (2005)
[Petcu 05] Petcu, A. and Faltings, B.: A Scalable Method for Multiagent Constraint Optimization, in *19th International Joint Conference on Artificial Intelligence*, pp. 266–271 (2005)
[Petcu 07] Petcu, A. and Faltings, B.: Optimal Solution Stability in Dynamic, Distributed Constraint Optimization, in *2007 IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pp. 321–327 (2007)
[Plaat 96] Plaat, A., Schaeffer, J., Pijls, W., and Bruin, de A.: Best-first fixed-depth minimax algorithms, *Artificial Intelligence*, Vol. 87, No. 1-2, pp. 255–293 (1996)
[Reinefeld 83] Reinefeld, A.: An Improvement to the Scout Tree-Search Algorithm, *ICCA Journal*, Vol. 6, No. 4, pp. 4–14 (1983)
[Russell 03] Russell, S. and Norvig, P.: *Artificial Intelligence: A Modern Approach*, Prentice Hall, second edition (2003)
[Silaghi 09] Silaghi, M. C. and Yokoo, M.: ADOPT-ing: unifying asynchronous distributed optimization with asynchronous backtracking, *Journal of Autonomous Agents and Multi-Agent Systems*, Vol. 19, No. 2, pp. 89–123 (2009)
[Yeoh 08] Yeoh, W., Felner, A., and Koenig, S.: BnB-ADOPT: an asynchronous branch-and-bound DCOP algorithm, in *7th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 591–598 (2008)
[Zivan 08] Zivan, R.: Anytime Local Search for Distributed Constraint Optimization, in *23rd AAAI Conference on Artificial Intelligence*, pp. 393–398 (2008)
[松井 05] 松井 俊浩, 松尾 啓志, 岩田 彰：分散制約最適化問題の非同期分散探索における上下界の導出と学習を用いた効率改善手法 (分散協調とエージェント), 電子情報通信学会論文誌. D-I,, Vol. 88, No. 8, pp. 1235–1246 (2005)
[馬場 11] 馬場 里美, 岩崎 敦, 横尾 真：敵対者に対応する協調問題解決：限量記号付き分散制約充足問題, 人工知能学会論文誌, Vol. 26, No. 1, pp. 136–146 (2011)

〔担当委員：岸本 章宏〕

2012 年 7 月 20 日 受理

著 者 紹 介

松井 俊浩（正会員）

1995 年名古屋工業大学電気情報工学科卒業．1999 年同大大学院博士前期課程修了．2006 年同博士後期課程修了．2006 年名古屋工業大学情報基盤センター助手．2007 年同助教．2011 年同准教授．現在に至る．分散協調処理，マルチエージェントシステム，分散制約最適化問題に関する研究に従事．博士 (工学)．電子情報通信学会，情報処理学会，各会員．

Marius C. Silaghi

1997 年 Cluj-Napoca 工科大学卒業．1997 年ブラウンシュヴァイク工科大学，Scientific Computing Laboratory, DAAD Researcher．1998–2002 年スイス連邦工科大学 Artificial Intelligence Laboratory, Research Assistant．2002 年スイス連邦工科大学，Ph.D．2002 年フロリダ工科大学，Assistant Professor．Ph.D. (Computer Science)．AAAI (2000–2007), IEEE (2001–2008) 各会員．

平山 勝敏（正会員）

1990 年 大阪大学基礎工学部制御工学科卒．1992 年 同大大学院基礎工学研究科博士前期課程修了．1995 年 同大大学院基礎工学研究科博士後期課程修了．博士（工学）．1995 年 神戸商船大学助手．1997 年 同講師．2001 年 同助教授．2003 年 神戸大学海事科学部助教授（神戸大学と神戸商船大学の統合による）．2007 年 神戸大学大学院海事科学研究科准教授．1999 年–2000 年 カーネギーメロン大学ロボティクス研究所客員研究員（文部省在外研究員）．マルチエージェントシステム，制約充足，組合せ最適化に関する研究に従事．電子情報通信学会，情報処理学会，日本オペレーションズリサーチ学会，AAAI 各会員．

横尾 真（正会員）

1984 年東京大学工学部電子工学科卒業．1986 年同大学院修士課程修了．同年 NTT に入社．1990 年〜1991 年 ミシガン大学客員研究員．2004 年 九州大学大学院 システム情報科学研究院 教授．2012 年より同大学 主幹教授マルチエージェントシステム，制約充足問題に関する研究に従事．エージェントの合意形成メカニズム，制約充足／分散制約充足等に興味を持つ．博士 (工学)．1992 年,2002 年 人工知能学会論文賞, 1995 年 情報処理学会坂井記念特別賞, 2004 年 Association for Computing Machinery (ACM) Special Interest Group on Artificial Intelligence (SIGART) Autonomous Agent Research Award, 2005 年 ソフトウェア科学会論文賞, 2006 年学士院学術奨励賞, 2010 年人工知能学会業績賞, International Foundation for Autonomous Agents and Multiagent Systems influential paper award 受賞．情報処理学会, AAAI フェロー，日本ソフトウェア科学会, 電子情報通信学会各会員．

松尾 啓志（正会員）

1983 年名古屋工業大学情報工学科卒業．1985 年同大学大学院修士課程修了．同年松下電器産業 入社．1989 年名古屋工業大学大学院博士課程修了．同年名古屋工業大学電気情報工学科助手．講師，助教授を経て，2003 年同大学大学院教授．2006 年同大学情報基盤センターセンター長（併任）．2011 年付属図書館長．現在に至る．分散システム，分散協調処理に関する研究に従事．工学博士．電子情報処理学会, 情報処理学会，IEEE 各会員．