原著論文 Original Paper

資源制約に束縛されないpseudo-treeを用いた資源制約付き分散制約最適化問題の解法

A Resource Constrained Distributed Constraint Optimization Method using Resource Constraint Free Pseudo-tree

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
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: distributed constraint optimization multi-agent resource

Summary

Cooperative problem solving with shared resources is important in practical multi-agent systems. Resource constraints are necessary to handle practical problems such as distributed task scheduling with limited resource availability. As a fundamental formalism for multi-agent cooperation, the Distributed Constraint Optimization Problem (DCOP) has been investigated. With DCOPs, the agent states and the relationships between agents are formalized into a constraint optimization problem. However, in the original DCOP framework, constraints for resources that are consumed by teams of agents are not well supported. A framework called Resource Constrained Distributed Constraint Optimization Problem (RCDCOP) has recently been proposed. In RCDCOPs, a limit on resource usage is represented as an n-ary constraint. Previous research addressing RCDCOPs employ a pseudo-tree based solver. The pseudo-tree is an important graph structure for constraint networks. A pseudo-tree implies a partial ordering of variables. However, n-ary constrained variables, which are placed on a single path of the pseudo-tree, decrease efficiency of the solver. We propose another method using (i) a pseudo-tree that is generated ignoring resource constraints and (ii) virtual variables representing the usage of resources. However the virtual variables increase search space. To improve pruning efficiency of search, (iii) we apply a set of upper/lower bounds that are inferred from resource constraints. The efficiency of the proposed method is evaluated by experiment.

1. ま え が き

資源の制約を伴う分散協調問題解決は，実際的な複数エージェントシステムにおいて重要である．資源制約は，限られた資源や予算を共有する分散タスクスケジューリングなどを表現するために必要となる．

複数エージェントの協調の基礎的なモデルとして，分散制約最適化問題 (DCOP) [Ali 05, Maheswaran 04, Mailler 04, Modi 05, Petcu 05] が研究されている．DCOP では，エージェントの状態とエージェント間の関係が，制約最適化問題として形式化される．近年，DCOP に資源に関する制約を明示的に導入した，資源制約付き制約最適化問題 (RCDCOP) が提案された [Bowring 06, Pecora 06]. RCDCOP では目的関数と資源制約が区別される．資源制約は，エージェントが用いる資源の総量の制限を表す多項制約として定義される．[Bowring 06] では，多項制約である資源制約と各エージェントのプライバシを考慮した，問題と解法が示されている．また，[Pecora 06] では，資源制約を伴う分散タスクスケジューリングが，多項制約を伴う RCDCOP として形式化されている．

従来手法では，DCOP の解法である ADOPT[Modi 05] を RCDCOP に適用している．ADOPT は制約網に対する pseudo-tree にもとづく変数の半順序を用いる．1 つの

多項制約で関係する複数の変数は pseudo-tree の 1 つのパスに配置される．したがって，arity の大きな資源制約を含む問題では，pseudo-tree の深さは大きくなり，分枝係数は小さくなる．その結果として ADOPT の探索処理における並列性は減少する．

一方で，基本的な資源制約は，複雑な組み合わせ探索問題を構成するものではなく，複数のエージェントにより共有される資源の合計量を制限するような，比較的簡単な制約である．そのため，pseudo-tree の複数の異なるサブツリーに含まれる変数が，同一の資源制約で関係することを許容するような拡張は可能であると考えられる．本論文では， 資源制約に制限されない pseudo-tree を許容するような ADOPT の拡張手法を提案する．提案手法では，(i) 資源制約を無視して pseudo-tree を生成し，(ii) 資源の使用量をあらわす仮想変数を新たに導入する．仮想変数は，pseudo-tree の部分木の間で，資源を分配するために用いられる．これらの手法により，資源制約を無視した pseudo-tree を用いた探索が可能となる．しかし，仮想変数の導入により探索空間は増大する．そのため特に資源制約を満足することが容易な問題では，提案手法は従来手法よりも多くの探索処理を要する．この対策として，さらに (iii) 資源制約から導出されるコストの境界を用いて，探索における枝刈りを行う手法を導入する．提案手法の効率は問題の性質に依存するが，上記 (iii) の枝刈りを用いることで，比較的効率的な探索が行われることを実験結果により示す．

以下では，まず 2 章で RCDCOP の定義を示し，3 章で RCDCOP に ADOPT を適用した従来手法を示す．そして，4 章において提案手法を示し，5 章 で実験による評価を示す．最後に 6 章でまとめを述べる．

2. 問 題 の 定 義

2·1 資源制約付き分散制約最適化問題 (RCDCOP)

分散制約最適化問題 (DCOP) は，エージェントの集合 A，変数の集合 X および 2 項関数の集合 F により定義される．エージェント i は自身の状態を表す変数 x_i を持つ．x_i は離散かつ有限の値域 D_i の値をとる．x_i の値はエージェント i のみにより決定される．変数値の割り当ての組 $\{(x_i,d_i),(x_j,d_j)\}$ のコストは二項関数 $f_{i,j}(d_i,d_j) : D_i \times D_j \to \mathbb{N}$ により定義される．問題の目的は，大域的コスト関数 $\sum_{f_{i,j}\in F,\,\{(x_i,d_i),(x_j,d_j)\}\subseteq\mathcal{A}} f_{i,j}(d_i,d_j)$ を最小化する，大域的最適解 $\mathcal{A}$ を得ることである．

資源制約付き分散制約最適化問題 (RCDCOP) では，DCOP に資源制約が加えられる．資源制約は資源の集合 R および資源の要求量の集合 U により定義される．資源 $r_a \in R$ は，$C(r_a) : R \to \mathbb{N}$ により定義される容量 $C(r_a)$ を持つ．各エージェントは自変数値に応じた各資源の量を要求する．変数値の割り当て (x_i,d_i) と資源 r_a に対して，資源の要求量 $u_i(r_a,d_i)$ が $u_i(r_a,d_i) : R \times D_i \to \mathbb{N}$ により定義される．各資源について，資源の要求量の合計は，資源の容量を超えてはならない．大域的な資源制約は，$\forall r \in R, \sum_{u_i\in U,\,\{(x_i,d_i)\}\subseteq\mathcal{A}} u_i(r,d_i) \le C(r)$ により定義される．資源制約は任意の arity を持つ．5 個の変数と 2 個の資源からなる RCDCOP の例を図 1 に示す．この例では，変数 x_0，x_2 および x_3 は資源 R_0 と関連する．また，変数 x_0，x_1 および x_4 は資源 R_1 と関連する．

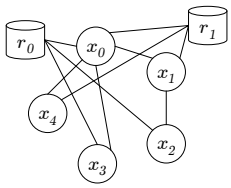


図 1 資源制約付き分散制約最適化問題 (RCDCOP)

本論文ではエージェントと変数が 1 対 1 に対応することから，記述の簡略化のために必要に応じて両者を区別せずに用いる．

3. 準備 : ADOPT を用いた RCDCOP の解法

従来研究では，多項制約である資源制約をともなう RCDCOP の解法として，ADOPT アルゴリズムが用いられている．ADOPT は，制約網に対する pseudo-tree を用いる，DCOP の解法である．本節では，pseudo-tree, ADOPT および多項制約のための ADOPT の拡張方法を示す．

3·1 Pseudo-tree

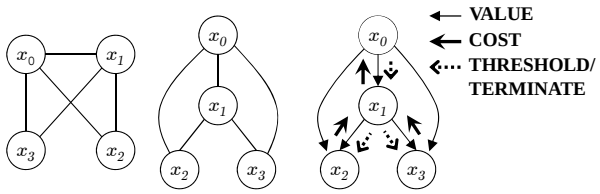
ADOPT アルゴリズムは，制約網に対する pseudo-tree にもとづく変数順序を用いる．あるグラフに対する pseudo-tree は次のような性質をすべて満たすような「疑似的な木」である．

(1) pseudo-tree は元のグラフと同一の辺と頂点からなる．
(2) pseudo-tree は元のグラフの生成木のいずれか一つに対応する．元のグラフのすべての辺は，生成木の辺 (木辺) かそれ以外の辺 (後退辺) に分類される．
(3) 生成木の根ノードからいずれか 1 つの葉ノードへのパスに含まれる 2 ノード間にのみ，後退辺がある．異なるパスに含まれる 2 ノード間には，後退辺はない．

pseudo-tree に対応する生成木の各頂点を，pseudo-tree においても同様に木の頂点とみなす．制約網と対応する pseudo-tree の例を図 2(a),(b) に示す．

典型的な pseudo-tree は，元のグラフに対する深さ優先探索木を木辺とする pseudo-tree である．また，元のグラフに任意に辺を加えて生成したグラフに対する psueudo-tree も元のグラフの pseudo-tree とみなす．このような拡張は，3·3 節で示すような場合に必要となる．

本論文では，pseudo-tree は制約網に対する深さ優先探索木にもとづき，ADOPT の前処理の段階で生成されるも



(a) constraint network (b) pseudo-tree (c) messages of ADOPT

図 2 pseudo-tree と ADOPT のメッセージ

のとする．上述の性質により，pseudo-tree の異なるサブツリー間には辺は存在しない．そのため，pseudo-tree の木辺にもとづく変数の半順序を用いることにより，ADOPT の探索処理をサブツリーごとに並列的に実行できる．並列度の高い pseudo-tree を生成するために，制約網に対する深さ優先探索において，次数の大きなノードを優先的に子ノードとして選択する方法が用いられる．

3·2 ADOPT

ADOPT[Modi 05] は DCOP の解法の一つである．3·1 節に示したように ADOPT では, 前処理により生成された pseudo-tree に基づく，変数の半順序を用いる．以下では，各変数の順序関係を，pseudo-tree の木辺に従い，親, 子，祖先，子孫などの用語で表現する．ここでは祖先は親を，子孫は子をそれぞれ含む．また，ある変数と制約で直接関係する他の変数を，近傍という用語により表現する．とくに祖先である近傍を上位近傍，子孫である近傍を下位近傍と表す．

ADOPT では pseudo-tree の辺に従って，次の 4 種類のメッセージを送受しつつ計算を行う．

- VALUE(x,d): 各変数 x の現在の値 d を，下位近傍の変数を持つエージェントに通知する．
- COST($context$,lb,ub): ある変数を根とするサブツリーにおけるコストの合計の上下界値 ub,lb を，その変数の親の変数を持つエージェントに通知する．上下界値の計算に用いた，祖先の変数についての現在の部分解 $context$ も同時に通知される．
- THREHOLD($context$,t): COST メッセージを用いて集計されたコストにもとづき，各エージェントは自身の変数値を決定する．集計されたコストから自身の変数値に対するコストを除いた残りのコストは，子の変数を持つ各エージェントに配分される．このコストの配分 t を THREHOLD により通知する．コストの配分にもとづき，必要であれば子の変数の値も変更される．t の計算に用いた現在の部分解 $context$ も同時に通知される．
- TERMINATE($context$): アルゴリズムの終了段階では，根から葉に向かって順に，最適解の変数値を決定する．このとき各変数とその祖先の変数の変数値からなる部分解 $context$ を，各変数の子の変数を持つエージェントに TERMINATE により通知する．

各メッセージの送受信経路の例を図 2(c) に示す．

これらのメッセージを用いる ADOPT の処理は次の 2 つの段階に分けられる．

- **大域的な最適コストの計算:** 各エージェントは大域的なコストの境界を pseudo-tree に従って計算する．この処理では VALUE, COST, THRESHOLD メッセージが用いられる．この計算の結果，pseudo-tree の根の変数を持つエージェントでは，コストの上下界値が最適値に収束する．
- **大域的な最適解の決定:** 最適コストが求められた後，pseudo-tree に従ってトップダウンに，最適解が決定される．この処理では他の 3 つのメッセージに加えて，TERMINATE メッセージが用いられる．

上記の処理の詳細はやや複雑であるため，本論文では次のように ADOPT の一部の処理に注目する．まず，提案手法により重要な変更が ADOPT に加えられる，大域的な最適コストの計算を議論の対象とする．大域的な最適コストの計算は，3 つのメッセージを用いて，変数値の変更と通知およびそれにもとづくコストの集計を反復する．このうち，THRESHOLD メッセージを用いる各変数値の変更の計算は，基本的には，既知のコストの最良の下界値に対応する変数値を選択するものとして，簡略する．残りの VALUE メッセージ，および COST メッセージにかかわる要素について詳細を示す．なお，ADOPT の詳細は [Modi 05] に示される．

エージェント i は次の情報を用いてコストの情報を計算する．

- x_i: エージェント i の変数. 変数 x_i の値 d_i は，制約で直接関連する x_i の下位近傍の変数を持つエージェントへ，VALUE メッセージを用いて送信される．
- $current\_context_i$: x_i の祖先の変数についての現在の部分解．VALUE メッセージに含まれる変数値により，$current\_context_i$ は，更新される．これは上位近傍の現在の変数値を保持するために必要である．また，x_i を根とするサブツリー全体のコストの集計のために，x_i の子孫の変数と制約で関係する，x_i の祖先の変数値も保持する必要がある．これらの変数値は，x_i の子の変数を持つエージェントからの COST の $context$ にもとづいて，更新される．
- $context_i(x,d), lb_i(x,d), ub_i(x,d)$: 変数 x_i の値 d と，x_i の子の変数 x を根とするサブツリーに対する，最適コストの上下界の情報．これらは,x_i の子の変数 x を持つエージェントから，COST メッセージにより受信される．もしも $current\_context_i$ が $context_i(x,d)$ を含むとき，コストの上界は $ub_i(x,d)$，下界は $lb_i(x,d)$ である．これらの情報は $context_i(x,d)$ が $current\_context_i$ に含まれるかぎり保持される．もしも $current\_context_i$ が $context_i(x,d)$ と矛盾するとき，$context_i(x,d)$, $lb_i(x,d)$ および $ub_i(x,d)$ はそれぞれ，$\{\}$，0 および ∞ に初期化される．

エージェント i におけるコストの計算は次のように示

される．変数 x_i の値 d および $current\_context_i$ に対する，局所コスト $\delta_i(d)$ は次式のように定義される．

$$\delta_i(d) = \sum_{\substack{j \text{ s.t. } (x_j,d_j)\in current\_context_i,\\ x_j \in upper\ neighborhood\ of\ x_i}} f_{i,j}(d,d_j) \quad (1)$$

変数 x_i の値 d および x_i を根とするサブツリーに対する，コストの上界 $UB_i(d)$ と下界 $LB_i(d)$ は次のように定義される．

$$LB_i(d) = \delta_i(d) + \sum_{j \text{ s.t. } x_j \in children\ of\ x_i} lb_i(x_j,d) \quad (2)$$

$$UB_i(d) = \delta_i(d) + \sum_{j \text{ s.t. } x_j \in children\ of\ x_i} ub_i(x_j,d) \quad (3)$$

x_i を根とするサブツリーに対するに対する上界 UB_i と下界 LB_i は次のように定義される．

$$LB_i = \min_{d\in D_i} LB_i(d) \quad (4)$$

$$UB_i = \min_{d\in D_i} UB_i(d) \quad (5)$$

初期の状態において，LB_i は 0 であり，UB_i は ∞ である．

各エージェント i は VALUE(x_i, d_i) メッセージを下位近傍の変数を持つエージェントに，COST($current\_context_i$, LB_i, UB_i) メッセージを親の変数を持つエージェントに，それぞれ送信する．これらのメッセージ交換にもとづき，x_i, $current\_cotnext_i$, $context_i(x,d)$, $lb_i(x,d)$, $ub_i(x,d)$ を更新する．

根の変数を持つエージェント r では，コストの境界が，$LB_r = UB_r$ なる大域的最適コストに収束する．大域的最適解はこの最適コストにしたがって決定される．

3·3 資源制約で関連する変数の直列化

従来研究 [Pecora 06] では，資源制約で関連する変数を直列化する手法を用いた ADOPT アルゴリズムが提案されている．pseudo-tree は資源制約を考慮して生成される．一つの多項制約で関連する変数は，pseudo-tree の根から葉への同一のパス上に配置される．

このような pseudo-tree は制約網に対する深さ優先探索を拡張した手法により生成できる．この手法は，基本的には，従来の深さ優先探索と同様に各ノードを列挙することで，pseudo-tree を構成する．ただし，資源制約で関係する変数を，根ノードから単一の葉ノードへのパスに配置するために，次のような規則を用いる．

> **規則 1:**あるノードの子ノードを選択する時，すでに木に含まれる変数と資源制約で関係する変数が，まだ木に含まれていなければ，そのような変数から子ノードを選ぶ．

しかし，この手法によって生成される木の辺は，元の制約網には存在しない場合がある．この矛盾を解消するために，元の制約網に冗長な辺を挿入したものとみなす．追加される冗長な辺に対応する評価関数の値は常に 0 とする．

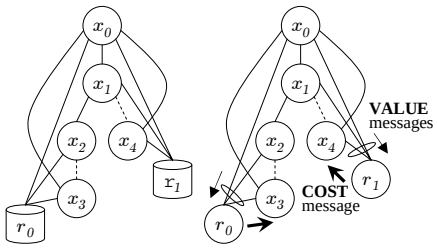


図 3 資源制約で関連する変数の直列化

例として，図 1 で示した RCDCOP に対して生成される pseudo-tree を図 3(a) に示す．この例では，資源 r_0 に関連する変数 x_0，x_2 および x_3 が pseudo-tree の同一のパス上に配置されている．資源 r_1 に関連する変数 x_0，x_1 および x_4 も同様である．もしも変数を直列化するために必要であれば，親子ノード間に木辺が挿入される．図 3(a) の例では，木辺 (x_2,x_3) および (x_1,x_4) が挿入されている．

ADOPT アルゴリズムにおいては，**資源評価エージェント**と呼ばれる，資源制約を評価するためのエージェントが導入される．資源評価エージェントは自身の変数を持たないが，直列化された変数のうちで最下位の変数の子とみなされる．たとえば，図 3(b) では，資源評価エージェントに対応する r_0 および r_1 が，それぞれ x_3 と x_4 の子として追加されている．各エージェントは自身の変数値を，**VALUE** メッセージを用いて，資源評価エージェントへ送信する．資源評価エージェントは，受信した各エージェントの変数値に従って，資源の要求量の合計を評価する．もしも，資源制約が満足されないならば，資源評価エージェントは親の変数を持つエージェントに **COST** メッセージを用いて通知する．このような資源制約の違反は∞のコスト値として表わされる．

上記のような資源評価エージェントを個別のエージェントとして導入することは可能である．しかし，そのエージェントは自身の変数を持たず，上位の変数値にもとづいて資源制約の評価のみを行う特別なエージェントとなる．それよりも，ある資源について最下位の変数を持つエージェントに，その資源評価エージェントの機能を統合する方が簡単である．このようにすれば，従来のエージェントに，資源制約に対応する評価関数を追加する以外にアルゴリズムの大きな変更は必要ない．

この方法での ADOPT アルゴリズムへの変更点は，資源評価エージェントの追加と，∞のコスト値に対応したコスト計算の一般化のみである．一方で，大きな arity を持つ資源制約は，pseudo-tree の深さを増加させ，分枝係数を減少させる．そのため，探索処理の並列性は減少する．

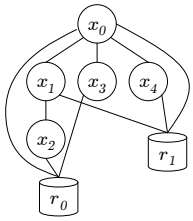
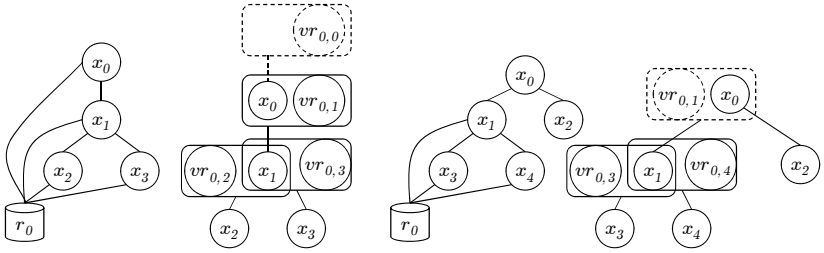


図 4 資源制約に束縛されない pseudo-tree



(a) pseudo-tree (b) virtual variables (a') pseudo-tree (b') virtual variables

図 5 仮想変数の導入

4. 資源制約に束縛されない pseudo-tree を用いた RCDCOP の解法

本研究では，資源制約に束縛されない pseudo-tree を用いた RCDCOP の解法を提案する．提案手法では，異なるサブツリーに配置された変数が同一の資源制約で関連することを許容する．すなわち，pseudo-tree は資源制約を無視して生成される．例として，図 1 で示した RCDCOP から生成される pseudo-tree を図 4 に示す．この例では，資源 r_0 について制約で関係する x_2 と x_3 は，2 つの異なるサブツリーに含まれている．同様に，資源 r_1 についての制約で関連する x_1 と x_4 も異なるサブツリーに含まれている．

元の ADOPT では，異なるサブツリーに含まれる制約辺は許容されない．そのような場合，祖先の変数への複数の異なるパスが存在するため，正確なコスト情報を含む **COST** メッセージを，親の変数を持つエージェントに通知することができない．

4·1 仮想変数の導入

提案手法の主要なアイデアは，資源の使用量を表す仮想変数を導入することである．各エージェントは，仮想変数を用いて，親および子の変数を持つエージェントと資源を分配する．仮想変数 $vr_{a,i}$ は，資源 r_a および変数 x_i に対して定義される．ここで，x_i を根とするサブツリーに含まれるいずれかの変数について，資源 r_a が要求される．仮想変数 $vr_{a,i}$ は x_i の親の変数のエージェントが持ち，その値を選択する．$vr_{a,i}$ の値域は資源の使用量を表す離散値の集合 $\{0,1,\cdots,C(r_a)\}$ で表わされる．簡単な例として，一つの資源制約と関連する pseudo-tree の例を図 5(a) に示す．この例では，資源 r_0 が変数 x_0，x_1，x_2 および x_3 と関係している．これらの資源と変数について，同図 (b) に示すように，仮想変数 $vr_{0,1}$，$vr_{0,2}$ および $vr_{0,3}$ が導入される．各仮想変数 $vr_{a,i}$ は，x_i の親の変数のエージェントが持ち，$vr_{a,i}$ の値はそのエージェントが選択する. ただし，根の x_0 には親の変数が無い．このような根の変数 x_0 についての仮想変数 $vr_{0,0}$ の値は仮想的な親のエージェントから与えられたものとする．ここで $vr_{0,0}$ は資源 r_0 の容量 $C(r_0)$ に等しい定数値をとる．

エージェント i が持つ仮想変数 $vr_{a,j}$ の値 $dr_{a,j}$ は，i の子のエージェント j に，**VALUE** メッセージを用いて，送信される．そのため，**VALUE** メッセージは，変数値の割り当て (x_i, d_i) に加えて，仮想変数の変数値の割り当て $(vr_{a,j}, dr_{a,j})$ も含むように変更される．また，エージェント j が，$(vr_{a,j}, dr_{a,j})$ を含む **VALUE** メッセージを受信したとき，エージェント j は $current\_context_j$ を $(vr_{a,j}, dr_{a,j})$ で更新する．

エージェント i において，資源 r_a に対する仮想変数の割り当ては，次式の資源分配制約 $c_{a,i}$ を満足しなければならない．

$$c_{a,i}:\ dr_{a,i} \geq u_i(r_a, d_i) + \sum_{\substack{j \text{ s.t. } x_j \in children\ of\ x_i, \\ x_j\ requires\ r_a}} dr_{a,j} \qquad (6)$$

ここで $dr_{a,i}$ は，i が親のエージェントから受信した，仮想変数 $vr_{a,i}$ の値を示す．仮想変数の値の割り当て $(vr_{a,i}, dr_{a,i})$ は $current\_context_i$ に含まれる．仮想変数の値が資源分配制約 $c_{a,i}$ を満足しないとき，資源分配制約の違反は∞のコスト値として報告される．各エージェント i は $current\_context_i$ に対する最適コストの境界を評価する．そして，そのコストの情報は i の親のエージェントに，**COST** メッセージを用いて送信される．ただし，**COST** に含まれる $current\_context_i$ の情報には，従来の変数値に加えて，i の親のエージェントが持つ仮想変数の値の割り当ても含まれる．

このような仮想変数の導入により，資源制約を無視して pseudo-tree を生成することが許容される．一方で，新たに追加される仮想変数のために，探索空間は増大する．

4·2 仮想変数の生成方法

一般には，各変数が複数の資源の一部または全部と関係する場合がある．このような場合を考慮し，仮想変数は次の規則に従って生成される．

(1) 基本的には，変数 x_i の子の変数 x_j を根とするサブツリーが，資源 r_a を要求するならば，エージェント i は仮想変数 $vr_{a,j}$ を持つ．ただし，以下の特殊な場合が優先される．

(2) 変数 x_i または x_i の子の変数を根とするサブツリーが，資源 r_a を要求するならば，$current\_context_i$ は仮想変数の割り当て $(vr_{a,i}, dr_{a,i})$ を含む. そのとき，$dr_{a,i}$ は次のように決定される．

a x_i のいずれの祖先の変数も，x_i が含まれない他のいずれのサブツリーも，資源 r_a を要求しないならば，x_i は r_a に対する**根**である．そのとき，$dr_{a,i}$ は資源 r_a の容量 $C(r_a)$ に等しい定数として初期化される．

このような場合の例を図 5(a'), (b') に示す．この例では (a') の変数 x_1 は，資源 r_0 に対する根である．このとき (b') に示すように，変数 x_1 のエージェントでは，変数 x_0 のエージェントから，仮想変数 $vr_{0,1}$ の値として定数値 $dr_{0,1} = C(r_0)$ が与えられたものとみなす．

b x_i が r_a に対する**根**ではないならば，x_i の親の変数 x_h のエージェントは，仮想変数 $vr_{a,i}$ を持つ．したがって，エージェント h から i に送信される **VALUE** メッセージは，仮想変数の値の割り当て $(vr_{a,i}, dr_{a,i})$ を含む．

(3) 変数 x_i が資源 r_a を要求し，x_i の子ノードを根とするいずれのサブツリーも r_a を要求しないのであれば，x_i は r_a についての**葉**である．そのとき，エージェント i は r_a に対する仮想変数を持たない．したがって，資源分配制約は $dr_{a,i} \geq u_i(r_a, d_i)$ により定義される．

(4) x_i の子ノード $x_j \in X'$ を根とするサブツリーが資源 r_a を要求するならば，x_i が r_a を要求するか否かにかかわらず，x_i は r_a を子ノード $x_j \in X'$ に分配しなければならない．したがって，エージェント i は仮想変数 $\{vr_{a,j} | j \text{ s.t } x_j \in X'\}$ を持つ．

仮想変数を生成するアルゴリズムの例を図 6 に示す．このアルゴリズムの事前に，資源制約を無視して制約網に対する深さ優先探索を行うことにより，pseudo-tree が生成されているものとする．簡単のため，アルゴリズムは 2 つの段階からなるように構成した．最初の段階では，各エージェント i は変数 x_i を根とするサブツリーに要求される資源の集合 R_i^- を計算する．次の段階では，各エージェント i は i または i の祖先のエージェントから分配される資源の集合 R_i^+ を計算する．これらの結果にもとづいて，エージェント i は自身の変数の集合 $\mathcal{X}_i$ を生成する．この処理は，ADOPT の前処理として，pseudo-tree の生成後に実行される．なお，より効率的な処理のために，pseudo-tree を生成する処理と統合することが考えられる．

```
1  Initiation_i{
2   Generate pseudo−tree ignoring resource constraint.
3   if(i is not root agent) p_i ← parent agent of agent i.
4   C_i ← a set of child agents of agent i.
5   R_i ← a set of resources required by agent i.
6   X_i ← {x_i}.
7   if ( i is root agent ){ call Rootward_i(). call Leafward_i(φ). } }
8  Rootward_i(){
9   R_i^- ← R_i.
10  for each j in C_i{
11   call Rootward_j() and receive R_j^- . R_i^- ← R_i^- ∪ R_j^- . } }
12 Leafward_i(R_pi^+){
13  R_i^+ ← φ.
14  for each r in R_i^- {
15   n ← number of agents j s.t. r ∈ R_j^- .
16   if (n ≥ 2 or ( n = 1 and (r ∈ R_i or r ∈ R_pi^+) ) ){
17    R_i^+ ← R_i^+ ∪ {r}. } }
18  for each j in C_i{
19   for each r in R_j^- {
20    if(r is contained in R_i^+) X_i ← X_i ∪ {vr_{r,j}}. }
21   call Leafward_j(R_i^+). } }
```

図 6 仮想変数を生成するアルゴリズム

4·3 探索空間の増大とその対処

仮想変数の導入により探索空間は増大する．エージェント i は変数の集合 $\mathcal{X}_i = \{x_i\} \cup \{vr_{a,j} | j \in Children_i, r_a \in R_j\}$ に対する変数値の割り当てを選択する．ここで，R_j は変数 x_j を根とするサブツリーで要求される資源の部分集合を表す．エージェント i におけるコストの評価は，それぞれ $\delta_i(\mathcal{D}_i)$, $LB_i(\mathcal{D}_i)$ および $UB_i(\mathcal{D}_i)$ のように変更される．ここで，$\mathcal{D}_i$ は，$\mathcal{X}_i$ に対する変数値の割り当ての集合である．

また，エージェント i の子のエージェント j のコスト情報は，$\mathcal{X}_{i,j} = \{x_i\} \cup \{vr_{a,j} | r_a \in R_j\}$ に対して評価される．したがって，i における変数 x_j を根とするサブツリーのコスト情報は，それぞれ $lb_i(j, \mathcal{D}_{i,j})$, $ub_i(j, \mathcal{D}_{i,j})$ および $context_i(j, \mathcal{D}_{i,j})$ のように変更される．

これらの変更の結果，仮想変数の数に関して，解の総数は指数関数的に増加する．そのため，探索における効率化手法が必要となる．

4.3.1 資源制約による枝刈り

エージェント i では，最適コストの境界 LB_i および UB_i を計算するために，$\mathcal{X}_i$ に対する探索処理が必要である．解の総数は $\mathcal{X}_i$ に含まれる仮想変数の数について指数関数的に増加する．しかし，資源制約により定義される境界により探索を枝刈りできる場合がある．

もし，仮想変数の値の割り当てが式 (6) に示される資源分配制約を満足しないとき，その割り当ては実行不可能解であり，そのコストは ∞ である．したがって，その割り当ては枝刈りされる．資源分配制約についての評価は，資源ごとに独立している．ある割り当てが資源 r_a についての資源分配制約に違反するならば，その割り当ては，たとえ r_a 以外の資源についての制約が満足されていても，違反解である．

4.3.2 子ノードのコスト情報

エージェント i の子のエージェント j のコスト情報は，それぞれ $lb_i(j, \mathcal{D}_{i,j})$, $ub_i(j, \mathcal{D}_{i,j})$ および $context_i(j, \mathcal{D}_{i,j})$ のように変更される．そのために必要な最悪の場合の記憶空間の大きさは，$\mathcal{X}_{i,j}$ に含まれる仮想変数の数について指数関数的に増加する．しかし，ADOPT アルゴリズムでは，子のエージェントからコスト情報が受信されていないときには，既定の初期値を持つコスト情報が用いら

れる．また，$current\_context_i$ が $context_{i,j}(j, \mathcal{X}_{i,j})$ と矛盾するときには，そのコスト情報は初期値に再初期化される．したがって，初期値を持つコスト情報は記憶する必要はない．コスト情報が初期値に再初期化されるとき，そのコスト情報は記憶から削除される．

4.3.3 資源制約から導出されるコストの上下界

提案手法は pseudo-tree に沿ってトップダウンに各資源の割り当て量を決定する．このような資源の割り当ては投機的である．また，エージェント i における，未探索の部分解 $\mathcal{D}$ について，コストの下界値 $LB_i(\mathcal{D})$ は 0 となる．そのため，ADOPT の最良下界優先探索では，未探索の資源の割り当てを網羅的に探索することが必要になる．4.3.1 節の資源制約による枝刈りは，実行不可能解を回避するのみであり，上記の探索空間を削減できない．

一方で，既にある資源の割り当てのもとで得られたコスト評価値の上下界は，他の未探索の資源の割り当てにおいても有効な場合がある．ここで，変数 x_i, 仮想変数 vr_a を含む割り当て $\mathcal{A}$ について，

$$\mathcal{A} \succ \mathcal{A}' \stackrel{\text{def}}{=} \tag{7}$$
$$(\forall u, u' \text{ s.t. } (vr_a, u) \in \mathcal{A}, (vr_a, u') \in \mathcal{A}',\ u \geq u') \wedge$$
$$(\forall d, d' \text{ s.t. } (x_i, d) \in \mathcal{A}, (x_i, d') \in \mathcal{A}',\ d = d')$$

のような資源割り当ての内包関係を記号 $\succ$ で表す．すなわち，$\mathcal{A}$ と $\mathcal{A}'$ は矛盾のない変数値の割り当てを含むが，資源の割り当ての量は，全ての資源について $\mathcal{A}$ が $\mathcal{A}'$ 以上であることを，$\mathcal{A} \succ \mathcal{A}'$ により表す．

これを用いて，資源割り当てにおけるコストの境界値の関係は次のようにあらわされる．

ある割り当て $\mathcal{A}$ においてコストの上界値 ub と下界値 lb が得られたとき，$\mathcal{A}' \succ \mathcal{A}$ のような割り当て $\mathcal{A}'$ においても ub は真の上界以上である．また，$\mathcal{A} \succ \mathcal{A}''$ のような割り当て $\mathcal{A}''$ においても lb は真の下界以下である．

上記の関係を各エージェント i の $ub_i(j, \mathcal{D}_{i,j}), lb_i(j, \mathcal{D}_{i,j})$ に適用することで，未探索の解の境界値を推定する．すなわち，未探索の $\mathcal{D}_{i,j}$ についての ub_i, lb_i の初期値を，それぞれ探索済の $\mathcal{D}_{i,j} \succ \mathcal{D}'_{i,j}, \mathcal{D}''_{i,j} \succ \mathcal{D}_{i,j}$ なる $D'_{i,j}, D''_{i,j}$ の ub_i, lb_i のうち，最も境界を狭くする値とする．

4·4 アルゴリズムの正当性と計算量

提案手法は，ADOPT に仮想変数を導入する．このため ADOPT の変数はベクトルに置き換えられる．この変更は ADOPT の単純な拡張となっている．各エージェントの持つ変数および各仮想変数についての部分解を，単一の変数であらわすことを考える．すなわち，変数の値域をあらわす離散集合および各仮想変数の値域をあらわす各離散集合すべての，直積として与えられる集合の要素を，ある一つの変数の値とみなす．その単一の変数について，ADOPT のコスト評価とそれに基づく処理は元の ADOPT と同様である．したがって，アルゴリズムの最適性，大域的終了については ADOPT と同様であり，

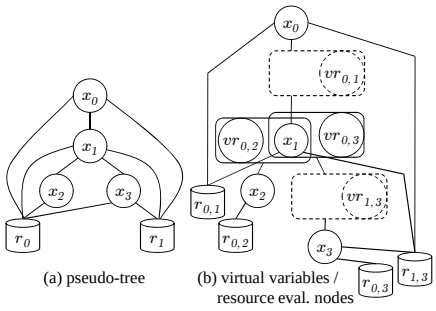


図 7 変数の直列化と仮想変数の併用

変数がベクトルの場合でも ADOPT の正当性に関する証明が成り立つ．また，提案手法は，資源制約が大域的に充足できないことを検出できる．そのとき，大域的コストは∞の値になる．

仮想変数の数に関して，解の総数は指数関数的に増加する．そのため各エージェントにおける最悪の場合の計算量は，仮想変数の数の増加に対して，指数関数的に増加する．また，メッセージ送受信回数の合計値も同様に，最悪の場合で指数関数的に増加する．しかし，4·3 節の効率化手法が有効な問題では比較的少ない探索コストで解が得られることが期待される．

5. 評　　価

5·1 問 題 の 設 定

提案手法の効果を実験により評価した．本研究で前提とする資源制約のモデルについての，一般的な評価用の例題はまだ存在しないと考えられる．そこで，初期の評価のためにグラフの三色彩色問題に資源制約を加えた問題を用いた．グラフの三色彩色問題および，その拡張された問題は，DCOP の解法の基礎的な評価のために用いられている [Modi 05, Ali 05, Silaghi 06, Farinelli 08]．特に，従来研究において，ADOPT アルゴリズムに対する結果が比較的多く得られていることから，本研究では三色彩色問題を基に例題を構成した．

問題はパラメータ (n, d, r, k) により生成した．また，パラメータにより決定される値 l, c, c_r を以下の説明で用いる．変数の数 n および制約辺の密度 d は，グラフ彩色問題の基本的なパラメータである．ここでは d は 1 または 2 とした．元のグラフ彩色問題では，この設定は疎な制約の問題を生成するために用いられる．各制約に対応する評価関数の値は，2 つの変数値が等しいとき 1，それ以外は 0 とした．さらに，以下に示す資源制約が加えられる．

資源制約はパラメータ r, k にもとづく．r は資源の数を表す．ここでは，単一の資源を全ての変数で共有する場合である $r = 1$，および複数の資源が複数の変数で共有される場合である $r = 4$ を用いた．資源の数 r は提案手法における仮想変数の個数に影響する．各資源が関連する変数の数，すなわち資源制約の arity を r, n に依存す

る値 l によりあらわす．各変数ノードは少なくとも一つの資源制約と関連するものとし，$l = \lceil n/r \rceil$ とした．

また，資源制約の充足可能性に関するパラメータ k により，資源の総容量 $c = \lceil n \times k \rceil$ とした．各資源の容量 c_r は，c を資源数 r で均等に分割した値に近い整数値とし，$c_r = \lceil l \times k \rceil \approx \lceil c/r \rceil$ により決定した．これらは，資源制約を充足する解が全ての解に占める割合が，k の値と正の相関があることを意図した．

従来手法は，最良優先探索戦略により選択された解が，資源制約を充足する確率の，影響を受ける．提案手法は，仮想変数を用いた探索において，資源制約を充足する実行可能解を列挙する際の枝刈りの効果の，影響を受ける．そこで，本実験では，資源制約を充足する解が全解空間に占める割合を r, k によりを変化するものとした．

各変数がある変数値をとるときに，関連する資源を要求する量は，各資源について 0 または 1 とした，すなわち，各変数の変数値について，関連する資源ごとに，1 単位量が要求されるか，全く要求されないかのいずれかである．特に，資源の要求量が 0 であることは，資源の容量によらず資源制約を充足する解を得る確率を増加することから，資源制約を充足する実行可能解を列挙する提案手法にとって，不利であると考えられる．この設定は，提案手法の欠点をより明確にすることを意図した．また，初期の検討として，資源が要求される場合の要求量を均一な値とした．各変数について，資源を要求する変数値が全値域に占める割合を $\frac{2}{3}$ とした. ただし，各例題では，少なくとも 1 つの解について大域的に資源制約を充足可能とするために，ランダムに選択した 1 つの解が資源制約を満足しない場合には，資源制約違反の原因となる変数値をランダムに選択し，その変数値の資源の使用量を 0 に強制した．実験では各設定の問題ごとに 10 個の例題を用いた．

次の各手法を用いる ADOPT について評価した．

- N: 変数の直列化を用いる従来手法.
- V: 仮想変数を用いる提案手法．ただし 4.3.3 節の効率化手法は未適用.
- M: 上記 N と V を併用する手法 (図 7)．この手法では，pseudo-tree において，資源制約で関連する変数が単一のパス上にある各区間の下端に，資源評価エージェント $(r_{0,1}, \cdots, r_{0,3}, r_{1,3})$ を配置する．複数の区間に資源を配分する必要がある変数 (x_1) のみ，仮想変数 $(vr_{0,2}, vr_{0,3})$ を導入する．仮想変数を介して各区間に割り当てられた資源について，その区間の資源評価エージェントが従来手法と同様に制約違反の評価を行う．これは，従来手法と提案手法の中間的な特性をもつ場合の評価のために構成した．
- VC: 上記 V に 4.3.3 節の枝刈りを適用した手法.
- MC: 上記 M に 4.3.3 節の枝刈りを適用した手法.

5·2 メッセージサイクル数

探索処理の性能を，最適解を得るまでのメッセージサイクル数により評価した．各変数エージェントはメッセージ送信キューと受信キューを持つ．各メッセージサイクルでは，まず，各エージェントは自身の受信キューにあるメッセージを処理し，必要に応じて送信キューにメッセージを格納する．全てのエージェントの処理が終わった後に，各ノードの送信キューの全てのメッセージを宛先の受信キューに移動し，次のメッセージサイクルを開始する．ここでは，各エージェント内の探索処理等は実装面の効率化が可能であり，計算時間よりもメッセージ通信時間の方が十分に大きいと仮定した．各実験は $T = 9999$ メッセージサイクルで強制的に中断した．その場合には最適解を得るまでに T メッセージサイクルかかったものとした．

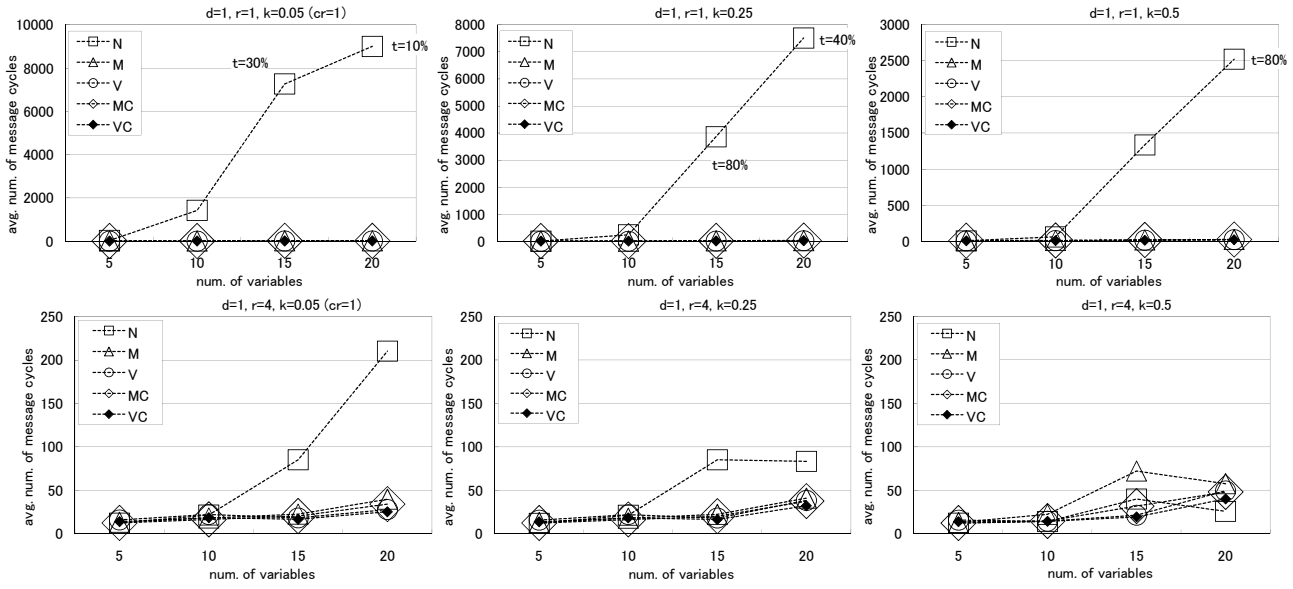
最適解を得るまでのメッセージサイクル数を図 8 に示す．$r = 1$ の場合，従来手法 N のメッセージサイクル数は提案手法 V より大きい．このとき，従来手法では pseudo-tree がほぼ線形となるため，ADOPT の探索処理において並列度が低下する．一方で，提案手法では pseudo-tree は資源制約を無視して生成されるため，ADOPT の探索処理は各サブツリーにおいて並列に実行される．

しかし，$d = 2, r = 4, k = 0.25$ および 0.5 の場合は，提案手法 V は従来手法 N よりも多くのメッセージサイクル数を要する．これらの問題においては，各ノードで複数の資源について仮想変数を生成する提案手法では，探索空間の増大の影響が大きくなる．

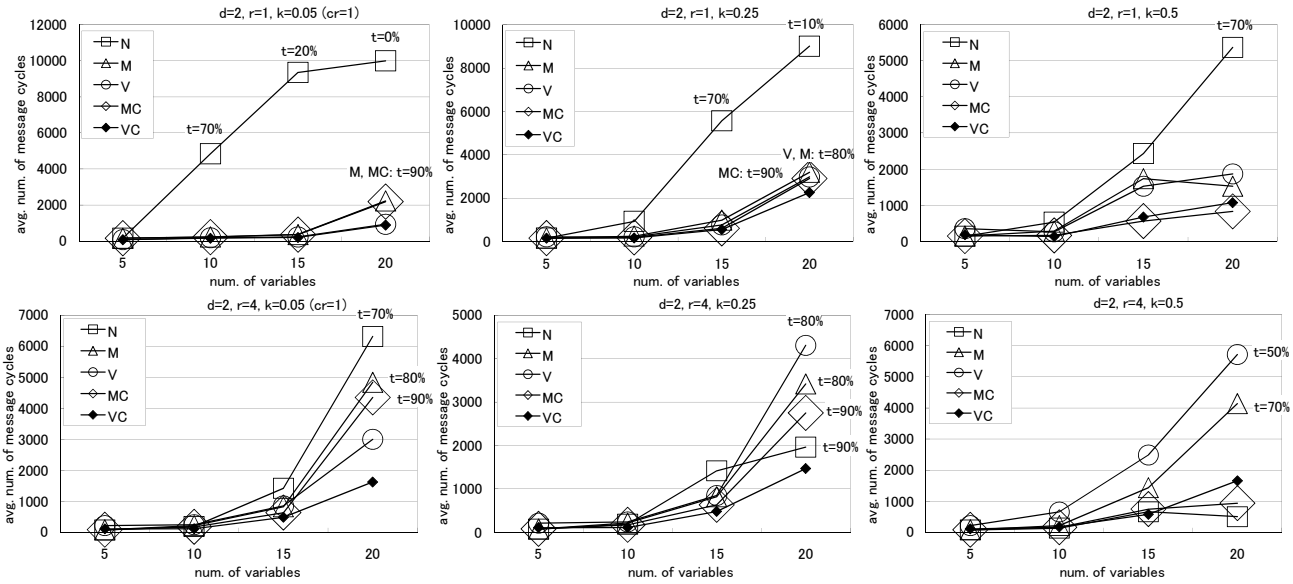
一方で，$d = 2, r = 4, k = 0.05$ の場合は，提案手法 V は従来手法 N よりもメッセージサイクル数が少ない．この問題の資源制約は，全解空間における実行可能解の範囲が極端に小さくなるよう設定されている．そのため，従来手法は∞のコスト値の部分解についての COST メッセージを多く生成し，メッセージサイクル数が増加する．これに対し，提案手法は資源分配制約の評価により主に実行可能解を探索するため，探索における効率的な枝刈りがなされると考えられる．

提案手法に，資源制約から導出される境界による枝刈りを併用した手法 VC では，資源の容量が比較的多い問題，および資源数が複数の問題でメッセージサイクル数が削減されている．$d = 2, r = 4, k = 0.5$ の場合を除いて，他の手法よりメッセージサイクルが少ないか同等程度となった．また，$d = 2, r = 4, k = 0.5$ の場合でも比較的大きなメッセージサイクル数の削減が得られている．

従来手法 N と提案手法 V を併用する手法 M は，N と V のいずれかに近い場合もあるが，ほぼ両者の中間的な結果となった．一方で，資源制約から導出される境界による枝刈りを併用した手法 MC では，VC と同様にメッセージサイクル数が削減されている．しかし，その削減の程度は $d = 2, r = 4, k = 0.05$ および 0.25 の場合では，VC よりも少ない．これは直列化部分の遅延の影響が除



(a) d=1



(b) d=2

図 8 メッセージサイクル数 (t: 探索がメッセージサイクル数の制限内で終了した割合，t を省略の場合は 100%終了)

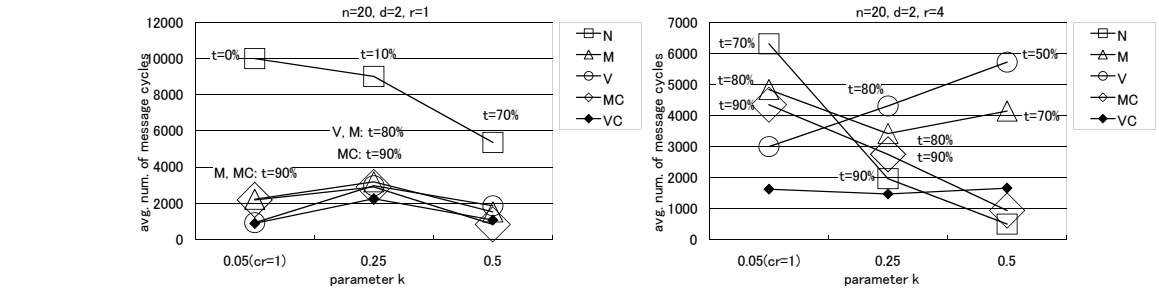
かれないためであると考えられる．

$n = 20, d = 2$ の問題について，資源の数 r または，資源制約の充足可能性に関するパラメータ k，を変化した場合のグラフを図 9 に示す．図中 (a) の $r = 1$ の場合を除いて，従来手法 N と提案手法 V は r, k の変化に対するメッセージサイクル数の変化が比較的顕著であり，その増減の関係が逆になっている．また手法 M は両者の中間的な性質を示している．提案手法 VC はいずれの場合も比較的少ないメッセージサイクル数となっている．

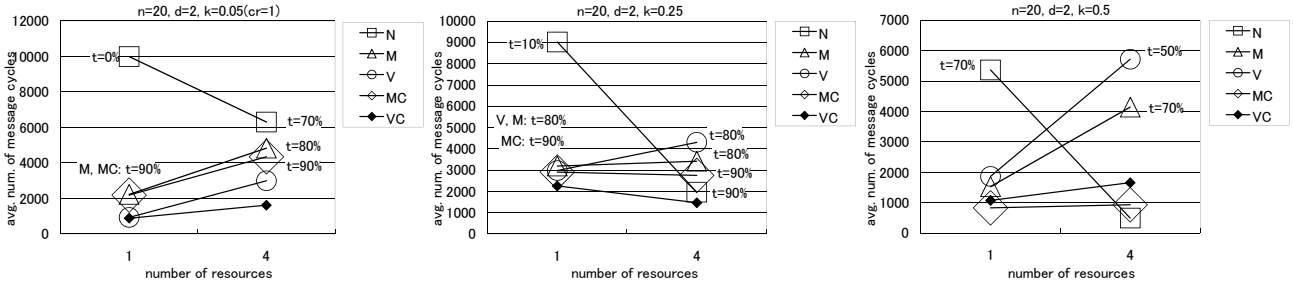
5·3 pseudo-tree のと部分解の規模，記憶されるコスト情報の数

生成された pseudo-tree の規模と各エージェントにおける部分解の次元の最大値を表 1 に示す．従来手法 N では，変数の多くが少数の資源と関係するとき，pseudo-tree の深さは大きい．提案手法 V では，各変数が多数の資源と関係するとき，各エージェントの部分解の次元は大きい．また，その次元は，同一の資源を要求するサブツリーについての分枝係数に応じて大きくなる．各エージェントが保持するコスト情報の数は，部分解の次元とともに増加する．また，手法 M では，必要最小限の仮想変数のみ生成されるため，部分解の次元が手法 V より少ない場合がある．

提案手法の各メッセージサイクルのスナップショットにおける，各エージェント i が記憶した子のエージェント j に対するコスト情報の数 (部分解 $\mathcal{D}_{i,j}$ の数) の，i,j についての総計の最大値を表 2 に示す．コスト情報の初期値を用いる場合には記憶が必要無いため，コスト情報の数は抑制されている．また，4.3.3 節の手法を適用することで，さらに抑制される．



(a) パラメータ k に対する変化



(b) 資源の数 r に対する変化

図 9 r, k に対するメッセージサイクル数の変化 (t: 探索がメッセージサイクル数の制限内で終了した割合，t を省略の場合は 100%終了)

表 1 pseudo-tree の規模と部分解の次元 (n=20)

d	r	cr	l	avg.max. depth of pseudo-tree		avg. branch. factor		avg.max. dim. of assign.	
				N	V	N	V	M	V
1	1	10	20	20.0	5.3	1.0	3.5	9.6	9.6
	4	3	5	10.8	5.3	1.2	3.5	12.9	13.0
2	1	10	20	20.0	11.2	1.0	1.5	3.7	3.7
	4	3	5	15.2	11.2	1.2	1.5	5.5	6.8

表 2 部分解に対するコスト情報の最大数 (d=2, n=20)

r	cr	l	avg. max. num. of cost information			
			V	VC	M	MC
1	1	20	56.8	55.2	52.6	51.8
4	1	5	137.1	86.3	69.7	66.3
1	5	20	118.5	103.8	95.8	84.7
4	2	5	332.3	118.8	92.8	73.9
1	10	20	176.9	131.7	129.1	103.6
4	3	5	559.4	147.8	102.5	77.2

5·4 資源制約に対する違反解の導出についての考察

提案手法は資源分配制約の評価により主に実行可能解を探索する．しかし，複数の資源がある場合，pseudo-tree に沿ったトップダウンな資源の配分の結果，あるエージェントでいずれの変数値を選択しても資源制約が充足できなくなる場合がある．このようなとき，資源制約についての違反解として，∞のコスト値を持つ部分解の情報が導出される．しかし本実験で提案手法がこのような違反解を導出することはきわめて少なく，大半の問題で 0 回であった．本実験の例題では，各変数の資源の要求量の最小値が 0 であり，これは資源を要求しないことを意味する．このため，多くの場合，違反解を導出することなく，実行不可能解を回避しているものと考えられる．このような問題の性質は，提案手法にとって不利な状況の一面を表すものであると考えられる．すなわち，この問題では 4.3.1 の資源制約にもとづく枝刈りにおいて，トップダウンな資源の配分の情報のみでしか枝刈りが行われない．4.3.3 の手法はこのような場合の枝刈りの効果を改善するためにも有効であると考えられる．

6. むすび

本論文では，資源制約を無視して生成された pseudo-tree を用いる，資源制約付き分散制約最適化問題の解法を提案した．提案手法では，pseudo-tree における異なるサブツリーと関係する資源制約を許容するために，資源の使用量を表現する仮想変数を導入した．一方で，仮想変数は探索空間を増大させるため，資源制約により定義される仮想変数値の境界により探索を枝刈りした．提案手法の効果は問題の性質に依存することを実験結果により示した．多数の変数が少数の資源と関係する問題，および実行可能解が少ない問題では，提案手法が従来手法よりメッセージサイクル数において効率的であった．さらに，実行可能な解空間が大きい問題への対策として，資源制約から導出されるコストの境界値による枝刈りを用いることで，メッセージサイクル数が削減されることを実験結果により示した．

提案手法は，pseudo-tree を用いる他の解法 [Petcu 05, Petcu 06] にも適用可能である．提案手法に適した資源制約と pseudo-tree についての詳細な解析および，より高度な枝刈りのための境界値の導入は今後の課題である．

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 19300048) 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)

[Bowring 06] Bowring, E., Tambe, M., and Yokoo, M.: Multiply constrained distributed constraint optimization, in *5th International Joint Conference on Autonomous Agents and Multiagent Systems*, pp. 1413–1420 (2006)

[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)

[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)

[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)

[Pecora 06] Pecora, F., Modi, P., and Scerri, P.: Reasoning About and Dynamically Posting n-ary Constraints in ADOPT, in *7th International Workshop on Distributed Constraint Reasoning, at 5th International Joint Conference on Autonomous Agents and Multiagent Systems* (2006)

[Petcu 05] Petcu, A. and Faltings, B.: A Scalable Method for Multiagent Constraint Optimization, in *9th International Joint Conferece on Artificial Intelligence*, pp. 266–271 (2005)

[Petcu 06] Petcu, A. and Faltings, B.: O-DPOP: An algorithm for Open/Distributed Constraint Optimization, in *National Conference on Artificial Intelligence*, pp. 703–708 (2006)

[Silaghi 06] Silaghi, M. C. and Yokoo, M.: Nogood based asynchronous distributed optimization (ADOPT ng), in *5th international joint conference on Autonomous agents and multiagent systems*, pp. 1389–1396 (2006)

〔担当委員：伊藤 孝行〕

2009 年 1 月 12 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. 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 年より 九州大学大学院 システム情報科学研究院 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, 1999 年, 2005 年, 2008 年 人工知能学会全国大会優秀論文賞, 2004 年 ACM 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 受賞. 電子情報通信学会, 情報処理学会, 日本ソフトウェア科学会, AAAI 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）. 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.