論　文

擬似的な木を用いた分散制約最適化手法のための木の幅を考慮した空間複雑度の削減

松井　俊浩[†]　　松尾　啓志[†]

Methods for reducing space complexity in distributed pseudo tree optimization

Toshihiro MATSUI[†] and Hiroshi MATSUO[†]

あらまし　分散制約最適化問題は分散制約充足問題を最適化問題に拡張したものとして，マルチエージェントシステム，分散協調問題解決の基礎的な分野として研究されている．近年，制約網に対する深さ優先探索木などの擬似的な木 (pseudo-tree) による変数順序付けと動的計画法に基づく分散制約最適化手法が提案された．本論文では，この分散制約最適化手法の補助的な手段として，擬似的な木の幅が制限される値を超える部分について各エージェントが必要とする記憶およびメッセージのサイズについての空間複雑度を制限するための基本的な手法を提案する．一部の変数について部分的にバックトラックを導入する手法，および，貪欲戦略の一つとして一部の変数の値を予め固定する手法を示す．また，提案手法の有効性について評価する．

キーワード　分散制約最適化問題，分散制約充足問題，マルチエージェント，擬似的な木

1. まえがき

分散制約最適化問題 [1]〜[6] は分散制約充足問題 [11] を最適化問題に拡張したものとして，マルチエージェントシステム，分散協調問題解決の基礎的な分野として研究されている．

近年，制約網に対する深さ優先探索木などの擬似的な木 (pseudo-tree) による変数順序付けを用いた解法が提案されている [5] [9] ．制約網に対する深さ優先探索木などの擬似的な木は，各部分木に配置された変数の間に制約辺が無いため，問題の持つ並列性を利用した探索の効率化が得られる．

このような擬似的な木を用いる，メモリ制限のある A*アルゴリズムに基づく非同期分散最適化手法が提案された [5]．この解法はメモリを多項式のオーダーに制限し，非同期な分枝限定法を主体とするために，メッセージ数が多い．

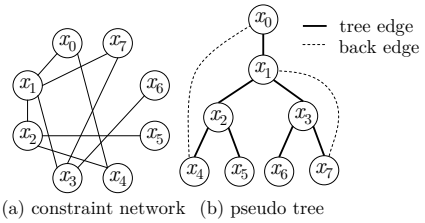
これに対し，特に擬似的な木の幅が小さい場合については動的計画法により，木の深さについて多項式時間で解を得ることができる [9]．通信のオーバヘッドを考慮した場合，自由度の高い非同期バックトラックに基づく手法よりも，メッセージ数を削減できる動的計画法を主体とする手法は有効である．

しかし，このような手法では，各ノード (エージェント) が必要とする記憶およびメッセージのサイズについての空間複雑度は，各ノードを根とするサブツリーから制約辺で関連する上位ノードの数によって示される，擬似的な木の幅に対して指数関数的に増加するため，適用可能な問題は制限される．制約密度が小さい問題に対して擬似的な木の幅は小さくなる傾向があるが，一般的な問題では木の幅を一定値以下にすることは厳密には難しい場合があると考えられる．

本論文では，動的計画法に基づく解法 [9] の補助的な手段として擬似的な木の幅が制限される値を超える部分について，各エージェントが必要とする記憶およびメッセージのサイズにおける空間複雑度を削減するために一部の変数について部分的にバックトラックを導入する手法を示す．また，貪欲戦略の一つとして一部の変数の値を予め固定する手法を示す．

以下では，先ず 2. で分散制約最適化問題について述べ，3. で従来手法として，制約網に対する擬似的な木を用いた動的計画法に基づく分散探索アルゴリズムについて述べる．4. で提案手法として，従来手法に擬似

[†] 名古屋工業大学，名古屋市
Nagoya Institute of Technology, Gokiso-cho,Showa-ku,Nagoya-shi, 466-8555,Japan



(a) constraint network (b) pseudo tree

図 1 制約網と擬似的な木
Fig. 1 Constraint network and pseudo tree

的な木の幅を考慮した記憶のサイズの制限とバックトラックを導入する手法を示す．5. では計算機実験による評価を示す．6. では関連研究および考察を示す．

2. 分散制約最適化問題

分散制約最適化問題は，分散制約充足問題を組み合わせ最適化問題に拡張した問題として形式化される．

ノード (エージェント) の集合を A とする．各ノード $i \in A$ は変数 x_i を持つ．変数 x_i は，x_i の値域 D_i の値 d_i をとる．以下では必要に応じて，ノード i と変数 x_i を区別せずに用いる．

C を制約の集合，F を制約に対する評価関数の集合とする．x_i は他ノード j の変数 x_j と 2 項制約 $c_{i,j} \in C$ により関係する．制約 $c_{i,j}$ に対応する評価関数 $f_{i,j} \in F$ により，変数値の割り当て $\{(x_i, d_i), (x_j, d_j)\}$ についてのコストが評価される．大域コストは全ての制約に対するコストの評価の総和であり，大域コストを最小化する変数値の割り当てが最適解である．

各ノードは任意の他のノードとの間に FIFO のメッセージ通信リンクを持つ．異なる通信リンク間のメッセージの順序は保存されない．また，ノードおよび通信リンクの故障は考慮しない．

3. 従来手法 — 制約網に対する擬似的な木を用いた動的計画法に基づく解法

3.1 制約網に対する擬似的な木

制約網に対する深さ優先探索木などの擬似的な木 (pseudo-tree) は，各部分木に配置された変数の間に制約辺が無い性質があり，探索手法の効率化に利用できることが知られている．図 1 に制約網と擬似的な木の例を示す．各ノード (変数) は木によって順序付けられる．ノード x_i がノード x_j の親ノードまたは祖先ノードであることを，記号 $\succ$ を用いて $x_i \succ x_j$ により表す．制約辺は，木の枝となる木の辺 (tree edge) とそれ以外の後退辺 (back edge) に分類される．以下では，ノード x_i と制約辺で関連するノードを次の表記により示す．

- $P(x_i)$：木の辺で x_i と関連する親ノード
- $PP(x_i)$：後退辺で x_i と関連する祖先ノードの集合
- $C(x_i)$：木の辺で x_i と関連する子ノードの集合
- $PC(x_i)$：後退辺で x_i と関連する子孫ノードの集合

さらに，表現の便宜のために次の集合を用いる．

- $\overline{PP}(x_i)$：x_i を根とする部分木に含まれるいずれかのノードと後退辺で関連する x_i の祖先ノード，および $P(x_i)$ からなる集合

すなわち，$\overline{PP}(x_i)$ は，x_i を根とする部分木に含まれるノードについて制約を評価する際に関係する，x_i の上位ノードの集合である．

$|\overline{PP}(x_i)|$ を x_i における擬似的な木の幅とし，$W(x_i)$ により表す．たとえば，図 1(b) のノード x_2 の場合は，祖先ノード x_0 と x_2 を根とする部分木に含まれるノード (子ノード)x_4 が後退辺で関連する．これに親ノード $P(x_2) = x_1$ を加えると，$\overline{PP}(x_2) = \{x_0, x_1\}$ であり，$W(x_2) = |\overline{PP}(x_2)| = 2$ である．一方，図 1(b) のノード x_3 の場合は，$P(x_3) = x_1$ であり，また，x_1 と子ノード x_7 が後退辺で関連する．したがって，$\overline{PP}(x_3) = \{x_1\}$ であり，$W(x_3) = |\overline{PP}(x_3)| = 1$ である．また，全ての x_i についての $W(x_i)$ の最大値を擬似的な木の全体についての幅とする．図 1(b) の擬似的な木全体の幅は $W(x_2) = 2$ と等しい．

3.2 擬似的な木に対する動的計画法の適用

一般的な組み合わせ最適化問題では，動的計画に必要な記憶のサイズは指数関数的に増大する場合があるため，記憶のサイズの制限は本質的に必要である．一方で，バックトラックの回数を抑制するために動的計画法を適用することは有効である．特に制約密度が小さく擬似的な木の幅が小さい問題は，動的計画法のみによる解法を適用できる．以下では動的計画法による基本的な解法を示す．この解法では前処理により，(1) 擬似的な木が既に生成されていて，(2) 上位ノードの値域を下位の近傍ノードが予め知る，ものとする．

制約網が木であるときは，後退辺が存在しない擬似的な木を生成できる．このような問題について，木の深さに対する多項式オーダの時間複雑度，値域に対する多項式オーダの空間複雑度の動的計画法による解法を適用できる [8]．これは次のような 2 段階の処理により構成される．

(1) コストの計算

各ノード x_i は，親ノード $P(x_i)$ の全ての変数値 $d_{P(x_i)} \in D_{P(x_i)}$ について，x_i を根とする部分木のコストのベクトル $Cost_{x_i,P(x_i)}$ を次式により計算する．この式は木の葉から根に向けてボトムアップに計算される．

$$Cost_{x_i,P(x_i)}(d_{P(x_i)}) = \begin{cases} \min_{d_i \in D_i} f_{i,P(x_i)}(d_i, d_{P(x_i)}) & C(x_i) = \phi \\ \min_{d_i \in D_i}\{f_{i,P(x_i)}(d_i, d_{P(x_i)}) & \text{otherwise} \\ \quad + \sum_{x_k \in C(x_i)} Cost_{x_k,x_i}(d_i)\} & \end{cases} \tag{1}$$

(2) 最適解の決定

コストの計算の後， 各ノード x_i は，最適な変数値 d_i^* を次式により決定する．この式は木の根から葉に向けてトップダウンに計算される．

$$d_i^* = \begin{cases} \mathrm{argmin}_{d_i \in D_i} f_{i,P(x_i)}(d_i, d^*_{P(x_i)}) & C(x_i) = \phi \\ \mathrm{argmin}_{d_i \in D_i}\{f_{i,P(x_i)}(d_i, d^*_{P(x_i)}) & \text{otherwise} \\ \quad + \sum_{x_k \in C(x_i)} Cost_{x_k,x_i}(d_i)\} & \end{cases} \tag{2}$$

一方，一般的な制約網に対する擬似的な木には後退辺が存在する．このため，動的計画法を適用するためには，各ノード x_i について，親ノード $P(x_i)$ の変数の値域だけではなく，ノード x_i を根とする部分木と後退辺で関連する全ての上位ノード $\overline{PP}(x_i)$ の変数の値域の組に対するコストの計算が必要になる．このため x_i で計算されるコスト $Cost_{x_i,P(x_i)}$ は hyper-cube になる．たとえば，図 1(b) のノード x_2 の場合は，$Cost_{x_2,P(x_2)}$ は $s \in D_0 \times D_1$ に対するコストを表す．一方，ノード x_3 の場合は，$Cost_{x_3,P(x_3)}$ は $s \in D_1$ に対するコストを表す．

擬似的な木について一般化された動的計画法に基づく解法である DPOP [9] の手続きを図 2 に示す．(注1) 図 2 の手続きは式 (1)(2) で示した，コストの計算，最適解決定，の 2 段階からなる処理を，上述の hyper-cube を用いて一般化したものである．また，図 2(A) における $\overline{PP}(x_i)$ は，ノード x_i が知る $P(x_i)$,$PP(x_i)$, および x_i が全ての子ノード x_k から受信したコスト情報 $Cost_{x_k,x_i}$ に含まれるノードからなる集合，の和集合として求められる．

(注1)：文献 [9] では利得をを最大化する問題として形式化されているが，本論文ではコストを最小化する問題として形式化した．また，便宜上アルゴリズムの表記を変更した．

[**Initialize**]
  $agentview_i \leftarrow \phi$;
  if x_i is a leaf node **then**
    compute_Cost($Cost_{x_i,P(x_i)}$**)**;
    send (**COST**, x_i, $Cost_{x_i,P(x_i)}$) to $P(x_i)$; **endif**
[**COST_message_handler(COST**, x_k, $Cost_{x_k,x_i}$)]
  store $Cost_{x_k,x_i}$;
  if COST messages arrived from all x_k in $C(x_i)$ **then**
    if x_i is the root node **then**
      choose_optimal(d_i^***)**;
      send (**VALUE**, x_i, $\{(x_i, d_i^*)\}$) to all $C(x_i)$;
    else compute_Cost($Cost_{x_i,P(x_i)}$**)**;
      send (**COST**, x_i, $Cost_{x_i,P(x_i)}$) to $P(x_i)$;
    endif endif
[**VALUE_message_handler(VALUE**, x_j, $agentview_j$)]
  $agentview_i \leftarrow agentview_j$;
  choose_optimal(d_i^***)**;
  send (**VALUE**, x_i, $agentview_i \cup \{(x_i, d_i^*)\}$)
    to all $C(x_i)$;
[**compute_Cost(var** $Cost_{x_i,P(x_i)}$**)**]
  $\overline{X} \leftarrow \overline{PP}(x_i)$; (A)
  foreach $\overline{d} \in \prod_{x_j \in \overline{X}} D_{x_j}$ **do**
  $Cost_{x_i,P(x_i)}(\overline{d}) \leftarrow \min_{d_i \in D_i}\{f_{i,P(x_i)}(d_i, d_{P(x_i)}) +$
    $\sum_{x_j \in PP(x_i)} f_{i,j}(d_i, d_{x_j}) +$
    $\sum_{x_k \in C(x_i)} Cost_{x_k,x_i}((\overline{d}, d_i))\}$;
[**choose_optimal(var** d_i^***)**]
  $d_i^* \leftarrow \mathrm{argmin}_{d_i \in D_i}\{f_{i,P(x_i)}(d_i, d^*_{P(x_i)}) +$
    $\sum_{x_j \in PP(x_i)} f_{i,j}(d_i, d^*_{x_j}) +$
    $\sum_{x_k \in C(x_i)} Cost_{x_k,x_i}((agentview_i, d_i))\}$;

図 2　DPOP アルゴリズム
Fig. 2　DPOP Algorithm

この解法は，後退辺が無い場合と同様に，木の深さの 2 倍のメッセージサイクルと辺の数の 2 倍のメッセージ数を要する．一方で，擬似的な木の幅と変数の値域が増加するに従い，各ノードが必要とする記憶とメッセージのサイズについて空間複雑度が指数関数的に増大する．従って，問題および，木の構築の際に，擬似的な木の幅を制限することが必要になる．このために制約密度が小さい問題に制限することや木の生成の際に適切な手法を用いることが考えられるが，一般

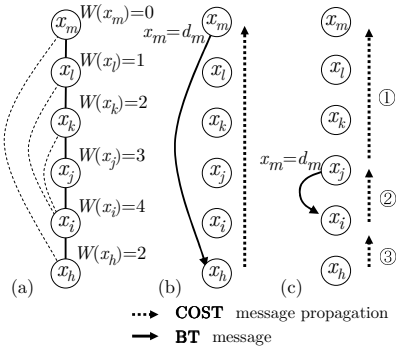


図 3　部分的なバックトラッキング
Fig. 3　Partial backtracking

的な問題では擬似的な木の幅を一定値以下にすることは厳密には難しい場合があると考えられる．

4. 提案手法 — 擬似的な木の幅を考慮した記憶のサイズの制限

本論文の提案手法である，擬似的な木の幅を考慮した，各ノードの必要とする記憶およびメッセージのサイズの削減手法について述べる．提案手法は，図 1(b) のように，擬似的な木が複数の隣り合う部分木を持つ一般的な場合を対象とする．ただし，ここでは説明を簡単にするために，図 3(a) に示す線形なノード順序の擬似的な木を用いる．

4.1 擬似的な木の幅を考慮した記憶の制限とバックトラッキングの導入

記憶のサイズを削減する方法として，一部の制約辺について上位の変数値を固定して探索空間を削減することが考えられる．たとえば図 3(a) では変数 x_i について幅 $W(x_i) = 4$ であり，**COST** メッセージのコスト情報の hyper-cube は $D_m \times D_l \times D_k \times D_j$ であるが，変数 x_m を定数値 d_m とすることで，hyper-cube は $\{d_m\} \times D_l \times D_k \times D_j$ となり，そのサイズは幅 $W(x_i) = 3$ の場合と等しくなる．しかし，網羅的な最適解の探索のためには変数値の変更が必要であるため，固定された変数値をコストの集計後に変更する．すなわち，一部の変数についてバックトラックを導入する．たとえば図 3(b) のように，現在の x_m の値についての **COST** メッセージによるコストの集計ごとに，**BT** メッセージによって x_m の値 d_m の変更を通知し，さらに **COST** メッセージによるコストの集計を行う．この処理は $|D_m|$ 回反復される．

4.1.1 バックトラックの対象とする変数の決定

バックトラックの対象とする変数は前処理によって

[**Intialize**]
  $X_i^{all} \leftarrow P(x_i) \cup PP(x_i)$; $X_i^{bt} \leftarrow \phi$, $f_i^{bt} \leftarrow$ **false**;
  if x_i is a leaf node **then**
    send (**PRE**, x_i, X_i^{all}, X_i^{bt}) to $P(x_i)$; **endif**
[**PRE_message_Handler**(**PRE**, x_j, X_j^{all}, X_j^{bt})]
  $X_i^{all} \leftarrow X_i^{all} \cup X_j^{all}$; $X_i^{bt} \leftarrow X_i^{bt} \cup X_j^{bt}$;
  if PRE messages arrived from all x_j in $C(x_i)$
  then
    remove x_i from X_i^{all};
    if x_i is included in X_i^{bt} **then**
      remove x_i from X_i^{bt}; $f_i^{bt} \leftarrow$ **true**; **endif**
    $X \leftarrow X_i^{all}$; $W \leftarrow |X_i^{all} - X_i^{bt}|$
    while $W > \overline{W}$ **do** (A)
      select x_k
        where $x_k, x_l \in X \wedge x_k \neq x_l \wedge \forall x_l, x_k \succ x_l$;
      $X_i^{bt} \leftarrow X_i^{bt} \cup \{x_k\}$;
      remove x_k from X; $W \leftarrow |X_i^{all} - X_i^{bt}|$
    enddo
    send (**PRE**, x_i, X_i^{all}, X_i^{bt}) to $P(x_i)$;
  endif

図 4　前処理 — バックトラックの対象とする変数の決定
Fig. 4　Preprocessing for backtracking variables

選択する．擬似的な木の幅についての制限値 $\overline{W}$ をパラメータとし，$W(x_i) > \overline{W}$ となる場合にその原因となる一部の上位ノードの変数をバックトラックの対象とする．このようなバックトラックの対象とする変数は一意には決定はできない．特に，バックトラックの対象となる変数の決定には，オーバヘッドの小さい分散処理を用いる必要がある．また，図 1(b) のような一般的な擬似的な木では，隣り合う複数の部分木について重複するバックトラックの範囲を考慮する必要がある．そこで，本論文では貪欲な手法の一つとして，より上位の変数をバックトラックの対象として優先的に選択する手法を用いる．これはバックトラックの対象とする変数を各ノードで共通にし，より広範囲の木の幅を削減することを意図する．

前処理の概要を図 4 に示す．図中で用いられる主要な変数の用途は次のとおりである．

- X_i^{all}: ノード x_i についての $\overline{PP}(x_i)$ の集計．
- X_i^{bt}: ノード x_i の上位ノードでバックトラックの対象となる変数の集合の集計．
- f_i^{bt}: ノード x_i がバックトラックの対象の変数であるか否かを表すフラグ．

この処理は擬似的な木に従って葉ノードから順にボトムアップに実行される．x_i が葉ノードのとき，x_i は，**PRE** メッセージによって速やかに，X_i^{all},X_i^{bt},f_i^{bf} を親ノード $P(x_i)$ に通知する．各ノード x_i は，全ての子ノード x_k から受信した X_k^{all} および，$P(x_i)$,$PP(x_i)$ により X_i^{all} を求め，$X_i^{all} = \overline{PP}(x_i)$ となる．このとき $|X_i^{all}| = |\overline{PP}(x_i)| = W(x_i)$ である．

その後，$W(x_i) > \overline{W}$ である場合は，上位ノードの変数をバックトラックの対象として選択し，X_i^{bt} に追加する．子ノード x_k から通知された X_k^{bt} が x_i を含むとき，受信処理の結果として X_i^{bt} も x_i を含む．このとき，$f_i^{bt} = true$ として x_i がバックトラックの対象であることを記録し，X_i^{bt} からは x_i を削除する．また，図 4 の処理と同時に，各ノード x_i について，x_i と制約で関連する子孫ノード $C(x_i) \cup PC(x_i)$ のうち，各部分木の最下位に位置するノードからなる集合 $PC^{lowest}(x_i)$ を求める．

4.1.2 バックトラックを導入した DPOP

バックトラックを導入した DPOP アルゴリズムを図 5 に示す．この手法では，変数 x_i がバックトラックの対象であるときノード i は現在の変数値 d_i を **BT** メッセージにより $PC^{lowest}(x_i)$ に含まれるノードに通知する．また，バックトラック対象でない変数値は $*$ により表し，初回のみ **BT** メッセージにより通知する．

ノード x_i が **BT** メッセージを受信したとき，$agentview_i$ に現在の上位ノードの変数値を記録する．また，**BT** メッセージで直接通知されない変数値は，**COST** メッセージにより子孫ノードから伝搬される．これにより $agentview_i$ には現在の祖先ノードの変数値が記録される．$agentview_i$ に含まれる変数値が $*$ では無い変数については，変数値を固定し，$*$ である変数については従来手法と同様に各値域の組のコストの計算を行う．たとえば，図 3(b) では，x_i について，$agentview_i = \{(x_m, d_m), (x_l, *), (x_k, *), (x_j, *)\}$ であり，$Cost_{x_i,P(x_i)}$ は $D_l \times D_k \times D_j$ について計算される．

これまでに子ノード x_k から受信した ($agentview_k$, $Cost_{x_k,x_i}$) について，$agentview_k$ が $agentview_i$ と矛盾するときは，その ($agentview_k$, $Cost_{x_k,x_i}$) は削除される．ここで，$agentview_k$ が $agentview_i$ について無矛盾であるとき，

$$\forall x_j,\ (x_j, d_j^k) \in agentview_k, (x_j, d_j^i) \in agentview_i,\quad d_j^k = * \vee d_j^k = d_j^i$$

[**Initialize**]
  $agentview_i \leftarrow \phi$; $value\_arrived \leftarrow$ **false**;
  if f_i^{bt} **then** $\overline{D}_i \leftarrow D_i$; select d_i from $\overline{D}_i$;
  remove d_i from $\overline{D}_i$; **else** $d_i \leftarrow *$; **endif**
  send (**BT**, x_i, d_i) to all $PC^{leaf}(x_i)$;
[**BT_message_handler**(**BT**, x_j, d_j)]
  if not $value\_arrived$ **then**
   add (x_j, d_j) to $agentview_i$; **endif maintain()**;
[**COST_message_handler**(**COST**, x_k, $agentview_k$, $Cost_{x_k,x_i}$)]
  if not $value\_arrived$ **then**
    replace $agentview_i$ by $agentview_k$; **endif**
  store ($agentview_k$, $Cost_{x_k,x_i}$);
  maintain();
[**VALUE_message_handler**(**VALUE**, x_j, $agentview_j$)]
  $agentview_i \leftarrow agentview_j$;
  $value\_arrived \leftarrow$ **true**; **maintain()**;
[**maintain()**]
  forall stored ($agentview_k$, $Cost_{x_k,x_i}$) **do**
   if $agentview_k$ is incompatible with $agentview_i$
    then remove ($agentview_k$, $Cost_{x_k,x_i}$);
     endif enddo
  if COST messages arrived from all x_k in $C(x_i)$
   then if f_i^{bt} **then**
      compute partial $Cost_{x_i,P(x_i)}$ for d_i; (A)
      if $\overline{D}_i \neq \phi$ **then** select d_i from $\overline{D}_i$;
        remove d_i from $\overline{D}_i$;
        send (**BT**, x_i, d_i) to all $PC^{leaf}(x_i)$;
      endif endif
    if $value\_arrived$ **or** x_i is the root node
    then choose_optimal(d_i^*);
      if f_i^{bt} **then**
        send (**BT**, x_i, d_i^*) to all $PC^{leaf}(x_i)$; **endif**
      send (**VALUE**, x_i, $agentview_i \cup \{(x_i, d_i^*)\}$)
        to all $C(x_i)$;
    else compute_Cost($Cost_{x_i,P(x_i)}$); (B)
      send (**COST**, x_i, $agetview_i$, $Cost_{x_i,P(x_i)}$)
        to $P(x_i)$; **endif endif**

図 5　DPOP とバックトラックの併用 (DPOP+BT1)
Fig. 5　DPOP with partial backtracking (DPOP+BT1)

である．このような $agentview_i$ の使用法はバックトラックを用いる分散最適化手法である ADOPT [5] で

用いられる current context を変数値が $*$ で表される場合について一般化したものである．

コストの計算 **compute_Cost()** および最適解の決定 **choose_optimal()** は一部の変数値が固定されることを除いて DPOP と同様である．たとえば，変数 x_l がバックトラックの対象であり，現在の値が d_l であるとき，x_l に関連する下位のノード x_i における $Cost_{x_i,P(x_i)}$ は，変数 x_l について値域が $D_l = \{d_l\}$ である場合と同様になる．

また，バックトラックの対象である変数 x_l のノードでは，**COST** メッセージによるコストの集計は現在の変数値 $d_l \in D_l$ ごとに行われる．このため，図 5(A) の箇所で各 d_l についての $Cost_{x_l,P(x_l)}$ を求めておき，最終的に図 5(B) の箇所で $\forall d \in D_l$ について統合された $Cost_{x_l,P(x_l)}$ が得られているものとした．また，再計算を削減するために，$Cost_{x_l,P(x_l)}$ の各要素に対応する変数値 d_l^* も記録しておくものとした．

4.2 バックトラックの範囲の削減

4.1 では，バックトラックの対象とする変数を決定する簡単な手法として，より上位の変数をバックトラックの対象として優先的に選択する手法を導入した．しかし，このような手法では，バックトラックの際に再計算が行われる範囲が冗長な広範囲に及ぶ可能性がある．たとえば，根ノードと，最も深い葉ノードとの間に制約辺がある場合に，根ノードの変数を変更すると，これらの 2 ノード間でコストの再計算が必要になる．一方，擬似的な木の幅が制限値を越える場所はそれらの中間の部分のみの場合がある．

このような冗長性を削減し，擬似的な木の幅が制限値を超えた部分のみバックトラックを適用するために，さらに，以下の 2 つの方法を用いる．

(1) 擬似的な木の幅 $W(x_i)$ が制限値を超えないノード x_i では，上位の変数 x_l がバックトラックの対象となる場合でも x_l の全ての値域についてのコストの集計を行う．また，その上位で擬似的な木の幅 $W(x_j)$ が制限値を超えたノード x_j について，x_j を $PC^{lowest}(x_l)$ に追加し，x_j を x_l からの **BT** メッセージを受信するノードとする．すなわちバックトラックを行う区間の下側を擬似的な木の幅が制限値を超えるノードまで上位に移動する．

(2) 擬似的な木の幅 $W(x_i)$ が制限値を超えないノード x_i では，ノード x_i を根とするサブツリーに，x_i の上位ノード x_l の **BT** メッセージを受信するノードがある場合，x_i は x_l の代わりにバックトラック処理を行う．すなわちバックトラックを行う区間の上側を擬似的な木の幅が制限値を超える直前ノードまで下位に移動する．

たとえば図 3(b) では，$\overline{W} = 3$ の制限値に応じて，x_m をバックトラックの対象としているが，さらに，図 3(c) のように，$W(x_i) > \overline{W}$ となる部分のみにバックトラックの範囲を制限するために，x_m の変数値の変更を x_j で行い，x_m の変数値の受信を x_i で行う．これにより，図 3(c) の①③の部分では，コストの計算のための **COST** メッセージの伝播は，1 回のみ実行される．一方，②の部分では $|D_m|$ 回の反復的な計算を行う．

このために 4.1 の手法を拡張する．前処理では，図 4(A) の $W > \overline{W}$ の場合に X_i^{bt} に変数を追加する処理に加えて，変数を除去する処理を行う．すなわち，$W < \overline{W}$ の場合は，X_i^{bt} に含まれる変数で最下位のものを除去する．また，除去した変数については，ノード x_i でバックトラック処理を行う．すなわち図 5 におけるバックトラックの対象の変数についての処理と同様に，**BT** メッセージの送信および，各変数値ごとの **COST** メッセージによるコストを統合する処理を行う．

4.3 近似解法としての利用

上記の手法は単純なバックトラックに基づく網羅的な探索となるため，バックトラックの対象となる変数の増加によりメッセージサイクル数が指数関数的に増加する．このため，実際的には，ある程度の解の質の劣化を許容したとしても，速やかに解を得ることが必要になる．上記の手法は，バックトラックの対象である変数値の変更回数を制限することで，近似解法とすることができる．本論文では特に簡単な場合として，4.1 の手法でバックトラックの対象として選択された変数の値を初期値のままで固定する貪欲戦略について検討する，

このような初期値の選択についての基準は問題に依存すると考えられる．本論文では，擬似的な木に配置された各変数 x_i の変数値 d_i についてのコストの下限値を求める前処理 [7] により得られるコスト値

$$h(d_i) = \sum_{x_k \in C(x_i)} \{ \min_{d_k \in D_k} \{h(d_k) + f_{k,i}(d_k, d_i)) + \sum_{x_j \in PC(x_k)} \min_{d_j \in D_j} f_{k,j}(d_k, d_j)\}\} \quad (3)$$

を用い，$h(d_i)$ が最小となる変数値を選択した．

表 1　擬似的な木の深さの平均
Table 1　Average depth of pseudo-tree

c \ n	25	50	100	200
n-1	6	7	9	11
1.125n	9	15	23	44
1.25n	11	18	32	59
1.5n	12	22	43	77

表 2　擬似的な木の幅の平均
Table 2　Average width of pseudo-tree

c \ n	25	50	100	200
n-1	1	1	1	1
1.125n	3	4	6	11
1.25n	4	6	10	20
1.5n	5	10	18	33

表 3　平均サイクル数 (DPOP)
Table 3　Average number of cycles (DPOP)

c \ n	25	50	100	200
n-1	11	14	17	20
1.125n	17	29	46	–
1.25n	21	36	–	–
1.5n	23	44	–	–

表 4　**COST** メッセージのサイズの総和の平均 (DPOP)
Table 4　Average of total size of COST messages (DPOP)

c \ n	25	50	100	200
n-1	72	147	297	597
1.125n	157	607	6020	–
1.25n	369	5147	–	–
1.5n	1458	235975	–	–

表 5　DPOP とバックトラックの併用 (木の幅の制限=4)
Table 5　DPOP with partial backtracking (limitation of width = 4

n	25	50	100
c	1.5n	1.25n	1.125n
num. of cycles			
DPOP	23	36	46
DPOP+BT1	102	344	322
DPOP+BT2	41	115	126
total size of COST messages			
DPOP	1458	5147	6020
DPOP+BT1	1854	7534	7134
DPOP+BT2	1612	5884	6757
max size of COST messages			
DPOP	2187	6561	6561
DPOP+BT1, DPOP+BT2	81	81	81

表 6　DPOP とバックトラックの併用 (木の幅の制限=8)
Table 6　DPOP with partial backtracking (limitation of width = 8

n	50	100	200
c	1.5	1.25	1.125
num. of cycles			
DPOP	44	–	–
DPOP+BT1	249	1188	4295
DPOP+BT2	84	217	891
total size of COST messages			
DPOP	235975	–	–
DPOP+BT1	271246	963243	4713141
DPOP+BT2	250409	823191	4480157
max size of COST messages			
DPOP	177147	–	–
DPOP+BT1, DPOP+BT2	6561	6561	6561

5. 評　　価

提案手法についてシミュレーションにより評価した．文献 [9] では比較的制約密度が小さいと考えられるイベントスケジュールなどの問題を例題として用いているが．本論文では問題の規模の議論を簡単にするために，ランダムに生成された 2 項制約で関連する問題を例題として用いた．値域は各ノード i とも $|D_i| = 3$ とした．制約辺に対応付けられた関数の評価値は，制約辺で関連する 2 つの変数の値の各組み合わせについて，1 から 10 の整数値をランダムに選択した．

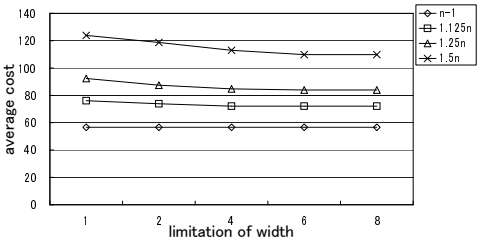
制約網は単一の連結成分を持つように，次の手順で構成した．(1) まず，全てのノードからなる集合からランダムに単一のノードを選択し，単一のノードからなるグラフを生成する．(2) グラフに含まれていないノードをランダムに選択しグラフに追加する．このとき，グラフが生成木になるように，ランダムに選択した制約辺を加える．全てのノードがグラフに追加されるまで，この操作を繰り返す．(3) 全てのノードを追加したとき，制約辺の数が不足する場合は，不足する数の制約辺をランダムに選択し，追加する．

変数の数 n に対して制約辺の数 c が $n-1$, $1.125n$, $1.25n$, $1.5n$ とし，各 n, c について 10 問の例題を生成した．深さ優先探索木を生成する際は最大次数のノードを優先して探索するものとした．

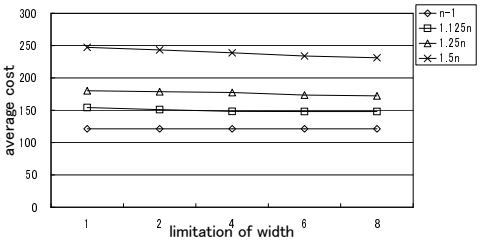
シミュレーションでは，システム全体はメッセージサイクルを単位とする反復処理を行い，各メッセージサイクルでは次の (1)(2) の処理を行うものとした．(1) 各ノードはそれぞれ受信メッセージキューと送信メッセージを持つ．各ノードは受信メッセージキューにあるメッセージを読み出し，処理の結果に基づいてメッセージを送信メッセージキューに書き込む．(2) 各ノードの送信メッセージキューにあるメッセージを，あて先ノードの受信メッセージキューに移動する．

5.1　生成された擬似的な木と DPOP の評価

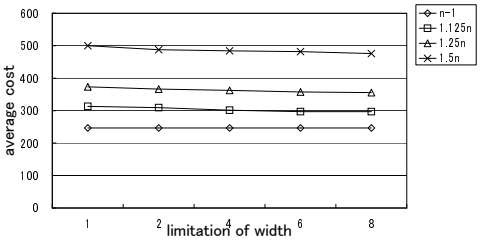
例題の制約網に対して生成された擬似的な木の深さ



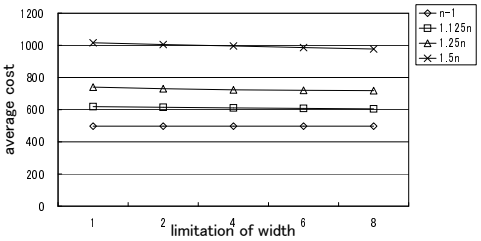
(a) n=25



(b) n=50

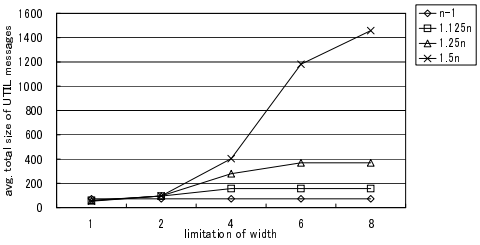


(c) n=100

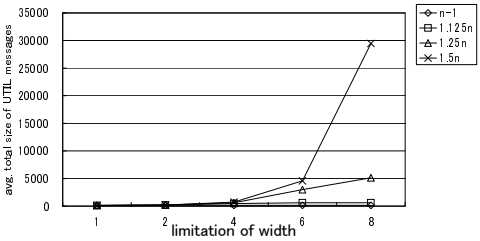


(d) n=200

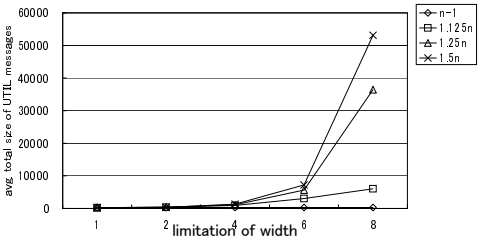
図 6　変数値を固定した場合 (コストの平均)
Fig. 6　Using fixed value (average cost)



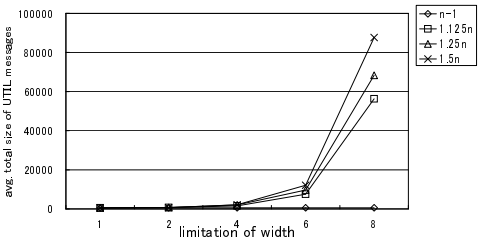
(a) n=25



(b) n=50



(c) n=100



(d) n=200

図 7　変数値を固定した場合 (**COST** メッセージのサイズの総和)
Fig. 7　Using fixed value (avg. total size of COST messages)

を表 1 に，擬似的な木の幅を表 2 に示す.

これらの問題について，DPOP に基づく手法を実行した際に要したサイクル数を表 3 に，**COST** メッセージで送信されたコストのサイズの総和を表 4 に示す．これらの結果では，**COST** メッセージに含まれるコストのサイズの最大値が 10^6 を超える問題が含まれる場合については空欄とした．表 3 に示されるように，DPOP の実行に要するサイクル数は表 1 に示される木の深さの 2 倍程度となる.

5.2　バックトラックによる遅延の評価

バックトラックを部分的に導入する場合，明らかにバックトラックの対象となる変数の個数に従って指数関数的にメッセージサイクル数が増大する．このため実際には 4.3 で述べたような手法の併用が必要である．ここでは，特にバックトラック処理による遅延について評価することを目的とし，高々数個の変数をバックトラックの対象とする場合について評価した.

擬似的な木の幅の制限値を統一するために，表 2 の (n,c) の組について，木の幅の平均が比較的近い {(25, 1.5n), (50, 1.25n), (100, 1.125n)}, {(50, 1.5n), (100, 1.25n), (200, 1.125n)} について，それぞれ木の幅の制限値を 4, 8 とした場合について評価した.

DPOP と 4.1 の手法 (DPOP+BT1) および 4.2 の手法 (DPOP+BT2) について比較した．

表 5 に擬似的な木の幅の制限値が 4 の場合，表 6 に擬似的な木の幅の制限値が 8 の場合の結果をそれぞれ示す．いずれの場合も，メッセージサイクル数については DPOP よりもバックトラックを行う DPOP+BT1,DPOP+BT2 の方が大きい．バックトラックの範囲を削減する DPOP+BT2 は DPOP+BT1 よりもメッセージサイクル数は少ない．また，**COST** メッセージのサイズの総和は，DPOP よりも DPOP+BT1,DPOP+BT2 の方が大きい．これは，DPOP+BT1,DPOP+BT2 では，解の決定の段階において，一部のコストの再計算を行うことによる．

コストの集計の段階において，バックトラックの範囲にあるノードは，一回のバックトラックごとに，バックトラックの対象である上位ノードの変数値の範囲についてのみコストを計算する．また，木の幅の制限に伴うメモリサイズの制限により，一回のバックトラックごとに計算した解の範囲のコストのみを記憶できる．このため，各ノードが最終的に保持する解のコストは，最後に計算された一部の解の範囲に限られる．

解の決定の段階において，各ノード i の変数値は，ノード i の上位ノードが決定した部分解 $agentview_i$ と各ノードの保持する解のコストに基づいて決定される．しかし，$agentview_i$ に対するコストを，ノード i が保持していない場合は，**COST** メッセージを用いて $agentview_i$ に対するコストを再度計算する必要がある．このような場合に **COST** メッセージのサイズの総和が増加する．このようなメモリの制限に起因する解の決定段階でのコストの再計算は，メモリ制限のある A*アルゴリズムに基づく解法である Adopt [5] で行われるものと同様である．

また，これらの実験では，5.1 と同様に，DPOP については，**COST** メッセージに含まれるコスト情報のサイズの最大値が 10^6 を超える問題が含まれる場合の結果は空欄とした，一方，DPOP+BT1 および DPOP+BT2 の結果については **COST** メッセージに含まれるコスト情報のサイズの最大値は木の幅の制限値に依存する．これらの実験での提案手法の **COST** メッセージの最大サイズは，表 5, 6，に示されるように，DPOP の場合よりも十分に小さい．

5.3 変数値を固定した場合の評価

バックトラックの対象とする変数値を固定した場合の評価として，各問題について擬似的な木の幅の制限値を 1, 2, 4, 6, 8 としたときのコストの平均を図 6 に，**COST** メッセージのサイズの総和の平均を図 7 に示す．制限値を大きくすることによりコストは改善するが，**COST** メッセージのサイズの総和については，擬似的な木の幅が大きい場合は制限値を大きくするに従って従い指数関数的に増大する．

擬似的な木の幅の増加にしたがって，DPOP では **COST** メッセージの最大サイズが指数関数的に増加する．また，一部にバックトラックを導入する場合はサイクル数が指数関数的に増加する．このため上記のような近似的な手法を導入することが実際的であると考えられる．特に，一部の変数値を固定しても他の変数値の変更によりコストを改善できる余裕がある問題や，コストの下限値による貪欲戦略が比較的有効である問題では，近似解解法を用いてもある程度の解の品質が得られると考えられる．図 6 に示す結果では，擬似的な木の幅の制限により，値が固定される変数の数が変化するが，極端な解品質の変化は示されなかった．

6. 関連研究と考察

提案手法は，動的計画法に基づく手法である DPOP [9] の一部の処理をバックトラックに置き換えるものである．この基本的なアイデアは [9] でも指摘されているが，本稿では，バックトラックに基づく解法である ADOPT [5] の機構の一部を導入し，ADOPT と DPOP の中間的な動作を行うための手法を示した．

また，さらに ADOPT 同様に各ノードでコストの上下界にもとづきバックトラックを行うように一般化することも可能であると考えられる．しかし，コストの上下界に基づく効率的なバックトラックの導入のためには，各 **COST** メッセージの処理により探索される解の範囲についての検討が必要であると考えられる．特に，バックトラックの対象とする変数についての値域を単純に網羅的に探索するのでは無く，多状態コミットメント [12] に類する手法の導入が考えられる．

記憶のサイズの制限のために **COST** メッセージで送受されるコストの表現を変更し冗長性を除くことが考えられる．複数の解とコストを内包関係を利用し統合する手法 [13] をメッセージの表現に応用することが考えられる．

DPOP のコストの集計において，一部のコスト情報が欠落することを許す近似的な解法が，示されている [10]．この手法では，解のコストの一部の集計を省

略し，コスト情報を表す hyper-cube の次元を削減する．コスト情報の次元を削減する際に，それまでに各ノードで求められたコストの上下界に基づいて，解の誤差を評価する．また，コスト情報の次元の削減の基準として，解のコストの近似比や，最大の木の幅を用いる．このような近似解法は，本質的にはヒューリスティクスに基づく手法であり，記憶のサイズと解の精度のトレードオフが重要である．

7. むすび

本論文では，擬似的な木と動的計画法に基づく分散制約最適化手法の補助的な手段として，擬似的な木の幅が制限される値を超える部分について記憶のサイズを制限するための基本的な手法を提案した．一部の変数について部分的にバックトラックを導入する手法，および，貪欲戦略の一つとして一部の変数の値を予め固定する手法を示した．また，提案手法の有効性について評価した．

一般的な組み合わせ最適化問題では動的計画法は指数関数的な空間複雑度を必要とする．従って，問題を構成する際に制約密度に制限を設けることや，擬似的な木の幅を小さくするようなノードの順序付けを生成することが重要である．提案手法はこのような制限の補助的な手段として利用することが適切であると考えられる．上記を含めた，実際的な問題への適用が今後の課題である．

文　献

[1] J. Liu and K. Sycara, "Exploiting problem structure for distributed constraint optimization", Proc. 1st Int. Conf. on Multiagent Systems, pp.246–253, 1995.

[2] K. Hirayama and M. Yokoo, "Distributed Partial Constraint Satisfaction Problem", Proc. 3rd Int. Conf. on Principles and Practice of Constraint Programming", pp.222–236, 1997.

[3] K. Hirayama and M. Yokoo, "An Approach to Over-constrained Distributed Constraint Satisfaction Problems: Distributed Hierarchical Constraint Satisfaction", Proc. 4th Int. Conf. on Multiagent Systems, pp.135–142, 2000.

[4] M. Lemaître and G. Verfaillie, "An Incomplete Method for Solving Distributed Valued Constraint Satisfaction Problems", In Proc. AAAI97 Workshop on Constraints and Agents Providence, 1997.

[5] P. J. Modi, W. Shen, M. Tambe and M. Yokoo, "Adopt: asynchronous distributed constraint optimization with quality guarantees", Artif. Intell., vol.161, no.1–2, pp.149–180, 2005.

[6] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce and P. Varakantham, "Taking DCOP to the Real World: Efficient Complete Solutions for Distributed Multi-Event Scheduling", Proc. 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp.310-317, 2004.

[7] S. M. Ali, S. Koenig and M. Tambe, "Preprocessing techniques for accelerating the DCOP algorithm ADOPT", Proc. 4rd Int. Joint Conf. on Autonomous Agents and Multiagent Systems, pp.1041–1048, 2005.

[8] A. Petcu and B. Faltings, "A distributed, complete method for multi-agent constraint optimization", 5th Int. Workshop on Distributed Constraint Reasoning, 2004.

[9] Adrian Petcu, Boi Faltings, "A Scalable Method for Multiagent Constraint Optimization", Proc. 9th Int. Joint Conf. on Artificial Intelligence, pp.266–271, 2005.

[10] A. Petcu and B. Faltings, "Approximations in Distributed Optimization", Proc. Principles and Practice of Constraint Programming 2005, pp.802–806, 2005.

[11] M. Yokoo, E. H. Durfee, T. Ishida and K. Kuwabara, "The Distributed Constraint Satisfaction Problem: Formalization and Algorithms",IEEE Trans. Knowledge and Data Engineering, vol.10, no.5, pp.673–685, 1998.

[12] 北村泰彦, 宮地智久, 横尾真, 辰巳昭治, "多状態コミットメント探索とその評価", 人工知能誌, vol. 14, no. 5, pp. 860–869, 1999.

[13] 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", 信学論, vol. J88-D-I, no. 8, pp. 1235–1246, 2005.

（平成年月日受付，月日再受付）

松井　俊浩　（正員）

平成 7 名工大電気情報工学科卒業．平成 11 名工大大学院博士前期課程修了．平成 18 名工大大学院博士後期課程修了．現在，同大学情報基盤センター助手．

松尾　啓志　（正員）

昭和 58 名工大・情報卒．昭和 60 同大学大学院修士課程了．同年松下電器産業（株）入社．平 1 名工大大学院博士課程了．同年名工大・電気情報・助手．講師，助教授を経て，平 15 同大大学院教授，現在に至る．分散システム，画像認識，分散協調処理に関する研究に従事．工博．情報処理学会，人工知能学会，IEEE 各会員．

Abstract Distributed constraint optimization problem (DCOP) is an area of research of multi-agent system and distributed cooperative problem solving. Recently, a dynamic programming based method for DCOP has been proposed. The dynamic programming method is performed using pseudo-tree of constraint network. However, space complexity and message size of the method is exponential in width of the pseudo-tree. In this paper, we propose efficient methods which reduce space complexity of the dynamic programming based method. The proposed methods apply backtracking to several variables according to width of the pseudo-tree. Additionally, range of the backtracking is cropped to reduce extra message passing. As another effective method, an approximate method is also shown. In the approximated method, values of some variables are fixed in preprocessing using a heuristics. Experimental results show efficiency of proposed methods.

Key words Distributed Constraint Optimization, Distributed CSP, Multi-agent, Pseudo-tree