非同期分散協調探索アルゴリズムのためのプログラミング言語に関する一考察

A study of prgorgamming laguages for asynchronous distributed cooperative search

松井 俊浩　松尾 啓志
Toshihiro Matsui　Hiroshi Matsuo

名古屋工業大学
Nagoya Institute of Technology

In this study, a consideration for distributed cooperative search algorithms is shown. Distributed constraint optimization problem (DCOP) is a fundamental framework of multi-agents cooperation. States and relationship of agents are represented using DCOPs. Distributed cooperative search algorithms are applied to solve DCOPs. In this research area, imperative programming languages are often used to implement the search algorithms. However, imperative programming of some DCOP solvers are rather complicated due to partially asynchronous distributed processing. Therefore some declarative programming paradigm will be necessary for practical programming. We analysis some DCOP search algorithm and consider requirements of programming language for DCOP solvers.

1. はじめに

近年，マルチエージェントシステムの協調問題解決を，制約最適化問題として表し，分散協調探索アルゴリズムにより解を得る分散制約最適化問題 (DCOP) が研究されている．DCOP ではエージェントの状態は変数として表現され，複数のエージェントの関係は，評価関数として表現される．各エージェントは自身の変数の値を決定する権限をもち，自変数および自変数と関係する評価関数のみを直接的に知る．このように，複数のエージェントにわたって配置された問題を解決するために，エージェントはメッセージ交換型の分散アルゴリズムとして構成された探索処理を実行する．このようなアプローチは，協調問題解決における本質的な処理を，離散組み合わせ最適化問題と探索アルゴリズムからなる枠組みを基礎として理解しようとする点に重要性があると考えられる．

DCOP の文献の多くでは，アルゴリズムの理論的な説明として，いくつかの命題により処理の本質的に重要な点があらわされているが，全体的な処理は手続き的な疑似コードにより説明されている．メッセージ交換をともなう分散処理を手続き的に理解することは直感に反する部分がある．また，実際のプログラミングに命令型 (手続き型) 言語を用いることが多いが，命令型のプログラミングは分散アルゴリズムの記述には適さない面がある．特に，ある程度の非同期性を許容する複雑な分散アルゴリズムを誤りなく記述することは容易ではない．

分散アルゴリズムの理解には宣言的表現が重要であり，特に，関数による計算の表現，不変な条件，およびそれらについての命題などは，DCOP の解法の理論面の正しさの説明や，プログラムの動作の検証でも必要である．これらの宣言的な表現を中心とする要素を適切に組み合わせて処理系を構成する作業を，プログラミング言語のレベルでサポートすることができれば，複雑な分散アルゴリズムの実装が容易になることが期待できる．

本論文では，DCOP の解法と，付随する前後処理や探索効率の改善手法の記述を目的として，非同期分散協調探索アルゴリズムに適したプログラミング言語について考察する．

連絡先: 名古屋工業大学，〒 466-8555 愛知県名古屋市昭和区御器所町，{matsui.t, matsuo}@nitech.ac.jp

2. DCOP

2.1 形式化

制約最適化問題 (COP) は，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．変数 $x_i \in X$ は離散かつ有限の値域 D_i の値をとる． 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 $f_{i,j}(d_i, d_j) : D_i \times D_j \rightarrow \mathbb{N}$ 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．COP は分散協調問題解決の基礎的な問題である，分散制約最適化問題 (DCOP) にも応用される．DCOP では，制約と変数が複数のエージェントに分散して配置され，各エージェント i は自身の状態を表す変数 x_i を持つ．制約で関係するエージェントのメッセージ交換を伴う分散アルゴリズムにより大域的最適解を探索する．

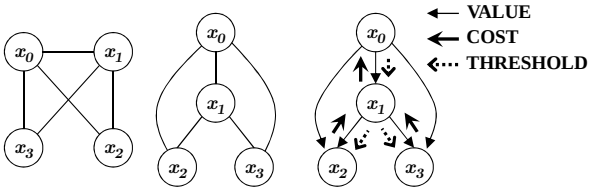
2.2 DCOP の解法

DCOP の解法は，制約最適化問題の解法をメッセージ交換に基づく分散アルゴリズムとして表現したものととらえることができる．これらの解法は，厳密解法と非厳密解法に大別される．厳密解法は分枝限定法，動的計画法，A*などにもとづく．Pseudo-tree にもとづく ADOPT[Modi 05], ADOPT-ing[Silaghi 06], DPOP[Petcu 05] や，仲介者エージェントを用いる OptAPO[Mailler 04] などが提案されている．とくに，分枝限定法や A*を応用した解法はその処理が比較的複雑である．一方で非厳密解法として，山登り方や確率的解法にもとづく．DSA[Zhang 05], Max-Sum[Farinelli 08] などが提案されている．非厳密解法の多くは，メッセージ交換を伴う分散処理の点では，比較的簡単である．本論文では，処理の記述が比較的難しい厳密解法について，プログラミングにおける課題について検討し，言語の要件について考察する．特に初期の検討事例として Pseudo-tree にもとづく厳密解法である DPOP,ADOPT を対象とする．

3. Pseudo-tree にもとづく解法

3.1 Pseudo-tree

ADOPT, DPOP, ADOPT-ing などは，制約網に対する pseudo-tree にもとづく変数順序を用いる．制約網と対応する pseudo-tree



(a) constraint network (b) pseudo-tree (c) computation in Adopt

図 1: Pseudo-tree/ADOPT

例を図 1(a),(b) に示す．pseudo-tree は解法の前処理の段階，または解探索の最中に生成される．pseudo-tree を生成する簡単な手法は，制約網に対して深さ優先探索に基づいてノードを列挙する方法である．元の制約網の辺は，pseudo-tree の木辺と後退辺に分類される．木辺は深さ優先探索木の辺に相当し，変数の半順序関係を表す．木辺以外の辺は後退辺である．pseudo-tree の異なるサブツリー間には辺は存在しない．この性質を用いて，探索処理をサブツリー間で並列的に実行することができる．

3.2 大域的最適評価値の計算

Pseudo-tree にもとづく DCOP 解法の最適評価値の計算の方法を示す．以降での解法の分析の議論を一般化するために，従来の ADOPT, DPOP の文献と異なる表現 [Silaghi 06] を用いる．

3.2.1 valued-nogood

valued-nogood は部分解について導出された評価関数の合計値の最小値を表す．評価関数 $f \in F$ をラベル f で識別できるものとし，そのラベルの集合を $\mathcal{F}$ によりあらわす．ある部分解 s にたいして，集合 $\mathcal{F}' \subset \mathcal{F}$ についての評価関数の合計の最小値が m であることを，valued-nogood $vn = (s, \mathcal{F}', m)$ によりあらわす．大域的に最適な評価値 m^* を表す valued-nogood は，空の部分解 ϕ，$\mathcal{F}$ により $vn^* = (\phi, \mathcal{F}, m^*)$ のようにあらわされる．このような最適評価値を導出するために，次節に示す sum-inference, min-resolution の 2 つの命題ににもとづく操作が用いられる．

3.2.2 sum-inference

sum-inference は 2 つの valued-nogood の評価値を結合する．valued-nogood $vn_a = (s_a, \mathcal{F}'_a, m_a)$ と，$vn_b = (s_b, \mathcal{F}'_b, m_b)$ について，部分解 s_a と s_b に含まれる変数値の割り当てが矛盾せずかつ，$\mathcal{F}'_a \cap \mathcal{F}'_b = \phi$ であるとき，sum-inference により $vn_c = (s_a \cup s_b, \mathcal{F}'_a \cup \mathcal{F}'_b, m_a + m_b)$ が導出される．

3.2.3 min-resolution

min-resolution は複数の部分解からある変数についての変数値の割り当てを除去する．変数 x_i の値域 D_i について，$d \in D_i$ なる割り当て (x_i, d) を部分解に含む valued-nogood $vn_d = (s_d, \mathcal{F}'_d, m_d)$ が D_i に含まれるすべての値について，導出されているものとする．このとき，すべての $d \in D_i$ について，s_d に含まれる変数値の割り当てが矛盾しなければ，min-resolution により，$vn_c = (\cup_{d \in D_i}(s_d \backslash \{(x_i, d)\}), \cup_{d \in D_i} \mathcal{F}'_d, \min_{d \in D_i} m_d)$ が導出される．

3.2.4 pseudo-tree に基づく最適値の計算

sum-inference と min-resolution を pseudo-tree にしたがってボトムアップに適用することにより，大域的に最適な評価値を計算できる．ここで，各変数は自身の変数と評価関数で直接関係する，上位の変数の値をすべて知っているという前提を置く．各葉ノード i は，自身の変数と評価関数で直接関係する上位の変数の変数値の組み合わせ s について，s に含まれる変数値の割り当て (x_j, d_j) と，自身の変数値の割り当て (x_i, d_i) に関する評価関数値を計算し，valued-nogood $vn_{\{(x_j,d_j),(x_i,d_i)\}} = (\{(x_j, d_j), (x_i, d_i)\}, \{f_{j,i}\}, f_{j,i}(d_j, d_i))$ を得る．これらの評価関数は明らかに重複しないため，sum-inference を適用して合計の評価値についての $vn_{s \cup \{(x_i,d_i)\}}$ を導出できる．また，すべての $d_i \in D_i$ についての $vn_{s \cup \{(x_i,d_i)\}}$ から，min-resolution により自身の変数値の割り当て (x_i, d_i) を除去した vn_s を導出できる．葉ノード以外のノード j は，葉ノードと同様に自身の変数と評価関数で直接関係する上位の変数の変数値の組み合わせ $s' \cup \{(x_j, d_j)\}$ についての valued-nogood および，全ての子ノードで $s' \cup \{(x_j, d_j)\}$ について導出された valued-nogood を sum-inference により結合し，$vn_{s' \cup \{(x_j,d_j)\}}$ を導出できる．さらに，min-resolution により，$vn_{s'}$ を導出できる．結局，根ノードでは，大域的に最適な評価値を表す valued-nogood が導出される．各ノード i を根とするサブツリーの，上位ノードの部分解 s ついての評価値の最小値 $m_{s,i}$ は次のように再帰的にあらわすことができる．ただし，C_i は子ノードの集合，U_i は変数 x_i と評価関数で直接関係する変数をもつ i の上位近傍ノードの集合を表す．

$$m_{s,i} = \min_{d \in D_i} m_{s,i,d} \quad (1)$$

$$m_{s,i,d} = \sum_{j \in U_i} f_{j,i}(d_j, d_i) + \sum_{k \in C_i} m_{(s \cup \{(x_i,d)\}),k}$$

3.2.5 評価値の上下界

ある部分解 s についての valued-nogood vn が未知のとき，vn についての評価値の最小値の下界値, 上界値はそれぞれ，0, ∞ とみなすことができる．valued-nogood はそれぞれ，$vn^{lb} = (s, \phi, 0), vn^{ub} = (s, \phi, \infty)$ のようにあらわされる．この上下界値により，未知の valued-nogood を含む導出が一般化される．

3.3 最適解の計算

大域的な最適コスト値が得られているとき，各変数 x_i の最適値 d_i^* は次のようにあらわされる．

$$d_i^* = \text{argmin}_{d \in D_i} m_{s,i,d} \quad (2)$$

根ノードでは $s = \phi$ であり，速やかに解が得られる．根ノード以外のノードでは，祖先ノードについての部分解 s が全て得られているときに解が得られる．

3.4 情報の分散配置とメッセージ交換

DCOP では各ノードのみが自身の変数値を決定する．また，初期状態において，各変数ノードが知る知識は局所的な情報に制限される．自身の変数，自身の変数に関する評価関数および，評価関数で直接的に関係する近傍ノードの知識が一般的である．ただし DPOP では近傍ノードの変数の値域も事前に知ることを前提としている．探索処理においては，変数値と valued-nogood がメッセージ交換により送受される．

3.5 DPOP

DPOP は，ボトムアップな 1 パスの処理で最適コストを計算し，トップダウンな 1 パスの処理で最適解を決定する．これは，簡単な動的計画法であるため，式 1, 2 を基礎として，再帰的な計算を，逐次的に実行し，各変数ノードの間の値の授受をメッセージ交換に置き換えることで分散アルゴリズム化できる．すなわち，まず，式 1 を葉ノードからボトムアップに計算してゆき，根ノードまで計算した後，式 2 を根ノードから葉ノードまでボトムアップに計算してゆけばよい．ただし，動的計画法にもとづくため，変数ノード i における $m_{s,i}$ は，$s \in S^{PP_i}$ なる部分解それぞれについて同時に計算する．ここで，S^{PP_i} は変数ノード i を根とするサブツリーに含まれるいずれかの変数と評価関数で関係する，祖先ノード全ての変数

の集合 X^{PP_i} についての部分解の集合を表す．つまり，すべての $s \in S^{PP_i}$ についての $vn_s = (s, \mathcal{F}', m_{s,i})$ からなる集合 VN^{PP_i} が式 1 の結果により得られ，i の親ノードに送信される[*1]．一方で，式 2 の計算では，各変数ノード i は親ノードから最適な部分解 s^* を受信したとき，自身の最適な変数値の割り当て $\{(x_i, d_i^*)\}$ を加えた部分解 $s* \cup \{(x_i, d_i^*)\}$ を子ノードに送信する．

DPOP は，再帰的な評価関数の表現から逐次的な分散アルゴリズムを導出し手続き的に実装することが，直感的にも容易な手法であるといえる．

3.6 ADOPT

ADOPT は分枝限定法/A*アルゴリズムに基づき，反復的な計算により最適コストを求め，最適解を決定する．DPOP と比較して，ADOPT の探索処理は複雑である．この手法では，valued-nogood の導出やコストの境界値の比較を伴う反復的な探索処理をメッセージ交換型の分散処理として構成する必要がある．ADOPT で用いられるメッセージを図 1(c) に示す．

3.6.1 分枝限定法/A*

ADOPT は分枝限定法/A*としてとらえられる．すなわち，あるスナップショットにおいて，pseudo-tree の根ノードからあるノードへのパス上では，トップダウンに分枝限定法が実行されている．さらに，各ノード i では，i の祖先ノードの現在の部分解 s と自身の各変数値 $d \in D_i$ からなる，部分解 $s \cup \{(x_i, d)\}$ それぞれについて，i を根とするサブツリーおいてボトムアップに計算された valued-nogood を保持している．これはメモリ制限のある動的計画法であるといえる．これらから構成される処理はメモリ制限された A*アルゴリズムに類似する．

3.6.2 決定過程としての側面と論理時刻

分枝限定法に基づく探索のため，現在探索中の部分解は，分散処理の遅延を考慮しなければ，システム全体で 1 つである．最良優先探索戦略により，各変数値は順次変更されるため，探索アルゴリズムはある決定過程としてふるまうと考えられる．この決定過程における各状態は，現在の部分解を構成する各変数値と，その変数値に対応する明示的あるいは暗黙的な論理時刻のベクトルにより区別することができる．また，現在の部分解に対して導出された valued-nogood は，その部分解に固有である．他の新しい時刻の変数値を含む部分解では無効となり破棄されなければならない．

3.6.3 変数値の伝達

各ノード i は valued-nogood を導出するために，i の祖先ノードの現在の部分解を必要とする．そのため，各変数値をメッセージにより各ノードに伝達する必要がある．すなわち，i を根とするサブツリーのいずれかのノードと評価関数で関係する祖先ノードの現在の変数値は，i に伝達されなければならない．

メッセージ数や，情報の伝達の遅延の程度を考慮して，変数値の伝達の経路を決定し，他のメッセージと統合することもできる．ADOPT では，あるノードと評価関数で直接関係する下位近傍ノードには，専用のメッセージ VALUE により変数値を伝達する．それ以外のノードには，valued-nogood をボトムアップに伝達するメッセージ COST に変数値を重畳させる．

各ノードでは，祖先ノードの部分解は一つしか保持されない．そのため論理時刻において古い変数値は新しい変数値により置き換えられる．このための簡単な方法は，各変数値に論理時刻を直接付加することである．また，変数値の配信の順序とタイブレークの規則を注意深く設計することで，変数値に関する論理時刻を暗黙的に扱うことができる[*2]．

3.6.4 valued-nogood の導出と伝達

ADOPT における valued-nogood の計算方法は基本的には式 1 にもとづく．ただし，各ノードの変数値が変更された直後は，その子孫ノードでは祖先ノードについての部分解が変更されるために，サブツリーにおける valued-nogood は未知となる場合がある．このような場合を含むため，3.2.5 節に示した未知の valued-nogood についての評価値の上下界値を用いる．真の valued-nogood が得られている場合は上下界値は同一である．

各ノード i が祖先ノードの部分解 s について導出した，valued-nogood $vn = (s, \mathcal{F}', m)$ は，式 1 の valued-nogood/評価値の導出のために i の親ノードで評価される．これは DPOP と同様であるが，祖先ノードの 1 つの部分解 s のみに対する valued-nogood を導出することと，valued-nogood が評価値の上下界値について導出される点が異なる．

さらに，各ノード i の子ノード j が導出した valued-nogood のうち，i の現在の変数値に対応する valued-nogood は子ノード j も知る．子ノードが，親ノードで実際に採用された valued-nogood を知ることは，子ノードの変数値を親ノードの決定と矛盾しないように変更することが必要であるか否かを効果的に決定するための，判断の基準を与える[*3]．

子ノードから親ノードおよび，親ノードから子ノードへの valued-nogood はそれぞれメッセージ COST,THRESHOLD により伝達される．

3.6.5 最良優先探索戦略による変数値の選択

各ノードは基本的には最良優先探索戦略によって変数値を決定する．すなわち，自変数の現在の値 d についての valued-nogood の評価値の下界値 m が，別の変数値 d' についての m' より大きいとき，変数値を d' を変更する．これにより，最小の下界値に対応する変数値が選択される．ただし，3.6.4 で述べたように，親ノードの現在の変数値に対応する valued-nogood vn^{thr} を知る場合は，自変数の現在の変数値に対する valued-nogood の評価値が vn^{thr} の評価値を上回るときのみ，自変数値を変更する．これにより非同期性に起因する冗長な探索が削減される．また，アルゴリズムの開始時には適当な変数値を選択する．

3.6.6 最適解の決定

根ノードではいずれ valued-nogood の上下界値が収束し，式 1 の値が得られる．その後の最適解の決定は基本的に式 2 に従う．また，その処理は DPOP の場合と同様にトップダウンに行われるが，必要に応じて式 1 の一部を再計算しなければならない．また，ここでは各ノードの自変数の最適な割り当ては valued-nogood の評価値の上界値に対応する値となる．

3.7 分散アルゴリズムの構成方法

ADOPT のアルゴリズムは，上記の概念にもとづく複数の不変な条件 (invaliants) を維持するように構成されている．たとえば，3.6.5, 3.6.6 節の，ノード i における変数値 d の選択のための条件は，祖先ノードの部分解 s および自変数値 d につ

∗1 DPOP や ADOPT では，式 1 に従う限り，sum-inference において，互いに矛盾しない s_a, s_b について $\mathcal{F}'_a, \mathcal{F}'_b$ の要素が重複することがない．そのため実際には $\mathcal{F}'$ は省略される．

∗2 ADOPT では暗黙的な論理時刻を用いているが，やや不明瞭であり，改善の余地があると考えられる．

∗3 実際の ADOPT ではノード i が各子ノードから受信した，部分解 $s \cup (x_i, d_i)$ にたいする valued-nogood について，各 valued-nogood の評価値をその上下界に含まれる値に投機的に決定する．そして子ノードには投機的に決定された評価値をもつ valued-nogood を伝達する．このような機構は valued-nogood が上下界値について与えられるために必要である．ここでは，このような投機的な配分を省略し本質的な概念を示した．

いての valued-nogood の上下界値，$m^{ub}_{s,i,d}$，$m^{lb}_{s,i,d}$, $m^{ub}_{s,i}$，$m^{lb}_{s,i}$ および，親ノードで選択された解に対応する valued-nogood の評価値 $m^{thr}_{s,i}$ を用いて

$$\begin{cases} m^{ub}_{s,i,d} = m^{ub}_{s,i} & \text{if 最適解の決定中} \wedge m^{ub}_{s,i} = m^{thr}_{s,i} \\ m^{lb}_{s,i,d} \leq m^{thr}_{s,i} & otherwise \end{cases} \quad (3)$$

のようにあらわされる．このような invaliants を中心として，初期化処理・メッセージ受信処理およびそれらから呼び出されるメッセージ送信処理をを矛盾なく構成することにより，正しく分散アルゴリズムを構成できる．しかし実際に命令型言語を用いてのプログラミングをする際には，非同期性のある処理の順序を慎重に考慮しなければならない．

4. プログラミング言語・環境の要件

以上の DPOP と ADOPT の事例から，プログラミング言語・環境の要件について考察する．

4.1 関数による計算の表現からの分散処理の導出

DPOP では 3.5 節に示したように，再帰的に定義される評価関数・最適解の表現から，分散処理系を導出することが容易である．すなわち，評価関数・最適解の関数と，各エージェントへの変数の配置の知識を適切に表現すれば，1) 再帰的な関数の各項の依存関係から逐次的な計算順序を得る．2) 各エージェントへの変数の配置の知識により各エージェントで行う計算を得る．これらにより必要な計算処理を構成する．3) 各エージェント間で伝達されるべき valued-nogood と変数値を得る．これらをメッセージとして表現し，メッセージ送受処理を挿入する．という手順で分散処理系を構成できる．このような規則に関する制約の記述をプログラムコードに付加し，それらを解決するような手法を用いることで，関数表現から分散アルゴリズムを導出するような手法が考えられる．関数型プログラミング言語にこのような機構を付加したものが有効と考えられる．

4.2 制約による処理要素の関係の表現

ADOPT は DPOP と比較して処理が複雑である．そのため関数による計算の表現とエージェントへの変数等の配置の表現に加えて，invaliants を含む断片的なアルゴリズムの規則などの要素とそれらが満たすべき関係を表す表現から，各要素を無矛盾に統合するようなプログラムの記述方法が有効であると考えられる．すなわちアルゴリズムを基礎となる複数の処理要素に分解しそれらの宣言的な表現をもとに，全体のアルゴリズムを導出することを考える．まず，全体の計算は，DPOP と同様にボトムアップなコストの計算と，それにつづくトップダウンな解の決定の計算に分解される．これらの計算では分枝限定法/A*にもとづく計算がおこなわれるが，それらは 1) ボトムアップかつ部分的な動的計画法による valued-nogood の導出，2) トップダウンな分枝限定法における最良優先探索戦略に基づく変数値の決定，3) 変数値，valued-nogood の関連ノードへの伝達，4) 最新の論理時刻の変数値に対する解と valued-nogood の保持，などに分解される． さらに各処理は invaliants とそれを維持するための断片的な手続きで表現される．これらの処理は単に逐次的に実行されるものではなく，並行的に実行される．そこで，各処理の構成要素とともに，それらの依存関係の制約を記述し，制約を解決することで，正しい処理の順序を導出し，さらにメッセージ交換の処理を導出して分散アルゴリズムを構成することが考えられる．

4.1 節の場合も含め，分散アルゴリズムにおいて特に複雑である処理の依存関係を制約として表現し，プログラミング言語の処理系により解決することが特に重要であるといえる．このような処理は制約論理型プログラミング言語により表現できる部分があると考えられる．しかし，基礎的な処理要素間の制約を解決して分散アルゴリズムを構成することを主眼においた言語は比較的少ないと考えられ，既存のプログラミング言語の適用可能性の調査およびそれらの拡張が必要であると考えられる．

5. 前後処理・効率化手法の統合

実際的な問題への適用や，探索アルゴリズム効率化のためには，基礎となるアルゴリズムに追加的な手法を統合することも必要である．たとえば，前後処理として，分散スナップショットによる制約網の構築，pseudo-tree の計算，大域的停止検出の処理を要する．また，一般に各エージェント内部で局所的に探索アルゴリズムを実行することで，エージェント間の通信を削減する効率化手法が用いられる．これらについても上述の議論と同様に，DCOP 解法のアルゴリズムとの関係を制約として表現することで，アルゴリズムの統合を行うことが考えられる．

6. まとめ

本論文では，DCSP/DCOP の基礎的な解法および，探索効率の改善手法や，前後処理を加えた解法の記述を目的として，非同期分散協調探索アルゴリズムに適したプログラミング言語の要件について検討・考察した．特に，初期の検討として pseudo-tree に基づく厳密解法である ADOPT と DPOP を事例に，基礎的な処理要素について分析した．また，分散アルゴリズムの構築において特に複雑である処理の順序性を解決するために，宣言的な表現を中心とした複数処理要素を，それぞれの依存関係に関する制約を満たすように統合する記述方法の必要性を指摘した．このような概念にもとづくプログラミング言語の調査・実装，および他の DCOP 解法についての検討が今後の課題である．

参考文献

[Farinelli 08] Farinelli, A., et al.: Decentralised coordination of low-power embedded devices using the max-sum algorithm, in *AAMAS*, pp. 639–646 (2008)

[Mailler 04] Mailler, R., et al.: Solving distributed constraint optimization problems using cooperative mediation, in *AAMAS*, pp. 438–445 (2004)

[Modi 05] Modi, P. J., et al.: ADOPT: Asynchronous distributed constraint optimization with quality guarantees, *Artif. Intell.*, Vol. 161, No. 1-2, pp. 149–180 (2005)

[Petcu 05] Petcu, A., et al.: A Scalable Method for Multiagent Constraint Optimization, in *IJCAI*, pp. 266–271 (2005)

[Silaghi 06] Silaghi, M. C., et al.: Nogood-based Asynchronous Distributed Optimization (ADOPT-ng), in *AAMAS* (2006)

[Zhang 05] Zhang, W., et al.: Distributed stochastic search and distributed breakout: properties, comparison and applications to constraint optimization problems in sensor networks, *Artif. Intell.*, Vol. 161, No. 1-2, pp. 55–87 (2005)