1C5-4

# メッセージ交換型の協調問題解決処理系の生成手法についての検討

## A Study of construction method of message-passing based distirbuted search algorithms

松井 俊浩[*1] 松尾 啓志[*1]
Toshihiro Matsui Hiroshi Matsuo

[*1]名古屋工業大学
Nagoya Institute of Technology

In distributed cooperative problem solving, distributed search algorithms that employ message-passing are necessary. The most of the search algorithms is intuitively constructed. However, to build such algorithm is complicated. Therefore, formal approaches to translate from non-distributed search algorithms to distributed processing are important. In this work, we study how to apply such translation to basic search methods that are represented by a form of function language.

## 1. はじめに

分散協調問題解決の処理系では，メッセージ交換を伴う分散処理として探索を含むアルゴリズムを記述することが必要となる．その記述と動作を理解することは比較的難しい．本研究では，このような処理系を生成するための手法について基礎的な検討を行う．このような目的のためには，協調問題解決の基礎的な構成要素とその依存関係およびメッセージ交換処理の指針についての表現をもとに，全体の処理系を構成する必要があると考えられる．本稿では初期の検討として，分散協調問題解決の一分野の分散制約最適化問題の解法として用いられる，探索アルゴリズムを，形式的に分散処理化する手段の構築を念頭に，関数型言語により表記された逐次型の探索アルゴリズムをもとに，メッセージ交換型の処理系を構成する基本的な手順について検討する．

## 2. 分散制約最適化問題

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 $X$, 二項制約の集合 $C$，二項関数の集合 $F$，により定義される．エージェント $i$ は自身の変数 $x_i$ を持つ．$x_i$ は，離散有限集合である値域 $D_i$ に含まれる変数値をとりうる．$x_i$ の値はエージェント $i$ のみが決定できる．制約 $c_{i,j} \in C$ は $x_i$ と $x_j$ の関係を表す．制約で関係する 2 変数についての，ある割り当て $\{(x_i, d_i), (x_j, d_j)\}$ 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．問題の大域的最適解 $\mathcal{A}$ はコスト関数値の合計 $\sum_{f_{i,j} \in F,\ \{(x_i,d_i),(x_j,d_j)\} \subseteq \mathcal{A}} f_{i,j}(d_i, d_j)$ を最小化する．このような大域的最適解を，複数のエージェントの協調的な問題解決により得ることが目的である．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．また，ここでは，簡単のために値域 $D_i$ は全ての変数で共通であるとする．

DCOP の厳密解法では変数の順序付けのために疑似木が用いられる．疑似木 (pseudo-tree) [Schiex 99] は制約網に含まれる変数に半順序関係を与えるような，グラフ上の構造である．あるグラフに対する疑似木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) 疑似木は元のグラフと同一の辺と頂点からなる．(2) 疑似木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．疑似木に対応する生成木の各頂点を，疑似木においても同様に木の頂点とみなす．疑似木ではサブツリーの間に後退辺は存在しない．この性質を用いることで，探索処理を並列に実行できる．疑似木において，エージェント $i$ と関係するエージェントの表現として，親 $p_i$，子の集合 $C_i$，$i$ と制約/評価関数で直接関係する祖先の集合 $N_i^u$，$i$ を根とする部分木に含まれるいずれかの変数と制約/評価関数で直接関係する祖先の集合 $PP_i$ を用いる．

連絡先: 松井俊浩，松尾啓志，名古屋工業大学，愛知県名古屋市昭和区御器所町，{matsui.t, matsuo}@nitech.ac.jp

## 3. 逐次的な解法の表記

分散アルゴリズムとしての解法は，その解法の本質的な処理の要素とその依存関係をもとに，データフローなどを理解し，メッセージ交換型の処理を導出することにより構成される．このために，比較的，解析が容易な形式により解法の本質を表現する必要がある．そこで，関数型言語を用いた大域的・逐次的な表現を用いることを試みる．ここでは，反復処理のない動的計画法，最良優先探索にもとづく解法を例とする．それぞれ，[Petcu 05] および [Modi 05] をごく簡略した解法の表記を意図する．

**動的計画法**: エージェント $i$ を根とする疑似木を $a_i = (i, N_i^u, PP_i, C_i, \{a_k | k \in C_i\})$ で表す．解法はコストの計算，最適解の決定の 2 段階からなる．エージェント $i$ を根とする疑似木について計算されたコストを $u_i = (i, U_i, \{u_k | k \in C_i\})$ で表す．ただし，$U_i = \{(s, v_s) | (x_j, d) \in s, x_j \in PP_i, d \in D_j\}$ であり，$v_s = \min_{d \in D_i}(\sum_{x_j \in PP_i, (x_j,d_j) \in s} f_{i,j}(d, d_j) + \sum_{k \in C_i, (s', v'_s) \in U_k, s \cup \{(x_i,d)\} \approx s'} v'_s)$ である．ここで $s \approx s'$ は 2 つの部分解 $s, s'$ に含まれる同一の変数の値が等しいことを表す．最適解に含まれる $x_i$ の値を $d_i^*$ で表す．疑似木の根を $r$ とするとき，$a_r$ を入力として，解法の全体の計算は，コストを計算する関数 $util$ と最適解を決定する関数 $value$ を用いて，次のように表される: $u_r = util(a_r) = (\cdots, \{u_k | util(a_k), k \in C_r\}, \cdots), s_r = \{\}, value(a_r, u_r, s_r) = (\cdots, (x_r, d_r^*) = opt(a_r, u_r, s_r), \{| value(a_k, u_k, s_r \cup \{(x_r, d_r^*)\}), k \in C_r\})$. ここで，’,’ は列挙子とし，最後の要素の値が関数の評価値と

なるものとする．また，$value$ は空の値を返す関数とする．関数 $opt(a_i, u_i, s_i)$ は $\mathrm{argmin}_{d \in D_i} v_s,\ s.t.\ (s, v_s) \in U_i, s \approx s_i \cup \{(x_i, d)\}$ なる最適解を返す．

**最良優先探索にもとづく解法**: エージェント $i$ を根とする疑似木を $a_i = (i, N_i^u, C_i, \{a_k | k \in C_i\})$ で表す．解法はコストの計算，最適解の決定の 2 段階からなる．$x_i$ の祖先の変数についての部分解 $s$ に対する，$x_i$ を根とする疑似木について計算されたコストの上下界値を $lb^*_{i,s}, ub^*_{i,s}$ で表す．ここで，$lb^*_{i,s} = \min_{d \in D_i} lb_{i,s,d}$ である．また，$lb_{i,s,d} = \sum_{x_j \in PP_i, (x_j, d_j) \in s} f_{i,j}(d, d_j) + \sum_{k \in C_i, (s', v'_s) \in U_k, s \cup \{(x_i, d)\} \approx s'} lb^*_{k,s'})$ である．$ub^*_{i,s}$ についても同様である．実際の計算では $lb_{i,s,d}, ub_{i,s,d}$ は，現在の上位の部分解 $s$ についてのみ保持する．$lb_{i,s,d}, ub_{i,s,d}$ が未知の場合は，それぞれ下限値 0，上限値 $\infty$ を用いる．最適解に含まれる $x_i$ の値を $d_i^*$ で表す．疑似木の根を $r$ とするとき，$a_r$ を入力として，解法の全体の計算は，コストを計算する関数 $cost$ と最適解を決定する関数 $term$ を用いて次のように表される: $s_r = \{\}, \{(x_r, d_r^*), lb^*_{r,s_r}, ub^*_{r,s_r}\} = cost(a_r, s_r) = (\cdots, \{LB_r, UB_r\} = initcosts(a_r), \cdots, loop(a_r, LB_r, UB_r, s_r)), s_r^* = \{(x_r, d_r^*)\}, term(a_i, s_r^*) = (\cdots, \{|(\{(x_k, d_k^*), lb^*_{k,s_r^*}, ub^*_{k,s_r^*}\} = cost(a_k, s_r^*), term(a_k, s_r^* \cup \{(x_k, d_k^*)\}), k \in C_r\}))$．ここで，$initcosts(a_i)$ は，$\{\{lb^*_{k,s_i \cup \{(x_i,d)\}} | d \in D_i, k \in C_i\}, \{ub^*_{k,s_i \cup \{(x_i,d)\}} | d \in D_i, k \in C_i\}, \}$ についての初期コストである上下限値からなる集合を返す．また，関数 $loop$ は次のように表される: $loop(a_i, LB_i, UB_i, s_i)) = (\cdots, (d_i = d\ s.t.\ d \in D_i, lb^*_{i,s_i} = lb_{i,s_i,d}), if(lb^*_{i,s_i} == ub^*_{i,s_i}, \{(x_i, d_i), lb^*_{i,s_i}, ub^*_{i,s_i}\}, (R = \{\{(x_k, d_k), lb^*_{k,s_i \cup \{(x_i,d_i)\}}, ub^*_{k,s_i \cup \{(x_i,d_i)\}}\} | cost(a_k, s_i \cup \{(x_i, d_i)\}), k \in C_k\}, LB'_i, UB'_i = update(LB_i, UB_i, R), loop(a_i, LB'_i, UB'_i, s_i))))$．なお，$if(c, a, b)$ は条件 $c$ が真であれば $a$，偽であれば $b$ を返す関数である．また，$update(LB_i, UB_i, R)$ は $LB_i, UB_i$ に含まれるコスト値を $R$ に含まれる対応するコスト値で置き換えたコスト値の集合を返す関数である．

## 4. 分散処理の構成手順

解法の分散処理化は基本的には以下の手順に従う．(1) 各エージェントに配置する関数を指定する．(2) 変数のうち，各関数の計算に必要であり他のエージェントの関数に不要なもの．および，各関数により値が決定されるものを抽出する．これらの変数は各エージェントの局所的な情報とする．(3) 各関数の計算に必要な変数のうち，他のエージェントで値が決定されるものを抽出する．これらの変数はメッセージにより交換される情報とする．(4) メッセージとして交換される変数を授受する，関数呼び出しを除去し，メッセージ送受信処理を挿入する．また，変数の依存関係に基づいて，各関数の計算および，その結果の送信の順序を制御する規則を挿入する．

単一代入型の関数型言語による表現から，これらの依存関係を形式的に導出することは比較的容易であるといえる．ここでは，上記の例について分散処理化の概要を示す．

**動的計画法**: (1) 各エージェント $i$ の処理として配置する関数は，$util(a_i), value_i(a_i, u_i, s_i)$ およびその内部の関数である．ただし，根のエージェント $r$ では，$u_r = util(a_r), s_r = \{\}, value = (a_r, u_r, s_r)$ の処理も行う．(2) これらの関数に必要であり，他のエージェントの関数に不要な変数は，$a'_i = (i, N_i^u, PP_i, C_i)$ である．これらの関数により計算される変数値は，$u'_i = (i, U_i)$ および $(x_i, d_i)$ である．但し，関数の内部で用いられる各変数も含む．ここでは，再帰的な関数の定義のためだけに必要な変数は除外し，各エージェントに保存される変数とする．(3) 各関数の計算に必要で，他のエージェントにより決定される変数は $util$ の内部で得られる $u_k, k \in C_i$ と $value$ の引数の $s_i$ である．(4) 各変数の依存関係により，葉のエージェントから順番に $util$ が最初に計算される．各 $util$ の戻り値は，親のエージェントに通知される．根のエージェントまで $util$ の値が伝搬した後，根から葉まで順番に $s_i$ の値が伝搬される．

**最良優先探索にもとづく解法**: (1) 各エージェント $i$ の処理として配置する関数は，$cost(a_i, s_i), term(a_i, s_i^*)$ である．ただし，根のエージェント $r$ では，$s_r = \{\}, \{(x_r, d_r^*), lb^*_{r,s_r}, ub^*_{r,s_r}\} = cost(a_r, s_r), s_r^* = \{(x_r, d_r^*)\}, term(a_r, s_r^*)$. の処理も行う．(2) これらの関数に必要であり，他のエージェントの関数に不要な変数は，$a'_i = (i, N_i^u, C_i)$ である．これらの関数により計算される変数値は，$(x_i, d_i^*), lb^*_{i,s_i}, ub^*_{i,s_i}$ である．(3) 各関数の計算に必要で，他のエージェントにより決定される変数は子 $k \in C_i$ から得られる，$(x_k, d_k^*), lb^*_{k,s_k}, ub^*_{k,s_k}$ および，$cost, term$ の引数の $s_i, s_i^*$ である．ただし，ここでは $(x_i, d_i^*)$ は本来 $i$ で処理可能だが記述の簡略化のために親ノードで処理している．(4) 各変数の依存関係により，根のエージェントから順番に $cost$ が計算される，各ノードでは現在の最良コストの境界が閉じた後に，親ノードに $(x_i, d_i^*), lb^*_{i,s_i}, ub^*_{i,s_i}$ を送信する．親ノードは関数 loop により他の最良解があれば，再び子ノードにその最良解を送信する．根ノードで最適解のコストの境界が閉じ後で，根ノードから葉ノードまで順番に変数値が再帰的に決定される．

## 5. まとめ

本稿では，分散制約最適化問題の解法として用いられる，探索アルゴリズムを，形式的に分散処理化する手段の構築を念頭に，関数型言語により表記された逐次型の探索アルゴリズムをもとに，メッセージ交換型の処理系を構成する基本的な手順について検討した．とくに，初期の検討として，単一の関数の定義により表される簡単な解法を対象とした，より高度 [Modi 05] な解法では，複数のパスでのメッセージ配信が行われる．これらは，同一の種類の解法を複数の異なる部分問題に適用することや，性質が異なる複数の解法を合成したものととらえることができる．このような複合的な解法では，メッセージ交換をともなう処理のタイミングの正確さを直感的に把握することが，難しいため，大域的にアルゴリズムを統合した表現を構成し，各処理の依存関係を形式的に扱い分散アルゴリズム化する手法は重要であると考えられる．一方で，単に再帰的な関数の定義のみで全ての処理を記述すると，非同期分散処理のタイミングに関する複雑な依存関係を網羅することは難しいと考えられる．このような依存関係の表現，および形式的な処理の変換についての検討が今後の主要な課題である．

## 参考文献

[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 *9th International Joint Conference on Artificial Intelligence*, pp. 266–271 (2005)

[Schiex 99] Schiex, T.: A note on CSP graph parameters, *Technical report 1999/03, INRA* (1999)