3E8-4

擬似的な木を用いたボトムアップな非同期分散最適化手法の効率改善

Efficient methods for distributed bottom up search using pseudo-tree

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

Distributed constraint optimization problem is an area of research in distributed cooperative problem solving. In recent years, distributed constraint optimization algorithms, which performs best-first search in bottom up manner according to pseudo tree, have been proposed. In this paper, we propose several efficient methods for the distributed bottom up best-first search. Derivation of partial solution is introduced to decrease number of backtracking among agents, Synchronization control method is applied to decrease number of communication message cycles. Additionally, error bounds are applied to obtain quasi optimal solution within less number of message cycles. Results of experiments are shown for the efficiency evaluation of the proposed heuristics methods.

1. はじめに

分散制約最適化問題 (DCOP) は [1] [2] [3] 分散協調問題解決・マルチエージェントシステムの基礎的な問題として研究されている．DCOP の解法として，pseudo tree を用いた分散探索手法 [3], [4] が提案されている．このような分散探索手法の一つとして，ボトムアップに最良優先探索を行なう手法が提案された [5]．本論文では，このボトムアップな最良優先探索の効率改善手法を提案する．

2. 問題の形式化

2.1 分散制約最適化問題

分散制約最適化手法は次のように形式化される．変数の集合を $X = \{x_1, \cdots, x_n\}$ で表す．変数の値域の集合を $D = \{D_1, \cdots, D_m\}$ で表す．2 項関数の集合を F で表す．$f_{i,j} \in F$ は $D_i \times D_j \rightarrow R$ なる関数である．$f_{i,j}$ は制約辺に対応する変数値の組合せ $\{(x_i, d_i), (x_j, d_j)\}$ の評価値を表す．問題の最適解は評価値の合計を大域的に最大化する変数値の組合せである．各エージェント (ノード) i は自身の変数 x_i を持つ．メッセージ通信を用いた分散アルゴリズムにより解の探索を行なう．

2.2 Pseudo tree

Pseudo tree[4] は制約網に対する変数の順序付けを与える．Pseudo tree は制約網の全てのノードと全ての制約辺を含む．制約辺は木の辺と後退辺に分類される．ノード i の親ノードを p_i で表す．ノード i の子ノードの集合を C_i で表す．ノード i とからの後退辺を持つ i の祖先ノードの部分集合を PP_i で表す．ノード i を根とする部分木の少なくとも一つのノードとの辺を持つ i の祖先ノードの部分集合を $\overline{PP}_i$ で表す．$\overline{PP_i}$ は p_i を含む．

2.3 pseudo-tree に基づく最良優先探索

pseudo-tree に基づくボトムアップな最良優先探索のための評価値の境界は次のように表される [5]．ノード i を根とする部分木に関する部分解の集合を S_i で表す．S_i は $\{i\} \cup \overline{PP}_i$ の各ノードの変数値の組合せを含む．部分解 s の評価値 $u(s)$ で表す．$u(s)$ は $[0, \infty)$ の値を取る．

連絡先: 松井俊浩, 名古屋工業大学 情報基盤センター，愛知県名古屋市昭和区御器所町，TEL 052-735-7242，FAX 052-735-5530，Email matsui.t@nitech.ac.jp

各ノードは部分解 s と評価値 $u(s)$ を親ノード p_i へ送信する．ノード i が s と $u(s)$ を p_i へ送信するとき，ノード i は (x_i, d_i) を s から除く．(x_i, d_i) を除いた部分解の集合を S_i^- で表す．i から p_i へ既に送信された部分解の集合を S_i^{-sent} で表す．S_i^{-sent} は S_i^- の部分集合である．ノード i は解と評価値を最良優先の順序に基づき送信する．ノード $j \in C_i$ から送信され，ノード i により受信された部分解の集合を $S_{i,j}$ で表す．$S_{i,j}$ は S_i の部分解である．また，最近に受信した部分解を $s_j^{last} \in S_{i,j}$ で表す．$S_{i,j}$ に対する評価値の境界は次のように表される．ここで $\simeq$ は 2 つの部分解が矛盾しないことを示す．

$$UB_{i,j}(s) = \begin{cases} \infty & S_{i,j} = \phi \\ u(s_j^{last}) & S_{i,j} \neq \phi, \\ & \{s' | s' \simeq s, s' \in S_{i,j}\} = \phi \\ \max\limits_{s \simeq s_j, s_j \in S_{i,j}} u(s_j) & otherwise \end{cases} \tag{1}$$

$$LB_{i,j}(s) = \begin{cases} \max\limits_{s \simeq s_j, s_j \in S_{i,j}} u(s_j) & \{s' | s' \simeq s, s' \in S_{i,j}\} = \phi \\ 0 & otherwise \end{cases} \tag{2}$$

ノード i と直接接続する祖先ノードと制約辺に対する評価値の合計を $\delta_i(s)$ で表す．$s \in S_i$ についての評価値の上界 $UB_i(s)$, $s \in S_i^-$ についての評価値の上界 $UB_i^-(s)$，は次式で表される．下界も同様に表される．

$$UB_i(s) = \delta_i(s) + \sum\nolimits_{j \in C} UB_{i,j}(s) \tag{3}$$

$$UB_i^-(s) = \max_{d_i \in D_i} \{UB_i(s \cup \{(x_i, d_i)\})\} \tag{4}$$

最良優先の順序で次の解 $s \in S_i^- \backslash S^{-sent}$ を送信するための条件は，次式で表される．次式の条件を満足するとき，ノード i は s と $u(s) = LB_i^-(s)$ を p_i に送信する．

$$\begin{aligned} &\exists s \in S^- \backslash S^{-sent}, \forall s' \in S^- \backslash (S^{-sent} \cup \{s\}), \\ &LB_i^-(s) \geq UB_i^-(s') \end{aligned} \tag{5}$$

pseudo tree の根ノード r では，最初の最良解が最適解である．ノード r は最適解を決定し，子ノードに通知する．他のノードは同様に親ノードの最適解に対応する最適解を決定し，子ノードに通知する．

ODPOP[5] は pseudo-tree に従って最良優先探索を行なう分散制約最適化手法である．この手法は次の 2 つの段階からなる．最初の段階では，各ノードは次の最良解を子ノードに問い合わせ (ASK メッセージ)，子ノードは親ノードに部分解と評価値を通知する (GOOD メッセージ)．この処理の反復によりボトムアップに最適解と評価値が集計される．次の段階では，最適解がトップダウンに決定される (VALUE メッセージ)．

3. 提案手法

3.1 部分解の導出

部分解の評価値のボトムアップな計算においては，各ノードは，祖先ノード間の変数値の組合せの評価値についての知識を持たずに最良解を選択する．このため，祖先ノードでの解の評価後に，広範囲にわたるバックトラックが生じる機会が増加する．そこで，各ノードにおいて，より小さいサイズの部分解を導出する．すなわち，$u(s^*) = u(s)$ であるが一部の祖先ノードの変数を含まない部分解 s^* を最良解として導出し親ノードに通知する．導出された部分解で変数を省略することにより，上位ノードでより多くの解と整合するため，バックトラックが削減される．

3.2 メッセージの同期の制御

ASK/GOOD メッセージの厳密な交換は処理の遅延の原因となる．一方で，ASK メッセージを用いず，GOOD メッセージを常に連続的に送信することは必ずしも効率的ではないと考えられる．そこで，各ノードは ASK メッセージに次の最良解の候補を付加して子ノードに送信し，子ノードはその解と整合する解を送信するまで，GOOD メッセージを連続的に送信する．

3.3 誤差の許容

近似的な解法として，各ノードでの解の評価に誤差を許容する．誤差の境界の範囲を b とするとき，誤差を許容する解 $s^{*\prime}$ と評価値 $u'(s^{*\prime})$ は，$u(s^{*\prime}) \geq u'(s^{*\prime}) \geq \max\{u(s^{*\prime}) - b, 0\}$ で表される．

4. 評価

提案手法の有効性をシミュレーションにより評価した．例題として 3 値変数の問題を用いた．制約の評価値は 1 から 10 のランダムな値とした．ノード数 n に対して，制約密度 $d = 1.125$ により $n \cdot d$ の制約辺を生成した．従来手法 odpop(sync) に 3.1 の手法 smallslt, 3.2 の手法 sync ctl, 3.3 の手法 err bounds を順に追加した手法を比較した．誤差のパラメータは $b = 1, 2$ とした．メッセージ数とサイクル数の例を図 1, 2 に示す．この結果においては，誤差無い場合の部分解の導出の効果は比較的小さい．これは，最適化問題の場合は評価値の組み合わせの数が多く，部分解が導出できる場合が少ないためであると考えられる．一方，誤差を許容する場合 (err bounds 1,2) の場合は，誤差が無い場合より部分解の導出の機会が増加すると考えられる．ASK メッセージと GOOD メッセージの同期を制御する場合は，ASK メッセージの数が減少する．また，メッセージ交換のオーバヘッドが減少するため，メッセージサイクル数が減少する．また，誤差を許容した場合は，誤差のパラメータの増加に伴いメッセージサイクル数が減少した．

5. まとめ

本論文では，pseudo-tree に基づく最良優先探索を行なう分散制約最適化のための，探索効率改善手法を提案した．より詳細な評価，および実際的な問題への適用は今後の課題である．

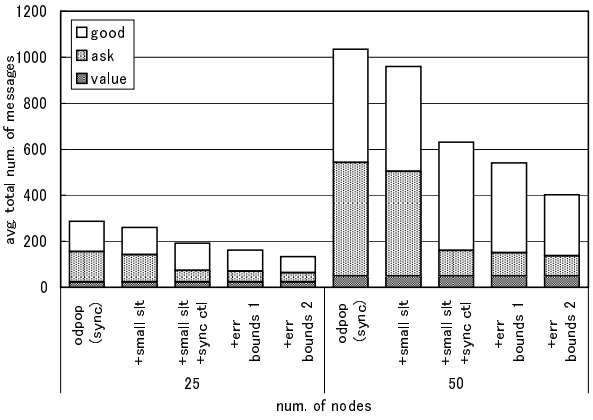

図 1: メッセージ数

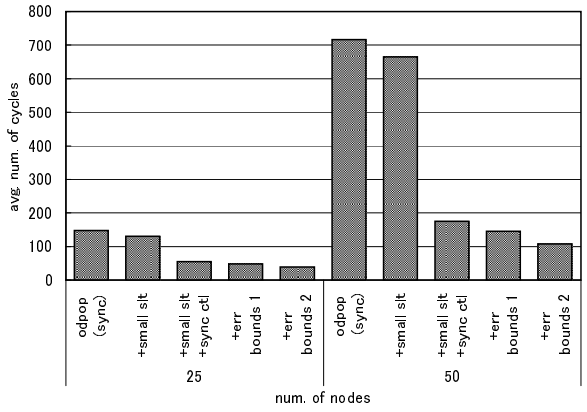

図 2: メッセージサイクル数

参考文献

[1] J. Liu and K. Sycara, “Exploiting problem structure for distributed constraint optimization”, *Proc. 1st Int. Conf. on Multiagent Systems*, San Francisco, CA, USA, pp.246-253, 6/95.

[2] K. Hirayama and M. Yokoo, “Distributed Partial Constraint Satisfaction Problem”, *Proc. 3rd Int. Conf. on Principles and Practice of Constraint Programming*,

[3] P. J. Modi, W. Shen, M. Tambe and M. Yokoo, “Adopt: asynchronous distributed constraint optimization with quality guarantees”, *Artif. Intell.*, V161, N1-2, pp.149-180, 1/05.

[4] A. Petcu, B. Faltings, “A Scalable Method for Multi-agent Constraint Optimization”, *Proc. 9th Int. Joint Conf. on Artificial Intelligence*, pp.266-271, 8/05.

[5] A. Petcu and B. Faltings, “O-DPOP: An algorithm for Open/Distributed Constraint Optimization”, *Proc. Int. Proceedings of the National Conference on Artificial Intelligence*, pp.703-708, 7/06.