F-042

非同期分散最適化アルゴリズムに基づく負荷分散のための探索手法

A Search Method for Load Distribution base on Asynchronous Distributed Optimization Algorithm

橋本 大樹 † 松田 充敏 † 能登 正人 †
Daiki HASHIMOTO Mitsutoshi MATSUDA Masato NOTO

1. はじめに

人工知能分野における様々な問題を定式化する枠組である制約最適化問題 (Constraint Optimization Problem: COP) の変数を自律的に動作，協調する複数のエージェントに分散させたものを分散制約最適化問題 (Distributed COP: DCOP) という [1]．DCOP の解は質またはコストの程度によって特徴づけられる．近年，DCOP の完全解法として深さ優先探索を基にした非同期分散最適化 (Asynchronous distributed optimization: Adopt) 手法が提案されている [2]．Adopt は，それまでの手法と比較して効率の良さ，必要とされる記憶領域などの利点が証明されている．上記手法の利点の一因は，全体の制約違反を最小にする解を導出するアルゴリズムによる．しかし現実的問題を考えた場合には，全体の制約違反を最小に抑えることよりも，局所的な負荷を考慮して最適化を行ったほうが適した問題もある．例えば，局所負荷を軽減することができれば，応用次第でネットワークのサーバなどにおいて 1 カ所に負荷をかければ処理の遅延等の恐れがある場合に対しても有効であると考えられる．

本稿では，Adopt に新たなメッセージを追加し，近傍エージェント間でコスト情報の交換をすることにより，既存のアルゴリズムの利点を生かしつつ，全体の最適化よりも局所的な負荷を考慮した分散探索手法を提案する．

2. 非同期分散最適化手法

Adopt は，前処理としてエージェント間の親，子の関係を定義した深さ優先探索木を形成する．図 1 は制約グラフを深さ優先探索木に変換したものである．図より x_1 は根エージェント，x_1 の子は x_2，x_2 の子は x_3 と x_4 とする．また x_1 の下位近傍エージェントは x_2 と x_3 で，x_2 の下位近傍エージェントは x_3 と x_4 とする．各エージェントは値域 D_i について $d_i \in D_i$ の値をとるような単一の変数 x_i を持ち，近傍エージェントとの制約および関数 $f(d_i, d_j)$ によって決定された子エージェント x_j までのコストのみを知る．親子間のコストである局所コストを含めた情報は VALUE，COST，THRESHOLD，

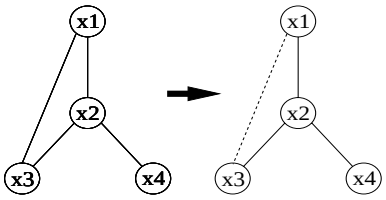

図 1: 制約グラフ

TERMINATE の 4 種類のメッセージ通信によって伝達され，エージェントの変数の変更等を実行させる．最終的に求める最適解は，全体の局所コストの総和が最小となる解である．

3. 提案手法

本研究では Adopt の利点を生かしつつ一点のみにコストの負荷がかかり過ぎないように改善する．

Adopt ではエージェント x_i は近傍エージェントとの間でメッセージを交換し，変数の値を割り当てる．中でも子 x_j から親 x_i へ送信されるメッセージを COST メッセージといい，子エージェント x_j 以下までの総コストと，子 x_j の変数値を送信する．子から親に送信されるメッセージはこれだけなので，親は下位近傍エージェントとの間のコストを知ることができない．そのため親であるエージェントにとって，自エージェントと近傍エージェントとの間のコストも，自エージェントのサブツリー内で距離の離れた場所にあるエージェントとの間のコストも，COST メッセージにより自エージェントに伝えられてくるサブツリーのコストの和が同一であれば，親であるエージェントはその違いを認識することができない．

以上の事柄に着目して，提案アルゴリズムでは上位近傍エージェントへ自分のコストを送信するメッセージを追加することで，上位近傍エージェントが近傍であるエージェントとの間に持つコストを知ることを可能とする (図 2)．

このメッセージを受信したエージェントはメッセージ内の情報をキャッシュに蓄積し，コスト計算時に参照することによりコストに重みを考慮する．つまり下位近傍

†神奈川大学工学部電子情報フロンティア学科, Department of Electronics and Informatics Frontiers, Kanagawa University

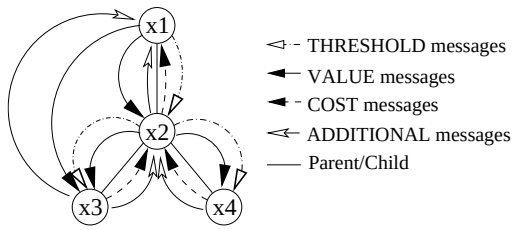

図 2: 提案した協調関係

エージェントとの間のコストが多いほどコストに重みが考慮されることになる．注意として，コストに考慮される重みは問題の規模，コストに設定する数値によって変化させる．これを含めメッセージ受信時の処理を以下にまとめる．なお，処理中に使用されている各用語は文献 [2] に従うものとする．

- TRESHOLD **メッセージ受信時**

  $context$ と $CurrentContext$ が一致した場合は $threshold = t$ とし，$threshold < LB$ ならば $threshold = LB$ とし，$threshold > UB$ ならば $threshold = UB$ とする．その後バックトラック処理を行う．

- TERMINATE **メッセージ受信時**

  受信した TERMINATE メッセージを記録し，$CurrentContext = context$ とする．その後バックトラック処理を行う．

- VALUE **メッセージ受信時**

  TERMINATE メッセージを受信していなければ，上位エージェントの解を $CurrentContext$ に記録する．次に $context$ と $CurrentContext$ が一致しなければ，子エージェントのキャッシュを初期化する．その時 $threshold < LB$ ならば $threshold = LB$ とし，$threshold > UB$ ならば $threshold = UB$ とする．その後バックトラック処理を行う．

- COST **メッセージ受信時**

  TERMINATE メッセージを受信しておらず，$CurrentContext$ と $context$ が一致していなければ，子エージェントのキャッシュを初期化，一致していれば子エージェントのキャッシュを上書きした後，$threshold$ と t を操作する．その後バックトラック処理を行う．

- ADDITIONAL **メッセージ受信時**

  $context$ と自エージェントの変数値が一致していない場合はメッセージを破棄し，一致している場合は下位近傍エージェントのキャッシュとして記録する．

4. 評価方法

提案手法を評価するために，隣接した領域では同じ色にならないよう，それぞれの領域に色を割り当てていくグラフ色塗り問題を用いる．各エージェントの変数の値域である色の数は 3 色とし，隣接したエージェント間は同一色に塗りわけないような 2 項制約を持つこととする．また各エージェントにおけるリンク密度 d は充足可能な問題が比較的多い $d = 2$ の場合と，過制約となり制約違反が多くなる $d = 3$ の場合について行う．局所的な制約違反について着目するため提案手法と既存手法において，それぞれ各場所に発生した制約違反の集合数の最大値を比較し，また全体での最適化についての評価も行う，

注意として，エージェント数が少ない場合では全体の最適化が局所負荷を考慮した場合においても最適となる場合が多いと考えられる．そのエージェントの数や，リンク密度を大きくし評価を行う．

5. おわりに

本研究では，DCOP の完全解法である Adopt において，制約違反が一点に集中しないための探索手法を提案した．今後の課題として，シミュレーション評価を行い，新たに提案したメッセージを受け取った後のコスト計算における重みを実際にはどのように設定するか考察する．本手法は制約違反が多くなるような問題でないと現実的には違いが現れない可能性もあるので，エージェントにおけるリンク密度が大きく過制約の問題や，エージェント数が多くなる問題なども扱う必要がある．また近傍エージェントではないが十分近いエージェントにも，ある程度メッセージを送信することにより，規模の大きな問題に対応する必要があると考えている．

謝辞

本研究の一部は，文部科学省ハイテクリサーチセンター整備事業の助成金によって行われた．

参考文献

[1] K. Hirayama and M. Yokoo: Distributed Partial Constraint Satisfaction Problem, Principle and Practice of Constraint Programming, Lecture Notes in Computer Science, Vol. 1330, pp. 222–236 (1997).

[2] P. J. Modi, J. W. Shen, M. Tambe and M. Yokoo: Adopt: Asynchronous Distributed Constraint Optimization with Quality Guarantees, Artificial Intelligence, Vol. 161, No. 1–2, pp. 149–180 (2005).