3E2-05

動的な分散制約最適化問題のための基本的な枠組みの提案

A dynamic framework using distributed constraint optimization method

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

A framework for dynamic ditribted constraint optimization probrem (DCOP) is proposed. The framework is based on an algorithm for DCOP which uses depth first search (DFS) tree of constraint network. In the framework, each agent can observe related constraints. Agents repeat cooperative processings as follows. Constraint networks are constructed. Pseudo DFS trees of the constaint networks are constructed by bottom up processing. Search processings using these trees are performed. Each processing is performed asynchronously. Effeciency of this framework is evaluated by simulations.

1. はじめに

分散制約最適化問題 (DCOP) [1][2][3][4] は，分散制約充足問題 (DCSP)[5] を最適化問題に拡張したものであり，分散協調処理のための基礎的な探索問題として研究されている．DCSP,DCOP のための探索アルゴリズムとして，制約網に対する深さ優先探索木により順序付けられたエージェントを用いる手法が提案された [6][7]．厳密な深さ優先探索木でなく，ボトムアップに生成されたサブツリー間に制約辺を持たない木であっても，これらの手法は適用が可能であることが示されている．このような擬似的な深さ優先探索木も制約網に対する生成木とみなすことができる．

動的に変化する制約充足問題として，制約充足/最適化問題の系列を順に解く，動的制約充足/最適化問題が提案されている [8][9][10]．DCSP, DCOP の動的環境への適用のためには，探索のみではなく，問題の設定，解の決定を含めたサイクル動作を分散アルゴリズムによって実現する必要がある．これらの処理は，特に探索手法が生成木を用いる場合，比較的容易に追加することができる．

本稿では，分散制約最適化解法である Adopt[7] に基づく動的な問題のための枠組み提案する．各ノード (エージェント) は制約に関する局所的な知識を持つものとし，制約網の構築から解の決定までの処理が 3 つの階層により構成される．これにより，関連するノードとの通信により制約網を構築し，制約網に応じた擬似的な深さ優先探索木をボトムアップに構築し，構築された木を用いた分散制約最適化アルゴリズムにより問題を解く，3 段階の処理が反復される．

2. 動的分散制約最適化問題

動的制約充足/最適化問題は，基本的には制約充足/最適化問題の系列として定式化され，その解法は問題の系列を順に解く．動的分散制約充足/最適化問題 (DyDCSP/DyDCOP) は DCSP,DCOP を動的問題へ適用したものであり，変数および制約が複数のノード (エージェント) に分散して配置され，分散アルゴリズムによって問題が解かれる．本稿では， DyDCOP の基本的な例題を想定する．ノードの集合を A とする．各ノード $i \in A$ は単一の変数 x_i を持つ．x_i の値はノード i のみが変更する．

連絡先: 名古屋市昭和区御器所町　名古屋工業大学情報工学科

問題の系列を $P = \{p_0, \cdots, p_s, \cdots\}$ とする．$p_s = (C_s, F_s)$ であり，C_s ,F_s はそれぞれ制約，制約に対する評価関数の集合である．現在の問題 p_s について，x_i は他ノード $\{j, \cdots\}$ の変数 $\{x_j, \cdots\}$ と制約 $c \in C_s$ により関係する．制約 c に対応する評価関数 $f \in Fs$ により，変数値の割り当てについてのコストが評価される．大域コストはすべての制約に対するコストの総和であり，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．現在の問題 p_s はあるスケジュールに従って次の問題 p_{s+1} へと変化する．適切な解法により，問題が変化するまでに何らかの解が得られる必要がある．

本稿では，各ノードは大域的に無矛盾な変数 (ノード) 名およびその値域についての知識を持つが，各制約は関連するノードのいずれかが観測できるものとする．従って, 各ノードは問題を解く前に関連ノードと無矛盾な制約の知識を持つ必要がある．

3. 制約の観測と制約網の構築

現在ノード i が知る制約および関連ノードの記述を Sts_i とする．Sts_i は次の要素から構成される．

C_i^{out}, F_i^{out} : 観測した制約, 評価関数の集合

C_i^{in}, F_i^{in} : 受信した制約, 評価関数の集合

Deg_i^{in} : 受信した次数の集合

Clk_i^{in} : 受信した時刻の集合

C_i, F_i : 制約, 評価関数の集合

Nbr_i : 近傍ノードの集合

C_i^{out}, F_i^{out} はノード i が観測した制約および評価関数の集合であり，その変化は相手ノード j に通知されなければならない．起こりうる変化は，(1) 新しい制約を観測する，(2) 制約の内容の変化を観測する，(3) 制約が観測されなくなる, のいずれかである．

C_i^{in}, F_i^{in} はノード i が他ノードから受信した制約および評価関数の集合である．ノード i は観測および受信による制約と評価関数の情報を統合し C_i, F_i を得る．

Nbr_i は C_i に含まれる制約によりノード i と関連する近傍ノードの集合である．次数 $|Nbr_i|$ の変化は近傍ノードに通知されなければならない．Deg_i^{in} は近傍ノードから受信した次数の集合である．

ノード i は現在の状態を識別するために時刻 clk_i を用いる．clk_i はノードの知る制約および関連ノードの状態が変化したとき増加される．clk_i は各メッセージに付加される．Sts_i について近傍ノードに通知されるメッセージ (STS メッセージ) は (1)C_i^{out} に含まれる制約により関連する近傍 j に，制約の追加および変化，または次数の変化を通知する場合, (2)C_i^{out} に含まれないが C_i^{in} に含まれる制約により関連する近傍 j に次数の変化を通知する場合, (3) 近傍 j に関する制約が C_i^{out} に含まれなくなったことを j に通知する場合，また j に関する制約が C_i^{in} にも含まれない場合は近傍からの削除を意味する, (4) 状態に変化が無いが，新しい時刻の通知を受けたため，時刻の更新のみ近傍 j に通知する場合, のいずれかである．

$j \in Nbr_i$ についての時刻を全て受信し，ノード i の clk_i および近傍から受信した時刻が全て等しければ，ノード i は局所的に状態が安定したものとする．このときの Sts_i をノード i の部分的な問題 Sts_{i,clk_i} として識別する．

4. ボトムアップな木の生成

各ノードはある時刻の問題に対し，関連ノードとともにボトムアップに擬似的な深さ優先探索木を構築する．このような木の生成方法については関連研究でも示されている．本手法では，木の生成を前述の制約網の構築に対する大域停止検出と兼ねあわせる．ノード i の知る現在の時刻 clk_i に対する，木構造の記述を $Tree_{i,clk_i}$ とする．ノード i は最新の時刻に対する $Tree_{i,clk_i}$ のみを保持する．$Tree_{i,clk_i}$ は Sts_{i,clk_i} の要素の複製 $C_i, F_i, Nbr_i, Deg_i^{in}$ および次の要素から構成される．

$Nbr_i^{upper}, Nbr_i^{lower}$：上位, 下位近傍ノードの集合

Deg_i：関連ノードの次数の集合

Rel_i：関連ノードの制約関連数の集合

$Children_i$：子ノードの集合

$Ancestors_i$：祖先ノードの集合

$parent_i$：親ノード

$num\_of\_leaf_i$：サブツリーの葉ノードの数

ノード i は現在の時刻 clk_i に対する $Tree_{i,clk_i}$ を保持するため，まだ Sts_{i,clk_i} が得られておらず，これらの複製が使用できない場合がある．このとき，Sts_{i,clk_j} が得られるまではノード i は $Tree_{i,clk_i}$ に関して，後述の TREE メッセージ受信以外の処理は行わない．現在の時刻が増加したとき，$Tree_{i,clk_i}$ は破棄され，次の時刻の $Tree'_{i,clk_i}$ が準備される．

$Nbr_i^{upper}, Nbr_i^{lower}$ は Nbr_i および Deg_i^{in} にもとづいて最大次数ヒューリスティクスにより分類される．Rel_i はノード i 以下のサブツリーから各上位関連ノードへの制約辺の数の集合であり，各要素の値は Nbr_i^{upper} および各子ノード j から通知された Rel_j の要素から合計される．また，$num\_of\_leaf_i$ はサブツリーにおける葉ノードの数であり，後述の探索の終了処理で用いるため，木の構築と同時に集計される．各ノード i は，条件 $rel_i = |Nbr_i^{lower}|$ の成立によりサブツリーを確定する．

サブツリーが確定したノード i は TREE メッセージ (TREE, $parent_i$, $Ancestors_i$, Deg_i, Rel_i, $num\_of\_leaf_i$, clk_i) を $j \in Ancestors_i$ に送信し，受信ノード j では各情報を更新する．現在の $Tree_{j,clk_j}$ より古い時刻のメッセージは無視される．葉ノードではサブツリーは速やかに確定され，根ノードまで順にサブツリーが確定される．木の生成過程で生じる，制約辺が存在しない親子ノードについて，矛盾を解消するために冗長な制約辺が挿入される．

5. 深さ優先探索木に基づく分散探索

提案手法は基本的には分散制約最適化解法の Adopt[7] を反復実行する枠組みであり，最上位層の処理である解探索はこの解法に基づく．このアルゴリズムはノードが事前に深さ優先探索木により順序付けされていることを前提とし，分枝限定法を基礎とする非同期探索を行う完全なアルゴリズムである．また，大域的に解を決定し探索を終了することができる．

提案手法では，速やかに近似解を得るために，適当な探索スケジュールに基づき上界の探索を優先する．ある上界値が得られている時点で，根ノードはコストの探索を中断し解の決定に移行する．決定された解に基づき，次回の探索の初期解を決定する．問題が変化する場合には，適切な写像により次の問題に応じた解に変換する．問題が変化しない場合は前回の暫定的な解からの改善が継続される．ノード i の知る clk, slt の問題に対する，Adopt のための記述を $Adpt_{i,clk,slt}$ とする．$Adpt_{i,clk,slt}$ は，Adopt のための情報に加えて，$Tree_{i,clk}$ の要素の複製 C_i, F_i, Nbr_i, Nbr_i^{upper}, Nbr_i^{lower}, $Children_i$, $parent_i$, $num\_of\_leaf_i$ を持つ．現在の問題に対する Adopt アルゴリズムの記述を $Adpt_i^{cur}$ により表す．また，各メッセージに clk, slt の情報を追加し，どの問題に対応するメッセージであるかを識別する．現在の問題 $Adpt_i^{cur}$ よりも古い問題のメッセージを受信した場合，そのメッセージは無視される．

動的な問題へ追従するために，探索アルゴリズムは現在の時刻の木に対して実行される．時刻の増加を検出したノードは，現在の木および探索を放棄し，次の木および探索のための記述を準備する．各ノードの保持する木が次の時刻のものに移行することにより，前回の木および探索のための情報は自然に失われ，新しい木に対する探索が開始される．

しかし，実際には，無矛盾な解を得るために解の決定は同期されなければならないと考えられる．そこで，2 相コミットメントに類する方法で解を決定する．解の決定の 1 相目で最適解を確保し，全ての葉ノードまで解が確保された後，2 相目で実際に解を決定する．1 相目の処理の途中で，根ノードが木の消失を検出したとき，決定中の解は 2 相目で取り消される．解の決定中に，下位ノードの処理待ちとなるノードは，次の問題に対する探索処理を準備する必要がある．また，動的な問題においては前回の解のコストを初期解とするなど，前回の探索の結果を確保する必要がある．このため Adopt アルゴリズムに関する記述は，現在の問題に対する $Adpt_i^{cur}$ に加えて, 解の決定中の $Adpt_i^{sub}$, 前回の探索結果の $Adpt_i^{cmt}$ の 3 つを用いる．

6. 評価

提案手法について，計算機実験による評価を行った．初期の検討として，現在の問題は一定期間の探索の後に次の問題に変化するものとし，問題が変化するまでに得られた解のコストを比較した．比較対象として，提案手法 (dadpt)，提案手法と同様に上界優先の探索を行うが全ノードが線形な順序で配置された Adopt アルゴリズム (gadpt)，近似解法である分散 Greedy Repair 法 [4](grpar) を用いた．三色のグラフ彩色問題の系列からなる動的問題について実験した．システムはサイクル動作するものとし，各サイクルで各ノードはメッセージ受信，内部処理，送信を行うものとした．

問題が変化するまでのサイクル数 500，変化する制約の割合 10%，ノード数 40，ノード数に対する制約辺の密度 d について各系列の 0 番目から 3 番目 (p_0〜p_3) までの最良解のコストの平均と最適解の数を表 1 に示す．$d = 1, 2$ の問題については提案手法の効果が得られた．一方，$d = 3$ の問題については，

表 1: 問題が変化するまでに得られた最良コスト

d	alg.	P0		P1		P2		P3	
		mean #opt.	(var.)	m. #o.	(v.)	m. #o.	(v.)	m. #o.	(v.)
1	gadpt	5.6 1	(5.68)	3.7 10	(7.28)	2.8 28	(8.29)	2.3 38	(7.03)
	grpar	3.0 2	(2.49)	1.1 29	(0.85)	0.6 53	(0.58)	0.4 72	(0.41)
	dadpt	0.0 100	(0.00)	0.0 100	(0.00)	0.0 100	(0.00)	0.0 100	(0.00)
2	gadpt	17.8 0	(18.33)	16.8 0	(20.03)	16.5 0	(19.89)	16.3 0	(18.14)
	grpar	11.7 0	(7.44)	6.8 0	(4.31)	4.9 0	(3.25)	4.0 1	(2.30)
	dadpt	3.3 11	(4.67)	1.6 36	(2.74)	1.1 62	(1.71)	1.1 63	(1.65)
3	gadpt	32.0	(38.46)	31.2	(36.79)	30.3	(32.34)	30.1	(30.54)
	grpar	22.6	(10.99)	16.0	(8.97)	13.0	(5.17)	11.2	(4.59)
	dadpt	19.6	(18.47)	18.0	(18.24)	17.4	(16.41)	16.6	(15.25)

表 2: 生成される木の規模

n	d	#root nodes		max. #depth		max. #nodes	
		mean	(var.)	mean	(var.)	mean	(var.)
10	1	2.1	(0.30)	4.4	(0.40)	8.7	(0.64)
	2	1.0	(0.02)	6.8	(0.85)	10.0	(0.02)
	3	1.0	(0.00)	8.3	(0.56)	10.0	(0.00)
20	1	3.5	(1.08)	6.4	(0.79)	16.7	(2.56)
	2	1.1	(0.10)	12.1	(1.58)	19.9	(0.10)
	3	1.0	(0.00)	14.3	(1.58)	20.0	(0.00)
40	1	7.5	(3.78)	9.4	(1.81)	32.3	(4.90)
	2	1.5	(0.48)	19.0	(2.47)	39.5	(0.48)
	3	1.1	(0.20)	24.9	(4.12)	39.9	(0.20)

提案手法のコストは近似解法のコストより大きくなった．これは提案手法がバックトラックに基づく探索を行うため，制約密度の高い問題では不利であるためであると考えられる．

各問題について生成された木の規模を表 2 に示す．制約密度の低い問題では，制約網が複数の連結成分から構成され，それぞれについて生成された木により複数の探索処理が並行して行われる．ただし，問題の生成方法のため，木が複数の同規模の連結成分に分離する場合は少なく，孤立した根ノードが多数生成される場合が多い．

7. まとめ

本稿では，深さ優先探索木に基づく分散制約最適化手法を，動的な問題に適用するための基本的な枠組みを示した．また，実験結果により，特に制約密度が低く並列性が高い問題では，提案手法が有効であることを示した．解の安定性や，探索結果の再利用による効率化などを含めた実際的な問題への適用の検討は今後の課題である．

参考文献

[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. Lemaitre, and G. Verfaillie: An Incomplete Method for Solving Distributed Valued Constraint Satisfaction Problems, In Proc. AAAI97 Workshop on Constraints and Agents, 1997.

[5] 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.

[6] Y. Hamadi: Interleaved search in distributed constraint networks, International Journal on Artificial Intelligence Tools, vol.3, no.4, pp.167–188, 2002.

[7] P. J. Modi, W. Shen, M. Tambe and M. Yokoo: Adopt: asynchronous distributed constraint optimization with quality guarantees, Artificial Intelligence, vol.161, Issue.1–2, pp.149–180, 2005.

[8] R. Dechter and A. Dechter: Belief Maintenance in Dynamic Constraint Networks, Proc. 7th National Conference on Artificial Intelligence, pp.37–42, 1988.

[9] G. Verfaillie and T. Schiex: Solution Reuse in Dynamic Constraint Satisfaction Problems, Proc. 20th National Conference on Artificial Intelligence, pp.307–312, 1994.

[10] 服部宏充, 伊藤孝行, 新谷虎松, ”動的重み付き制約充足問題に基づくナーススケジューリングにおける解の安定性のため第 18 回人工知能学会全国大会予稿集 CD-ROM, 2004.