F-029

分散制約最適化手法を用いたコアロケーションスケジューリングへの適用

Co-allocation scheduling using distributed constraint optimization technique

行方 裕紀* 松井 俊浩* 松尾 啓志*
Yuki Namekata Toshihiro Matsui Hiroshi Matsuo

1 はじめに

グリッドコンピューティング技術の進展により, 広域ネットワーク上にあるコンピュータ資源を用いた大規模科学技術計算が実現可能となった. このような広域・大規模グリッド計算環境においては分散した計算・通信資源を同時に確保する事（コアロケーション）[1] が必要である. 近年, このような環境における複数スケジューラによる分散スケジューリングの研究が行われている. 資源スケジューリングは一種の制約充足・最適化問題として扱うことができる. そこで, 本研究では, 非同期分散環境における協調問題解決のモデルである分散制約最適化問題（Distributed Constraint Optimization,DCOP）の, コアロケーションのための分散スケジューリングへの適用を提案する. ここでは, 複数のローカルスケジューラが分散協調を行うモデルを想定する. DCOP を探索アルゴリズムとして, 制約網に対する擬似的な木（Pseudo tree）による変数順序付けを用いた解法が提案されている. 本研究では, このような擬似木を用いた動的計画法に基づくアルゴリズム [2] の適用を検討する. この手法は完全性を持ち, 最適解を得られる解法である. メッセージ数が線形であり通信回数が少ないが, ノード間で交換されるメッセージのサイズが木の幅に対して指数関数的に増加する. そこで, 擬似的な木の幅を制限することを考慮した問題の緩和手法を提案する.

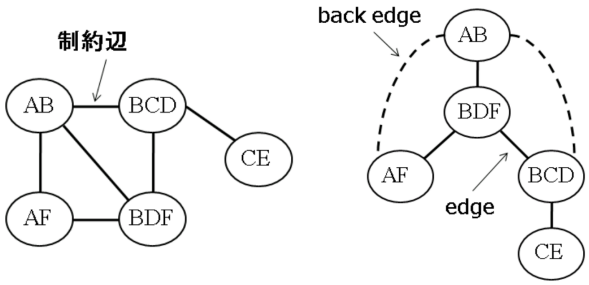
2 コアロケーションシケジューリング問題の分散制約最適化問題としてのモデル化

2.1 分散制約最適化問題 (DCOP)

分散制約最適化問題 (DCOP) は, 制約最適化問題を分散探索問題に適用したものであり, 従来の分散制約充足問題では記述不能な問題を扱うものである. ノード (エージェント) の集合を A とする. 各ノード $i \in A$ は変数 x_i の領域 D_i の値 d_i をとる. C を制約の集合,F を制約に対する評価関数の集合とする.x_i は他ノード j の変数 x_j と２項制約 $c_{i,j} \in C$ により関係する. 制約 $c_{i,j}$ に対応する評価関数 $f_{i,j} \in F$ により, 変数値の割当 $\{(x_i, d_i), (x_j, d_j)\}$ についての利得が評価される. 大域的な評価値はすべての制約に対する利得の評価の総和であり, 大域的な評価値を最大にする変数値の割当が最適解である.

2.2 コアロケーションスケジューリング問題

コアロケーションスケジューリング問題では, リソース集合 $R := \{R_1, ..., R_n\}$, ジョブ集合 $\epsilon := \{E^1, ..., E^k\}$,



(a) EAV で作成した制約網 (b) Pseudo tree

図 1: グラフ構造

時間領域 $\tau := \{1, ..., T\}$, の情報が与えられる. ジョブ E^k には, ジョブの必要とするリソース $A^k(\subset R)$, 時間枠 L^k, ジョブの価値 V^k, の３つの情報が, 関連付けられている. コアロケーションでは分散した資源を同時に確保しなければならず, ジョブの必要とするクラスタ (リソース) がジョブに割当てられなければならない. 一方で, 同じクラスタ (リソース) を同じ時間に使うことはできない. このような制約を考慮したスケジューリングが必要となる. コアロケーションスケジューリング問題では, 制約を充足し, かつ, 時間領域 τ の中でスケジュールされたジョブの価値の和が最大になる解を探すことが目的である.

2.3 制約網の作成

制約網の作成には, イベントスケジューリング問題を制約最適化問題として形式化する EAV(Events as Variable)[3] を用いた. EAV では, ジョブごとに変数を作り, ジョブの開始時間を変数の領域とする. 制約辺は, 共通するリソースを必要とするノード間に作成される. また, ノード間には評価関数が設けられ, 大域利得を最大にする変数の割当てを考える. 図 1(a) に, 例として, ジョブ=$\{E^1, E^2, E^3, E^4, E^5\}$ があり, 各々の必要リソース=$\{AB, AF, BCD, BDF, CE\}$ の場合に,EAV で制約網を作成したものを示す例の説明をする.

2.4 Pseudo tree

Pseudo tree は, 制約網に含まれるノードの半順序関係を与える構造である. 各ノード (変数)[1]は木の深さによって順序付けされ, 制約辺は, 木の枝となる木辺（tree edge）とそれ以外の辺である後退辺（back edge）に分類される. Pseudo tree は, サブツリー間に制約辺がないため, 探索手法の効率化に役立つことが知られている.

*名古屋工業大学 Nagoya Institute of Technology

[1]本研究では便宜上, 各ノードは一つの変数のみを持ち, ノード＝変数とみなす

図 1(a) の制約網に対する Pseudo tree を図 1(b) に示す. ノード x_i と制約辺で関連するノードを次の表記により示す.

- $P(x_i)$：木辺で x_i と関連する x_i の親ノード
- $PP(x_i)$：　x_i を根とするサブツリーと, 制約辺 (木辺・後退辺) で関連する x_i の祖先ノードの集合
- $C(x_i)$：木辺で x_i と関連する x_i の子ノードの集合
- $PC(x_i)$：制約辺 (木辺・後退辺) で x_i と関連する子孫ノードの集合

2.5 DPOP アルゴリズム

DPOP アルゴリズム [2] は Pseudo tree を用いる, 分散制約最適化アルゴリズムである．DPOP アルゴリズムは,2 段階の処理からなる．各段階ではそれぞれ,UTIL メッセージ・VALUE メッセージの伝搬による処理が行われる.

(1) 大域的に最適な評価値の計算

UTIL メッセージの伝搬による処理では, まず, 各ノード x_i において,x_i を根とする部分木の解の評価値のベクトル $Util_{x_i,P(x_i)}$ が次式により計算される.

$$Util_{x_i,P(x_i)}(d_{p(x_i)}) = \begin{cases} \max_{d_i \in D_i} f_{i,P(x_i)}(d_i, d_{p(x_i)}) & C(x_i) = \phi \\ \max_{d_i \in D_i}\{f_{i,P(x_i)}(d_i, d_{p(x_i)}) & otherwise \\ \quad + \sum_{x_k \in C(x_i)} Util_{x_k,x_i}(d_i)\} & \end{cases}$$

評価値のベクトルは UITL メッセージとして, 葉ノードから根ノードに向け木辺に従って, ボトムアップに送信される．このような計算は動的計画法に基づくものがあるが,Pseudo tree には後退辺が存在するため, 各ノード x_i について, ノード x_i を根とする部分木と, 親ノード $P(x_i)$ のみではなく, 後退辺で関連する全ての上位ノードの変数の値領域の組に対する評価値の計算が必要になる．このため, ノード x_i で計算される評価値 $Util_{x_i,P(x_i)}$ はハイパーキューブになり, ノード x_i を根とするサブツリーと制約辺で関連する x_i の祖先ノードの数の増加に伴い UTIL メッセージサイズが指数関数的に増加する. また, 一番大きい UTIL メッセージの関連ノード数を木の幅と呼ぶ.

(2) 最適解の決定

VALUE メッセージの伝搬のよる処理では, 全てのノード x_i に対して最適な変数値 d_i^* が次式により決定される.

$$d_i^* = \begin{cases} argmax_{d_i \in D_i} f_{i,P(x_i)}(d_i, d^*_{p(x_i)}) & C(x_i) = \phi \\ argmax_{d_i \in D_i}\{f_{i,P(x_i)}(d_i, d^*_{p(x_i)}) & otherwise \\ \quad + \sum_{x_k \in C(x_i)} Util_{x_k,x_i}(d_i)\} & \end{cases}$$

最適な変数値は VALUE メッセージとして, 根ノードから葉ノードへと木辺に従って, トップダウンに送信される.

3 提案手法

DPOP では, 木の幅が増加すると UTIL メッセージのサイズが指数関数的に増える．そこで, 本論文では, 問題の緩和により, 木の幅を削減する手法を提案する. 提案手法を以下に示す.

3.1 後退辺の削除

コアロケーションでは, 同じリソースを持つジョブが同時に, 同じ変数（タイムスロット）を取ることがない. この制約を利用して, 木の後退辺の削除を行う.

ノード x_i とノード x_i の子孫ノード集合 $PC(x_i)$ で継がれた後退辺の削除を考える．この 2 つのノードの変数領域 d_{x_i},$d_{PC(x_i)}$ の論理積が空集合 ϕ となるように変数領域の割当てを行う．変数領域の割当てでは, ノード x_i に $\tau := \{1, ..., L^k\}$, 子孫ノード集合 $PC(x_i)$ に $\tau := \{L^k + 1, ..., T\}$ と変数の値域の制限を行う. 子孫ノード集合に変数を多く割当ることにより, 解の探索範囲が増える．これにより, ジョブが必要としている同じリソースが同じ時間にスケジューリングされることはなくなり, 矛盾無くスケジューリングが行われる．また, 後退辺の削除を効率良く行うために, 木の幅が一番大きいノードの祖先ノードと子孫ノードを端点とする後退辺を削除する．この操作によって,Pseudo tree の最大の木の幅を 1 減らすことができる．後述 4. の実験では, この操作を複数回行うことによって幅を規定値まで下げ, 図 2 に, 後退辺の削除のアルゴリズム WD を示す. 図 2 中では, ノード x_i が x_j の親ノードまたは祖先ノードであることを記号 $\succeq$ を用いて $x_i \succeq x_j$ により表す.

Algorithm 1: *WD - Width delete*

1:$max_{x_n \in A}\{Util_{x_n,P(x_n)} + \sum_{x_k \in C(x_n)} Util_{x_k,x_n}(d_n)\}$;
2:**if**$(x_i \succeq x_n$ **&&** $x_n \succeq x_j \in PC(x_i))$
3:**then**:$\{PC(x_i) = PC(x_i) - x_j$;
4: $PP(x_j) = PP(x_j) - x_i$;
5: $d_{x_i} \cap d_{PC(x_i)} = \phi$;$\}$

図 2: 後退辺の削除アルゴリズム WD

変数の値域を変更したノードでは値域の範囲が制限されることにより解探索の範囲が削減される．一方で, 解の精度が低下する場合がある．本研究の変数の値域の制限における戦略では, 親ノードのジョブは子孫ノード集合のジョブよりも早い時間領域に割り当てられる．これは, 戦略を変えることにより変更できる．また, ある同じ資源を必要とするジョブが多い場合, 解探索の範囲の制限は顕著に行なわれる.

3.2 問題の分割

問題を分割し, 問題を縮小することで木の幅を小さくする方法を提案する．本研究では初期の検討として問題の分割では, ジョブ数が半分になるような分割方法を用いる.

問題として, リソース集合 $R := \{R_1, ..., R_n\}$, ジョブ集合 $\epsilon := \{E^1, ..., E^k\}$, 時間領域 $\tau := \{1, ..., T\}$, の情報が与えられたとする．このジョブ集合 ϵ を, ジョブ集合 $\epsilon_1 := \{E^1, ..., E^{\frac{k}{2}}\}$ とジョブ集合 $\epsilon_2 := \{E^{\frac{k}{2}+1}, ..., E^k\}$ の 2 つの問題に分割を行なう (但し, リソース集合 R と, 時間領域 τ は変化しない). まず初めに, ジョブ集合 ϵ_1 の解の探索を行なう．次に, 任意のジョブ $E^i (\in \epsilon_1)$ と任意

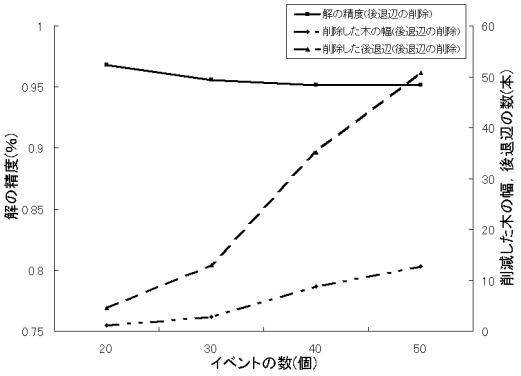


図 3: 後退辺の削除によるイベント数と解の精度の関係

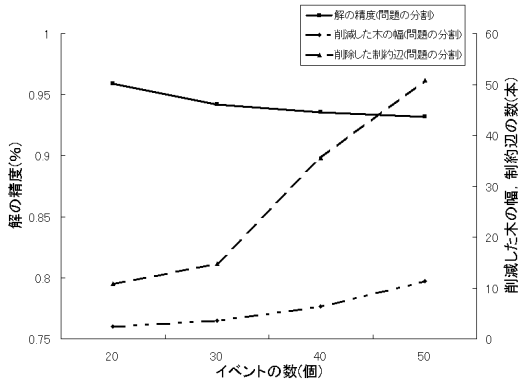


図 4: 問題の分割によるイベント数と解の精度の関係

のジョブ $E^j(\in \epsilon_2)$ が共通の資源 R_i を必要とするとき $(A^i \cap A^j \neq \phi)$, 変数の領域の制限を行なう必要がある. 変数の領域の制限では, ジョブ E^i の解を t とすると, ジョブ E^j の変数の領域を $\tau := \{1, ..., t-1, t+L^i+1, ..., L^k\}$ とし, ジョブ集合 ϵ_2 に対して解探策を行う. この手法により, 問題を分割しても同じリソースが同じ時間にスケジューリングされることはなくなり, 矛盾なくスケジューリングが行なわれる.

この手法では, 探索すべき解の組合せの範囲が削除される. 後退辺の削除と同様に, 組合せの範囲の限定により最適解が変化する場合が生じる. 2 つに分割したジョブ集合のうち, 先に解の探索を行なったジョブ集合は, 後で解の探索を行なったジョブ集合に比べて早い時間領域にジョブが割当てられる. 分割後の解探索の順番によって, ジョブ集合の時間領域の割当てが変化する. また, 使用可能な資源が少なくジョブが同じ資源を使う率が多い場合に組合せ範囲の限定が多く行なわれる.

4 シミュレーションによる評価

提案手法で述べた, 制約辺の削除, 問題の分割の有効性を検討するため, シミュレーション実験を行なった. 実験では, 複数のローカルスケジューラが協調処理を行なうモデルを想定した.

4.1 実験方法

文献 [6][7][8] を参考に, 以下の環境で実験を行なった. 初期の検討として, 問題のサイズはジョブ数が 20,30,40,50 個とした. リソース (クラスタ) の数は 15 個とした. 各ジョブには必要とするリソースは 2 個, タイムスロット数は 1〜3 となる場合を想定した. 各問題は, 単一の連結成分からなる制約網が作成されるように生成した. また, 大規模な時間単位を想定し, ノードの持つ変数領域は $\tau := \{1, ..., 20\}$ とした. 提案手法と従来手法の解の精度と UTIL メッセージのサイズを, 問題のサイズを変化させ比較した. 後退辺の削除では, 木の幅が 5(送信メッセージサイズが 1.2MB) になるまで後退辺の削除の操作を複数回を繰り返した. グラフの分割では, ジョブの数が半分になるように分割した. 評価の尺度として, 解の精度を定義した. 解は, ジョブを可能なかぎり値の小さい (順序が先の) 時間領域にスケジュールしたときのタイムスロットの最大長で表し, 解の精度はどれだけ最適解に近いかを表すため「最適解／提案手法の解」とした. ここで最適解とは, ジョブスケジュールされたタイムスロットの最大長が最も小さい解である.

また, グラフの分割ではメッセージサイクル数[2]も評価の対象とした. グラフ分割におけるメッセージサイクル数は 2 分割したグラフの問題についての和をとった.

4.2 結果

後退辺の削除を行った場合の問題のサイズと解の精度, 削減した後退辺の数, 削減した木の幅との関係を図 3 に, 問題の分割を行なった場合を図 4 に示す. また, 削減した後退辺, 制約辺の数と解の精度との関係を図 5 に示す.

図 3, 図 4 より, 後退辺の削除, 問題の分割ともにジョブ数が増加した場合でも解の精度の低下は比較的小さいと考えられる. 図 5 より、後退辺の削除, 問題の分割共に, 削減した後退辺, 制約辺が増えるに従って解の精度が減少している. これは, 削除される後退辺, 制約辺の数が増加したため, 探索範囲がより限定され, 緩和された問題では解の精度が減少したためであると考えられる.

問題の分割を用いた場合のメッセージサイクル数の変化を図 6 に示す. 問題の分割では図 6 より, 分割前と分割後でのメッセージサイクル数に大きな変化が見られないことがわかる. メッセージサイクル数の比較より, 提案手法と従来手法では解探索の時間は, ほぼ同じであることがわかる.

UTIL メッセージのサイズを後退辺の削除を行なった場合を表 1 に, 問題の分割を行なった場合を表 2 に示す. 表 1 より, 後退辺の削除では, 木の幅が既定値になるまで後退辺の削除を行ない, 解探索の限定を行なった為,UTIL メッセージのサイズが削減されていることが確認できる. 表 2 より, 問題の分割では, 問題を分割したことにより解

[2]システム全体はメッセージサイクルで同期している. 各メッセージサイクルでは, 各ノードは他ノードからのメッセージ受信, ノード内部の処理, および他ノードへのメッセージの送信を行う.

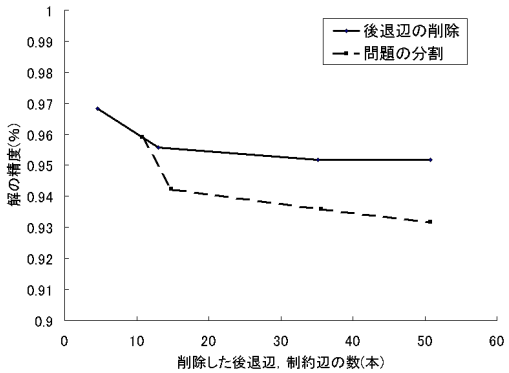


図 5: 解の精度と削除した後退辺, 制約辺の関係

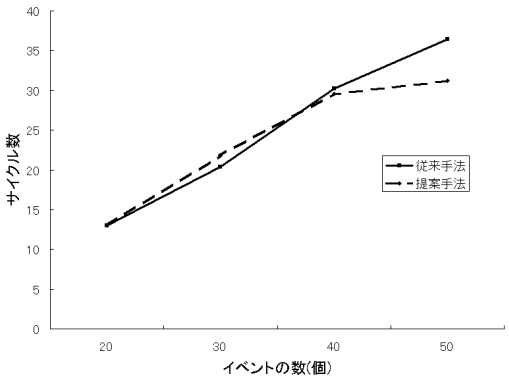


図 6: 従来手法と問題の分解との UITL メッセージのメッセージサイクル数の関係

の組合せの範囲が削除され,UTIL メッセージのサイズが削減されいることが確認できる. 探索範囲を限定し, メッセージのサイズの削減を行なったが, 解の精度は後退辺の削除では 0.95〜1 程度, 問題の分割では 0.93〜1 程度の解の精度がある.

ジョブ数 (個)	20	30	40	50
改善前 (MB)	4.4×10^2	4.6×10^4	3.0×10^{12}	4.7×10^{17}
改善後 (MB)	1.2×10^1	1.2×10^1	1.2×10^1	1.2×10^1

表 1. 後退辺の削除による UTIL メッセージのサイズの削減

ジョブ数 (個)	20	30	40	50
改善前 (MB)	4.4×10^0	1.8×10^3	1.7×10^9	1.0×10^{16}
改善後 (MB)	0.3×10^{-3}	3.7×10^{-2}	8.1×10^0	1.4×10^1

表 2. 問題の分割による UTIL メッセージのサイズの削減

5 おわりに

本研究では分散制約最適化問題をグリッド計算環境のコアロケーションのための分散スケジューリングに適用した. 解探索は動的計画法に基づくアルゴリズム DPOP を用い, メッセージサイズの増加を抑制するために 2 つの問題の緩和手法を示した. 後退辺の削除による方法では,Pseudo tree の後退辺を削除することにより木の幅をさげ, 解探索の範囲を限定した. また, 問題の分割による方法では, 問題を半分に分割することにより問題のサイズを小さくし, 解の組合せの範囲を限定した. これらの手法により, 問題を緩和することで UTIL メッセージのサイズを減少させた. シミュレーションを用いた初期の評価においては, 提案手法により, メッセージサイクル数, メッセージのサイズ, が削減され, 一方で, 解の精度の低下は比較的小さいことが示された.

今後の課題として, 後退辺の削除, 問題の分割におけるヒューリスティクスの検討が挙げられる. 後退辺の削除では変数の割当て, 問題の分割では分割場所を変えることによって探索範囲を改善し, 効率のよい探索が期待できる. また, 実際的な問題への適用, 他のスケジューリング方法との比較も今後の課題である.

参考文献

[1] 竹房あつ子, 中田秀基, 工藤知宏, 田中良夫, 関口智嗣, ”計算資源とネットワーク資源を同時確保する予約ベースグリッドスケジューリング ” SACSIS 2006, pp. 93-100, 2006.

[2] A. Pecu and B. Faltings, ”A Scalable Method for Multiagent Constraint Optimization”, Proc. 9th Int. Joint Conf. on Artifical Intelligence, pp. 266-271, 2005.

[3] R. T. Maheswaran, M. Tambe, E. Bowring, J. P. Pearce and P. Varakantham, ”Taking DCOP to the RealWorld:Efficient Complete Solutions for Distributed Multi-Event Scheduling”, Proc. 3rd Int. Joint Conf. on Autonomous Agents and Multiagent Syatems, pp.310-317, 2004.

[4] Z. Collin and R. Dechter, ”Self-Stabilizing depth-first search”, Information Processing Letters49(6), pp297-301, 1994.

[5] A.Pecu and B.Faltings, ”Superstabilizing,Fault-containing Multiagent Combinatorial Optimization”, In Proceedings of the National Conference on Artificial Intelligence, AAAI-05, Pittsburgh, USA, July 2005.

[6] 梅田典宏, 中田秀基, 松岡聡, ”大規模環境向け情報共有手法を用いた分散ジョブスケジューリング ”, 情報処理学会 HPC 研究会, pp.223-228, 2006.

[7] 秋岡明香, 竹房あつ子, 中田秀基, 松岡聡, 三浦 謙一, ”グリッド環境におけるスーパースケジューラ連携手法の検討 ”, 情報処理学会 HPC 研究会, pp.55-60, 2005.

[8] 竹房あつ子, 松岡聡, ”グリッド計算環境でのデッドラインを考慮したスケジューリング手法の性能 ”, 信学論 D-I, Vol.J86-D1, No.9, pp661-670, 2003.