RA-006

通信を考慮したタスクスケジューリング問題の効率的な並列探索解法の提案
Development and Evaluation of Parallel Search Method to Solve Task Scheduling Problems on Account of Communication Overhead

栗田 浩一† 宇都宮 雅彦† 塩田 隆二† 甲斐 宗徳†
Koichi Kurita Masahiko Utsunomiya Ryuji Shioda Munenori Kai

1. はじめに

マルチプロセッサシステムのためのタスクスケジューリングは，実行プログラム中の並列実行可能な部分処理を複数のプロセッサに割り当てる順序とタイミングの最適化により，プログラムの実行時間を最小化する技術である．

このタスクスケジューリング問題は強 NP 困難な計算複雑度を持つ組合せ最適化問題であるため，最適解を求めるには，本質的に分枝限定法を用いた全探索が必要である[1]．強 NP 困難な複雑度を持つ組合せ最適化問題では，問題サイズの増大につれて探索範囲が指数関数的に増大してしまう．そのため，サイズの大きい問題ではその最適解を実用的な時間で得ることが非常に困難となる．

プロセッサ間の通信を考慮しないタスクスケジューリング問題を解決する実用的な全探索アルゴリズムとして DF/IHS（Depth First / Implicit Heuristic Search）[2]，さらにそれをマルチプロセッサで並列探索する PDF/IHS（Parallelized DF/IHS）が提案されている[3]．これらのアルゴリズムは全探索解法であるため，理論的には最適解を求めることができる．しかし，これらは通信を考慮したアルゴリズムではないため，そのスケジューリング結果に従って実際の並列実行を行うと，通信時間により並列処理の性能を十分に発揮できない場合がある．従って通信の組合せをタスクスケジューリング時に考慮することが必要と考える．ただし，それは組合せ最適化問題の計算複雑度をさらに増加させることになるので，実用的な時間内で最適解または最適解に近い精度の解を得るためには，探索時間を削減する工夫が必要になる．分枝限定法に基づく探索時間の削減には，効率の良い枝切りと探索の並列化が有効である．そこで，本論文では，効率の良い枝切りと探索の並列化のために考案した「通信を考慮したタスク下限値の導出法」，および「並列探索の効率を向上させる探索枝生成操作」を提案する．これらの提案アルゴリズムは，通信を考慮したタスクスケジューリングの探索時間を大幅に削減することが可能となることを示す．

2. タスクスケジューリング

2.1 タスクグラフ

タスクとは，実行プログラムを並列処理するために分割した部分処理である．タスクをノード，先行制約を有向エッジで表した非循環有向グラフをタスクグラフと呼ぶ．図 2．1 に本研究が対象とするタスクグラフの例を示す．ノードの中の数字がタスク番号，ノードの左側の数字がタスクの処理時間を表している．

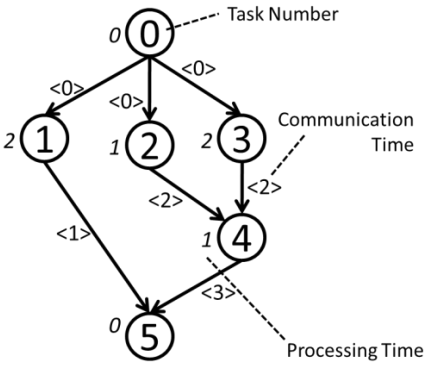

図 2．1 タスクグラフ

本研究では，先行制約の存在する 2 つのタスクを異なるプロセッサに割り当てる際の依存データの送受信に要する通信時間を考慮する．そのため，有向エッジに通信時間が付加されている．<>で囲まれた数値が通信時間を表す．

各タスクは，有限個のプロセッサに割り当てられ，並列に処理される．「一般形状の先行制約で，各タスクの処理時間が異なり，プロセッサ数も任意」という，一般化されたタスク集合のモデルに対して行うタスクスケジューリング問題は，強 NP 困難な計算複雑度を持つ．

2.2 通信のタイミング

先行タスクからの情報を別のプロセッサに割り当てられた後続タスクに送信する場合，それを実現するモデルは複数存在する．図 2．2 のタスクグラフを例として，複数の送受信パターンが考えられることを以下に述べる．

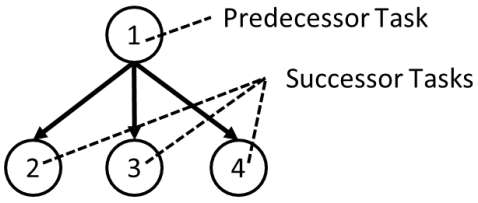

図 2．2 タスクグラフ上の先行後続関係

最も一般化されたモデルは，通信の重複に制限のない Fully Coupled Model である．しかし，情報の転送のための通信ポートが無制限に用意されていることは現実的ではないため，本研究ではより実用的なモデルとして，各プロセッサが情報の送信ポートと受信ポートを 1 つずつ用意している分散メモリ型マルチプロセッサモデルを用いる．Fully Coupled Model と我々が仮定するモデルでの通信の生じ方の一例を図 2．3 に示す．この図の縦軸は処理要素の番号（PEn），横軸は時刻を表している．Fully Coupled Model

† 成蹊大学理工学研究科理工学専攻 Graduate School of Science and Technology, Seikei University

では，一度にすべての後続タスクへ通信を行うことが可能であるため，アイドルプロセッサが後続タスク数以上存在する場合，後続タスクに通信を開始するタイミングは図 2．3 左に見られるようにすべて同時刻になる．一方，実用的なモデルでは，通信順序に関する組合せが生じることにより，図 2．3 右に示す場合を含む複数の組合せを考慮しなければならない．このため，探索空間は以前よりも広大化することとなる．

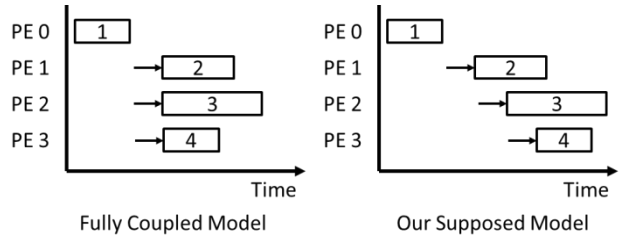

図 2．3 モデルごとに異なる通信オーバヘッドの例

3. スケジューリング手法

3.1 近似解アルゴリズム

強 NP 困難な組合せ最適化問題であるスケジューリング問題において近似解の精度に一定の評価を得ているヒューリスティックアルゴリズムとして，CP（Critical Path）[4] および CP/MISF（Critical Path / Most Immediate Successors First）[2]と呼ばれるリストスケジューリングの手法が考案されている．CP は各タスクからタスクグラフの出口ノードまでの最長パス長がより長いタスクから優先してアイドルプロセッサへ割り当てる．CP/MISF も CP と同じ各タスクから出口ノートまでの最長パス長を基準とするが，その最長パス長が等しいタスク同士において，直接後続タスク数が多いものを優先するというヒューリスティックを取り入れている．

3.2 全探索アルゴリズム

スケジューリング結果に，近似解では満足できないような高い精度が求められる場合，厳密解を求めることが必要である．厳密解を求めるための実用的な最適化アルゴリズムとしては，DF/IHS，PDF/IHS が笠原らにより提案されている[2，3]．DF/IHS は，探索木を構成しながら分枝限定法に基づく深さ優先探索を行う逐次最適化アルゴリズムである．

図 3．1 では，図 2．1 のタスクグラフを実際に 2 プロセッサに割り当てるスケジューリング問題に対して DF/IHS で探索した場合の探索空間である．RT(Ready Task)はその時点でアイドルプロセッサに割り当て可能なタスクの集合を表す．各ノードは RT から CP/MISF に基づく優先度の高いタスクの順序で選んだ割り当てタスクの集合を表し，各エッジは先行制約を表す．Idle Task の割り当ては，プロセッサをアイドル状態にすることを意味する．DF/IHS は図のような探索空間を左端から右端にかけて深さ優先探索を行う．

この探索では，CP/MISF のヒューリスティック効果を取り入れることで，生成される探索空間の左側により良いと考えられる解を集め，良質な初期解を得た状態から探索を開始できる．さらに,より良い解があるとされる枝を優先的に探索することで,早期に精度の良い暫定解を得られ，枝切りが効率的に行われる． PDF/IHS はこれを並列処理用に拡張した並列最適化アルゴリズムとして，その有効性が確認されている[3].

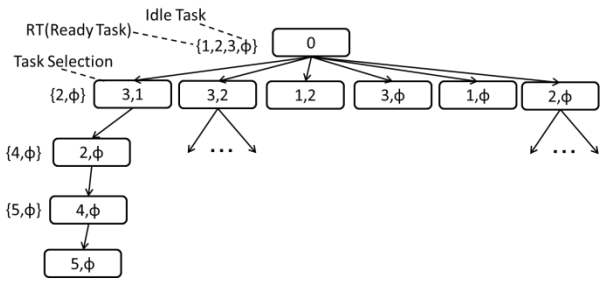

図 3．1 DF/IHS の探索空間

DF/IHS と PDF/IHS のアルゴリズムの実行時間の比較は，各アルゴリズムを計算機上に実装し，その処理時間の比較によって行われている[5]．このように探索のアルゴリズムの評価が探索処理時間で行われているのは，現時点では一般的な問題の最適解を得る多項式時間アルゴリズムが得られていないためである．

3.3 通信を考慮したスケジューリングアルゴリズム

プロセッサ間の通信を考慮したスケジューリングのアルゴリズムとして，ETF（Earliest Task First）[6]，CP/DT/MISF （Critical Path / Data Transfer / Most Immediate Successors First）[7]，などが提案されている．ETF はデータ転送を考慮した上で，最も早く実行できるタスクとプロセッサを組にして優先順位を決定する．CP/DT/MISF は，CP/MISF のアルゴリズムに加え，タスクを局所的なデータ転送コストが最小となるように，リストスケジューリングでプロセッサへ割り当てる．

一方で，通信を考慮した全探索のスケジューリングについては，汎用的に有効な手法が確立されていないのが現状である．DF/IHS は通信を考慮しないスケジューリングについては有用性が示されている．しかし，そのヒューリスティックは，通信を考慮していないアルゴリズムである CP/MISF に基づいている．このヒューリスティックのまま探索空間に通信時間を算入した解候補を生成しても，通信を考慮したヒューリスティックの意味で探索空間の左側に良い解を集められるとは限らない．その結果，枝切りの効果が十分に発揮されない場合が存在する．従って，ヒューリスティックの段階で通信時間を考慮することが重要と考えられる．

4. 通信を考慮した下限値の提案

4.1 枝切りに及ぼす下限値の効果

各タスクの下限値とは，そのタスクの処理開始から出口タスクの処理終了までのタスクグラフ上の最長パス長となる．枝切りは，スケジュール長の短縮が図れない部分木の探索を打ち切る操作である．スケジューリングの途中で，あるタスクの割り当てまで済んだときに，そこまでのスケジュール長とそのタスクの直接後続タスクの下限値の和が，既に得られている最良のスケジュール長より長い場合に枝切りを行う．図 4．1 では，タスク②の枝が枝切りの対象

となる．

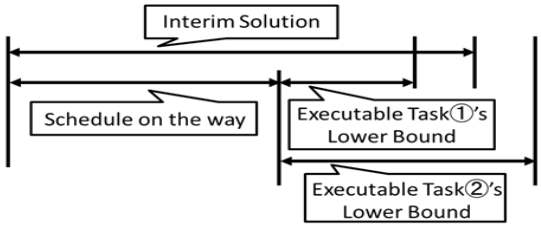

図 4．1 枝切りによる限定操作

DF/IHS では CP/MISF を利用しているので，下限値はタスク処理時間だけを元に算出されており，タスク間の通信時間は考慮されていなかった．通信を考慮したタスクスケジューリング問題を DF/IHS で解くことは可能であるが，必要となる通信時間を考慮していないため，各タスクの下限値が過小評価され精度が悪くなる場合が生じる．精度の悪い下限値を用いると，枝切りがしにくくなり，結果的に探索時間が増大することになる．通信を考慮してより良い精度の下限値を使用することができれば，より多くの枝切りを行い探索時間の削減を期待できる．

さらに，各タスクの下限値は，プロセッサへの割り当ての優先順位（プライオリティレベルと呼ぶ）としてそのまま利用されている．プライオリティレベルの高いタスクほど優先的に割り当てられなければならないので，通信時間を考慮されずに求められた下限値をプライオリティレベルに用いることは，ヒューリスティックの効果を下げることにつながると考えられる．この点でも通信を考慮した下限値をプライオリティレベルに用いることが望ましい．

4.2 通信を考慮した下限値

4.2.1 前提条件

通信を考慮した下限値計算において，後続タスクの全ての通信状況の組合せを考慮すると，探索の前準備である下限値計算自体が探索問題となってしまう．そこで，下限値計算が複雑化することを緩和するために以下の条件を設定した．

① プロセッサの数は無制限で，全てのプロセッサはアイドル状態とする．
② 下限値の計算には直接後続タスクのみを用いる．
③ 後続タスク同士の先行関係を考慮しない．

提案手法の説明にあたり，以下の定義を導入する．

A : 下限値を求めるタスク
t(A) : タスク A の処理時間
P_A : タスク A を処理したプロセッサ
$P_{\sim A}$: タスク A を処理していないプロセッサ
S_A : タスク A の直接後続タスクの集合
T_A : タスク A の直接後続タスク
$com(T_A)$: タスク A からタスク T_A への通信時間
$lb(T_A)$: タスク T_A の下限値
$lb'(T_A) : lb(T_A) - t(T_A)$
$pass(T_A)$: タスク A の処理後，タスク T_A を通る出口タスクの処理が完了するまでに必要な処理時間
pcpl(A) (Partial Critical Pass Length) : S_A から全ての T_A のプロセッサに対する割り当て方ごとの $pass(B)\{B|B \in S_A\}$ の最大値

4.2.2 下限値計算手順

4.2.2.1 P_A で全てのタスクを処理する場合

まず，P_A に全ての直接後続タスクを割り当てる場合を考える．この時，タスク A 終了後の最低限必要な処理時間(remaining distance)は，lb'の降順にタスクを処理した時の pcpl(A)となる．

P_A で全ての直接後続タスクが処理される場合，タスク A の処理後，lb'の降順で i 番目に処理するタスクを i とする．図 4．2 にそれを示す．

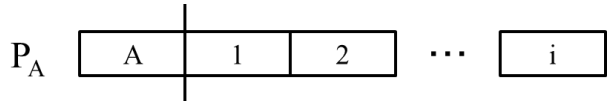

図 4．2 P_A におけるタスクの割り当て順序

また，タスク A の直接後続タスクを P_A に割り当てた場合，k 番目に処理されるタスクを k とし，タスク k は自身の pass が pcpl(A)となる．以下では，図 4．3 に示すように，S_A の中で k 以前に処理するタスクを ${}_{pre}k$，k 以後に処理するタスクを ${}_{post}k$ と呼ぶ．

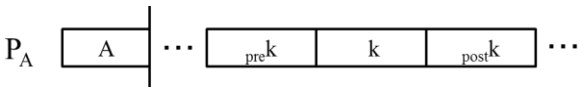

図 4．3 タスク ${}_{pre}k$，k，${}_{post}k$ の位置関係

タスク k の処理後までの部分の処理時間 α は，

$$\alpha = t(1) + \ldots + t(k)$$

となる．従って，pcpl(A)は，

$$pcpl(A) = pass(k) = \alpha + lb'(k)$$

という式で求められる．

① タスク k とタスク ${}_{post}k$ を入れ替えると α に $t({}_{post}k)$が加えられる．従って，pass(k)は増加する．
$$\alpha + \ldots + t({}_{post}k) + lb'(k) > pass(k)$$
② 次にタスク ${}_{pre}k$ とタスク ${}_{post}k$ の処理順を入れ換えると pass(k)は増加する．このとき pcpl(A)は $pass({}_{pre}k)$となる．
$$pass({}_{pre}k) = \alpha + \ldots + t({}_{post}k) + lb'({}_{pre}k)$$
$$lb'({}_{pre}k) > lb'(k)$$
③ また，lb'の降順の順序で，これより前にタスク k を処理すると元のタスク k の順で処理するタスク ${}_{pre}k$ が生じるので，pass(k)は増加する．このときの pcpl(A)は $pass({}_{pre}k)$となる．
$$pass({}_{pre}k) = \alpha + lb'({}_{pre}k)$$
$$lb'({}_{pre}k) > lb'(k)$$

従って，P_A に全ての直接後続タスクを割り当てる場合 lb'の降順に直接後続タスクを処理すると pcpl(A)が最小となる．

4.2.2.2 他のプロセッサも使用する場合

次に $P_{\sim A}$ も使用して処理する場合を考える．P_A で処理しようとしたタスク i を，$P_{\sim A}$ で処理するとその pass(i)は，

$$pass(i) = com(i) + lb(i)$$

となる．以下に，$P_{\sim A}$ も使用して処理する場合の remaining distance を求める．

① S_A の全ての直接後続タスクを P_A に割り当てる．
② P_A で処理しているタスクの中で com(i)+lb(i)が最小となるタスク i を探索する．P_A で処理するタスクがタスク i のみなら，その pass(i)が pcpl(A)となる．

③ P_A で処理する j 番目のタスクを j と仮定し，pass(j)が pass の最大値と仮定する．pass(j)と com(i)+lb(i)を比較する．com(i)+lb(i)が大きいなら pass(j)が pcpl(A)となる．そうでなければ，タスク i を P_A から取り除く．

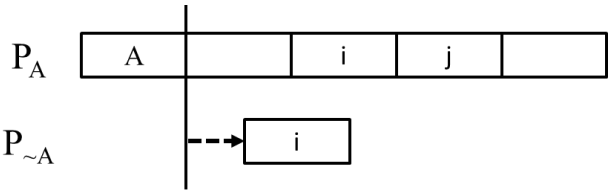

図 4．4 タスク i を処理するプロセッサの選択

④ ②に戻って pcpl(A)を再計算する．

以上①～④の手順で pcpl(A)にタスク A の処理時間を加えた値がタスク A の下限値となる．

4.2.3 実行と評価

4.2.3.1 評価方法と実行環境

通信を考慮しない下限値と通信を考慮した下限値を用いてそれぞれ全探索を行った処理時間を比較し，その処理時間の削減量でスケジューリング効率の向上を評価する．

評価にはランダムに生成したタスクグラフを使用した．生成のパラメータは，タスク数 15，タスク処理時間 1～30，通信時間 1～30，後続タスク数 1～3，先行タスク数 1～3 である．生成されたタスクグラフを 4 プロセッサに割り当てるスケジューリング問題を 50 個解いた．

探索を行ったマシンの仕様は以下の通りである．

CPU : Intel(R) Xeon(R) X5550 @ 2．67GHz x 2
OS : Linux
RAM : 12GB

4.2.3.2 通信を考慮した下限値を用いた探索時間評価

図 4．5 は横軸がタスクグラフ番号，縦軸が探索時間削減率となっている．

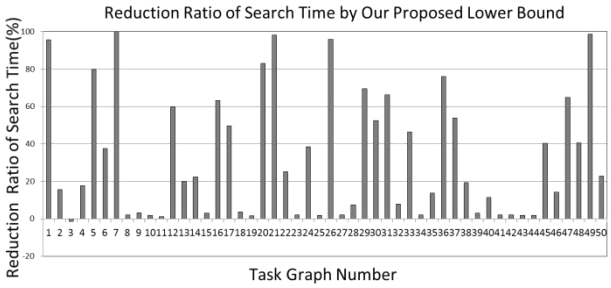

図 4．5 探索時間評価の結果

図 4．5 から 0%の横軸の上側が全探索時間を改善したケースである．全探索時間の削減率はタスクグラフに依存しているが，結果から探索時間が削減されている傾向にあることが分かる．一部では探索時間が長くなったが，これは通信を考慮した下限値をタスクのプライオリティレベルとして使用し，本来なら優先して処理すべきタスクの割り当て順に変化を与えてしまったためと考えられる．それでも一部では 90%以上探索時間が削減されるタスクグラフも存在した．平均的に探索時間が 32．81%削減されているため，提案する下限値がスケジューリング効率の向上に貢献していると判断できる．実時間では，図 4．5 の番号 7 のタスクグラフが，通信を考慮しない下限値を使用した探索ではおよそ 900 秒間の探索時間を費やし，提案アルゴリズムを使用することで，およそ 1 秒間の探索時間に削減することができた．

5. 並列探索アルゴリズム

5.1 PDF/IHS

提案手法は m 台のプロセッサを使用して探索を行う PDF/IHS の改良を試みたものである．まず，PDF/IHS について図 5．1 を用いて説明する．探索に使用するプロセッサのうち 1 つをマスター（PE0）とし，残りの全てのプロセッサ（PE1～PE_{m-1}）をスレーブとして扱う．マスターは，DF/IHS に従って探索空間の左端から右端へ深さ優先探索を行う．スレーブは，探索空間の右端から左端へ深さ優先探索を行う．両側からのアプローチにより，探索木を挟み打ちの形で探索する(図 5．1)．マスターとスレーブの探索の重複を避けるため，探索を行った枝番号を示すセレクションポインタ(以降 SP)を使用する．これにより，マスターとスレーブは互いに探索木上の位置を確認できる．スレーブは，マスターから割り当てを受けると，常に右端から探索を開始する．マスターは，この割り当てと SP 送信以外では DF/IHS の深さ優先探索のみを行うため，探索の並列化にともなうオーバヘッドを極力低減する方法となっており，並列化による減速異常は起こさない．

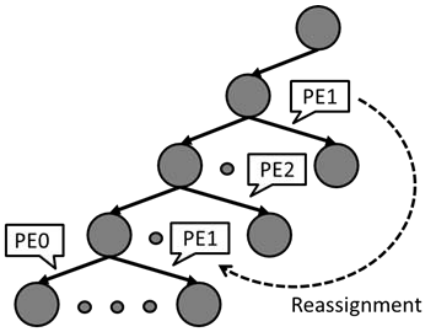

図 5．1 PDF/IHS における挟み打ち探索

5.1.1 セレクションポインタの使用方法

SP はノード毎に探索空間の左側から数えて通った枝番号を示す．マスターは，階層を一つ下がる直前に，その階層での SP をスレーブへ送信する．スレーブは，それを元にマスターの SP テーブルを作成する．スレーブも，自身の階層が下がる毎に，SP テーブルを作成する．SP テーブルは，探索木の根でインデックス番号を 1 とし，インデックス番号は階層の深さと同値な構造体配列になっている．マスターの SP データを受信する毎にスレーブは自身の SP テーブルを更新し，自身とマスターの位置を確認することで探索枝の選択の重複を避ける．

5.1.2 スレーブの割り当て方法

探索開始直後，図 5．1 に示したように，マスターは探索木の左端から探索し，自分が通った枝情報となる SP と最初の暫定解をスレーブに送信する．スレーブは，この 2 つの情報を受け取ると，根より SP を使用してマスターを追尾する．マスターを追尾する間，スレーブは右端から独立に探索できる枝を自身で発見し，発見次第マスターへ割

り当てポインタとして，
・階層
・探索の選択枝(処理時間から受信パターンの内 1 つを示す番号)
の 2 つの整数情報を送信する．探索における選択枝は，SP の階層以外のデータのインデックス番号に相当する．マスターはその情報から各スレーブの探索が重複しないよう管理し，可能ならば割り当て要求を行ったスレーブに探索枝を割り当てる．スレーブは，割り当て許可を受けると，マスターの追尾を止め，割り当てられた探索枝以降の探索空間に対して右側から独立に探索を開始する．割り当てが発生しなかった場合は，再度マスターを追尾し，次に独立に探索できる位置を探す．

5.2 提案アルゴリズム

5.2.1 タスク選択時の枝生成操作

PDF/IHS において，スレーブは探索木の右端から探索を行うため，探索空間の左から順にヒューリスティック的に良いとされる解を集めた場合，スレーブはヒューリスティックの効果を十分に受けられない．図 5．2 左に示す枝順序がその一例である．そこで，枝生成の順序に関する新たなヒューリスティックとして，探索の並列化効率を向上させる探索枝生成操作を考案した．

本手法では，ヒューリスティック的により良いと考えられる探索枝から順に，タスク選択時に生成される枝を左右に振り分けながら生成する．この手法により生成される探索枝の順序が図 5．2 右に相当する．

探索枝を左右に振り分ける枝生成操作により，スレーブが探索空間の右端から探索を行う場合にもヒューリスティックの効果を享受でき，ヒューリスティック的により良いとされる解を優先的に探索する並列探索を実現できる．

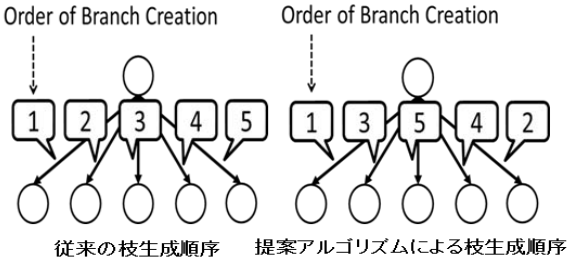

図 5．2 枝生成操作による枝生成順序の変化

5.3 並列探索アルゴリズムの評価

5.3.1 評価方法と実行環境

PDF/IHS をベースとする全探索で提案した枝生成を行った場合と行わなかった場合との処理時間を比較し，その処理時間の削減量でスケジューリング効率の向上を評価する．

評価に用いるタスクグラフは以下の条件でランダムに生成したものを使用した．生成のパラメータは，タスク数 19，先行タスク数 1～3，後続タスク数 1～3，タスク処理時間 1～30，通信時間 1～30 とした．これにより生成されたタスクグラフを 3 プロセッサに割り当てるスケジューリング問題を 50 個解いた．

使用したマシンの仕様は 4．2．3．1 と同様である．並列実行には，8 プロセッサを使用した．

5.3.2 台数効果についての実行結果と評価

図 5．3 は横軸がタスクグラフ番号，縦軸が探索時間削減率となっている．

グラフから 0%の横軸の上側が全探索時間を改善したケースである．

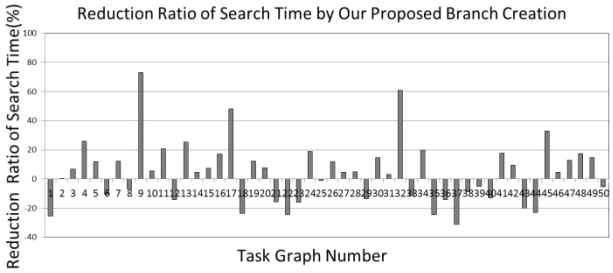

図 5．3　探索時間評価の結果

評価結果から最大で 73%の改善率が得られ，平均で 4．42%の改善が得られた．枝の生成順序の操作により，CP/MISF のヒューリスティックがより活かされた探索空間が構築されたことで枝切りが効率的に行われたためと考えられる．一部で探索時間の改善が見られないグラフも存在したが，提案アルゴリズムはスレーブを右端に割り当て，マスターと両端から挟み打ちの形で探索する場合に有効である．しかし，マスターのみが探索する部分問題では従来の枝生成順序で探索したほうが有効である．提案アルゴリズムでは，マスターのみの部分問題でも探索枝を左右に振り分けた枝生成を行ってしまうため探索時間が伸びてしまったと考えられる．

6. 2 つの提案手法の併用による相乗効果

6.1 評価方法と実行環境

PDF/IHS に今回提案した 2 つのヒューリスティックアルゴリズムを加えた場合と，それらを加えない場合について，限定した実行時間内の探索における暫定解の精度を比較し，実行時間内の探索における暫定解の精度からスケジューリング効率の向上を評価する．今回の性能評価では各タスクのプライオリティレベルは通信を考慮しないものを使用し，通信を考慮した下限値は，枝切りにのみ用いた．これは 4．2．3．2 の評価結果から，通信を考慮した下限値を各タスクのプライオリティレベルにも使用した場合，必ずしもスケジューリングが改善されるとは限らなかったからである．

評価にはランダムに生成したタスクグラフを使用した．生成パラメータは，タスク数 50，タスク処理時間 1～30，通信時間 1～40，後続タスク数 1～3，先行タスク数 1～3 とした．これにより生成されたタスクグラフを 4 プロセッサに割り当てるスケジューリング問題を 5 個解き，スケジューリング効率の向上を評価する．このタスクグラフの規模になると並列探索を持ってしても全探索までには膨大な時間が必要となる．そこで，900 秒間の指定実行時間内スケジューリングにより，暫定解の精度がどの程度向上するかを評価した．

使用したマシンの仕様は 4．2．3．1 と同様である．

6.2 提案手法の相乗効果に関する実験結果と評価

図 6．1 に実験結果を示す．縦軸は，今回提案した 2 つのアルゴリズムを加えた場合の暫定解の精度が，アルゴリズムを加えていない場合と比較して向上した割合を示す．横軸はタスクグラフの番号である．

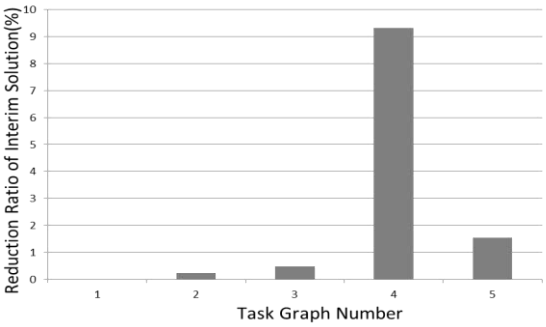

図 6．1 指定実行時間内の暫定解精度の比較

最も提案手法の効果が表れた番号 4 のタスクグラフでは，暫定解が 9．3% 向上した．最も提案手法の効果が表れなかった番号 1 のタスクグラフでは，どちらの並列探索においても 900 秒間の探索において同値の暫定解までしか求解できなかった．

この結果から，設定した探索時間の上限が，評価に用いたタスクグラフの規模に対して短すぎたことが考えられる．探索空間の規模に対して，与えられた探索時間が十分でない場合，アルゴリズムの効果が顕著に表れる以前に探索が終了し，提案手法を適用しなかった場合との暫定解の差が生じにくい状態になることが考えられる．しかし，そのような大規模の探索空間に対する非常に短い時間での探索においても，最大で約 9．2%という解の精度向上が見られた．これは，2 つの提案手法を組み合わせて使用したことがスケジューラの探索効率化に影響し，従来の並列最適化アルゴリズムである PDF/IHS よりも効率の良い探索が行われたためと考えられる．

7. おわりに

本論文では，厳密解を求めようとするタスクスケジューリング問題に対し，下限値と並列探索手法という 2 つの側面から計算時間の削減を試みた．

今回提案した通信を考慮した下限値を探索中の枝切りに使用することで，スケジューリング効率が向上し，通信を考慮しない下限値を使用するよりも短時間で求解できることが確かめられた．

並列探索手法では，PDF/IHS と枝生成操作を組み合わせることで通信を考慮した探索空間を構築し，並列処理における加速異常を有効に引き出すことで，探索時間の削減を実現できた．

一方で，各タスクのプライオリティレベルとして提案した下限値を利用することについてはすべてのケースで必ずしも良い結果を出すことはできていない．従って，今後は通信を考慮したタスクのプライオリティの設定に関する調査・研究を進めていくことで，通信を考慮したタスクスケジューリングにおいて更なる効率化が期待できる．

謝辞

本研究の一部は，文部科学省戦略的研究基盤形成支援事業の補助を受けて行ったことをここに記し，謝意を表します．

参考文献

[1] M. R. Garey, D. S. Johnson, “Computers and Intractability: A Guide to the Theory of NP-Completeness”, W. H. Freeman; First Edition (1979).

[2] H. Kasahara, S. Narita, “Practical Multiprocessor Scheduling Algorithms for Efficient Parallel Processing”, IEEE Trans. on Computers,Vol. C-33, No. 11, pp. 1023–1029, Nov. (1984).

[3] H. Kasahara, A. Itoh, H. Tanaka, K. Itoh, “A Parallel Optimization Algorithm for Minimum Execution-Time Multiprocessor Scheduling Problem”, IEICE Vol. J74-D-I, No. 11, pp. 755–764, Nov. (1991).

[4] E. G. Coffman, “Computer, Job-shop Scheduling Theory”, John Wiley & Sons, (1976).

[5] C. V. Ramamoorthy, K. M. Chandy, Jr. Mario, J. Gonzalez, “Optimal Scheduling Strategies in a Multiprocessor System”, IEEE Trans. on Computers, Vol. C-21, No. 2, pp. 137–146, Feb. (1972).

[6] Jing-Jang Hwang, Yuan-Chien Chow, Frank D. Anger, ChungYee Lee, “Scheduling Precedence Graphs in Systems with Interprocessor Communication Times”, SIAM J. Computers, Vol. 18, No. 2, pp. 244–257, Apr. (1989).

[7] H. Kasahara, H. Honda, S. Narita, “Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR(Optimally Scheduled Advanced Multiprocessor)”, Proc. of IEEE ACM Supercomputing ’90, Nov. (1990).