成蹊大学理工学研究報告
J. Fac. Sci. Tech., Seikei Univ.
Vol.47 No.1 (2010) pp.39-47

分枝限定法を用いたタスクスケジューリングのための通信を考慮した下限値による限定操作の提案

塩田　隆二*1，甲斐　宗徳*2

Evaluation of A New Bound Operation Taking Account of Communication Overhead for Task Scheduling based on Branch and Bound Method

Ryuji SHIODA*1, Munenori KAI*2

ABSTRACT : The task scheduling technique is the very important key for the efficient parallel processing. The task scheduling problems belong to the class of NP complete or strong NP complete combinatorial optimization problem. So in many practical cases, heuristic algorithms are used to solve them. Taking account of communication overhead into task graphs makes these problems more difficult to solve by any search algorithms because the search spaces become much larger. In this paper, we propose, taking account of communication time among tasks, an extended version of original priority levels used in Critical Path(CP) Method. Priority level of each task is the length of the longest path from the task to the last task in the task graph. It is the lower bound of the parallel processing time after the task. The search is performed by depth first search manner using branch and bound method. By using our newly proposed priority level as both heuristic and bound operation in the search, it is shown that the scheduling time become shorter than the branch and bound search with the original priority level.

Keywords : task scheduling, branch and bound search, heuristic algorithm, combinatorial optimization problem

（Received March 25, 2010）

１．はじめに

近年，マルチコアプロセッサやマルチプロセッサの技術が向上し，並列処理を行う環境が広まっている。

並列処理をより高速に実行するためには，並列処理マシンの持つ計算資源の性能を十分に引き出すことが重要となる。効率の良い並列処理の為には，プログラムを解析して並列実行可能なタスク集合を見つけ，そのタスク集合を並列処理時間が短くなるようにどのようにプロセッサへ割り当てるかを決めるスケジューリングが必要になる。

しかし，このスケジューリング問題は強 NP 困難なクラスに属す組み合わせ最適化問題であり，すべてのケースで最適解を求める方法を得ることは，事実上不可能であることが知られている。そのため，実用的に最適または近似的な解を求めるスケジューリング手法がいくつか提案されている。これらは人間の経験に基づくヒューリスティックアルゴリズムであり，得られた解が最適解であるという保証はない。また，全探索アルゴリズムでは理論的には最適解が求まるが，実用的な時間で最適解を得ることは不可能に近い。

これらのスケジューリングアルゴリズムにはスケジューリング効率を左右する下限値という値が存在する。本研究では，従来の下限値計算で求まる下限値より精度の高い下限値を求める計算法を提案することでスケジューリング効率の向上を目標とした。

*1：工学研究科情報処理専攻修士学生
*2：工学研究科情報処理専攻教授 (kai@st.seikei.ac.jp)

２．スケジューリングモデル

2.1　タスクグラフ

並列実行可能な１つの処理単位をタスクと呼び，タスク全体をタスク集合と呼ぶ。タスク集合は，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で表現できる。このグラフをタスクグラフと呼ぶ。図 2.1 は本研究の対象とするタスクグラフの例である。ノード

の中の数字がタスク番号，ノードの右側の数字がタスクの処理時間を表している。また，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情報共有に要する通信時間を考慮している。そのため，有向エッジに通信時間が付加されている。

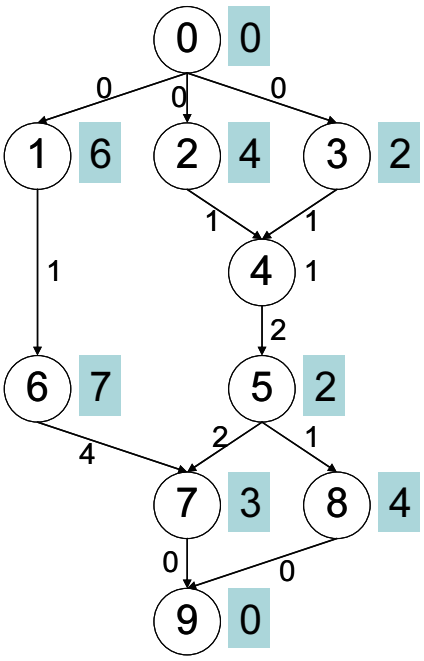

図 2.1　タスクグラフ

タスクスケジューリング問題とは，タスクグラフの各タスクを与えられた任意の数のプロセッサに並列処理時間を最小にすることを目的にして割り当てを決定することである。

2. 2　通信のタイミング

プロセッサ間の通信オーバヘッドを考慮する際，相互結合網に完全結合を想定すると，複数の宛先に同時にデータを通信することが可能となる。しかし，将来的なプロセッサ（コア）数のスケーラビリティを考慮すると，完全結合は現実的ではないので，本研究では 1 対 n の通信が 1 対 1 の通信を n 回逐次的に行う相互結合網を対象とする。そこで，以下のような分散メモリ型マルチプロセッサをモデル化した。各プロセッサはローカルメモリ，入出力通信バッファが同じ構成である。

- メモリ及び通信バッファの容量は十分あるとする。
- データは非同期に送受信される。送信側プロセッサは送信データを通信バッファに書き込み，受信側プロセッサは受信データを通信バッファから読み込む。
- 送受信中プロセッサは占有されず，データが送り元通信バッファから送り先通信バッファへ転送される。
- モデルとする相互結合網では，プロセッサ間 1 対 1 の通信は一定コストでなされる。1 対 n 及び n 対 1 の通信を同時刻に行うことは許されず，その場合，n 回の通信コストの和だけ，コストがかかる(図 2.2)。
- 一組の通信による通信経路の占拠は通信スロットによって表される。通信スロットは分割できないものとする。

このような分散メモリ型マルチプロセッサモデルを対象としているため，通信の有無だけではなくタイミングも考慮してスケジューリングを行う。そのため，タスクのプロセッサに対する割り当ての組み合わせだけでなく，組み合わせに対する順列もプロセッサへの割り当てパターンとして考慮に入れなくてはならない。このように通信を考慮すると探索空間が通信を考慮しない場合よりも広大化することになる。

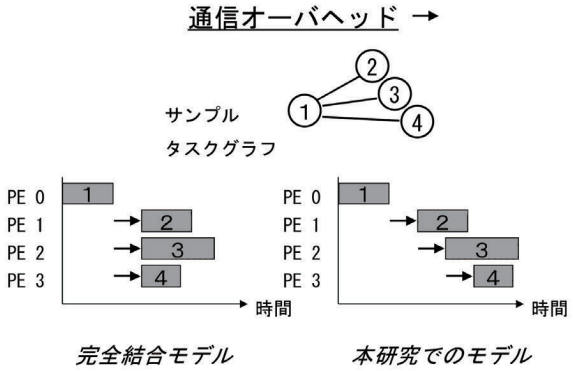

図 2.2　マルチプロセッサモデルと通信オーバヘッド

3　スケジューリング手法

3. 1　リストスケジューリング

スケジューリング問題を解くヒューリスティックアルゴリズムにリストスケジューリングがある。リストスケジューリングとは，実行可能となったタスクを空いているプロセッサに割り当てていく方法である。

3. 2 CP/MISF 法（Critical Path/Most Immediate Successors First）

CP/MISF 法はヒューリスティックアルゴリズムであり，リストスケジューリングにおいてタスクの優先順位を考慮するものである。実行可能となったタスク数がアイドルプロセッサ数より多いとき，割り当てタスクの選択の優先度を決定するために次のような経験則を用いる。

① 各タスクから出口ノードまでの最長パス長を計算。

② 最長パス長の降順にプライオリティリストを作成。
③ 最長パス長が等しければ，その中で直接後続タスクの多い順に，プライオリティリストを作成。
④ 先行タスクの実行が終わり，実行可能なタスクの中でプライオリティリストの優先度が高いものから空いているプロセッサに割り当てる。

3. 3 DF/IHS 法(Depth First/Implicit Heuristic Search)法[1]

全探索手法として有効な解法に DF/IHS 法がある。ここで述べる DF/IHS 法はプロセッサを意図的にアイドル状態にすることも考慮して，最適解あるいは精度の保証された近似解を得ることができる。DF/IHS 法は CP/MISF 法におけるヒューリスティック効果と，組み合わせ最適化問題に対して最も有効な分枝限定法と呼ばれる探索手法を取り入れた探索アルゴリズムである。

図 3.1 は DF/IHS 法での探索空間の例である。各ノードは実行可能タスクのうち，アイドルプロセッサに割当てるタスク番号の組合せを表すものであり，φはプロセッサをアイドルにすることを意味している。同一の深さの探索ノードは，実行可能タスクの中でプライオリティレベルの高いタスクの組合せの順に左側から並んでいる。分枝限定法は，この様な探索空間の中を深さ優先探索していき，その時点で求まっている暫定解よりもよい解が得られる毎に解を更新する。一方で，よい解が得られそうにない部分木は，そのタスク処理開始から最終タスク処理終了まで最低限必要な処理時間であるタスクの下限値を使用し，早い時期に枝切りという限定操作を行って無駄な探索量を削減し，最適解を見つける方法である。

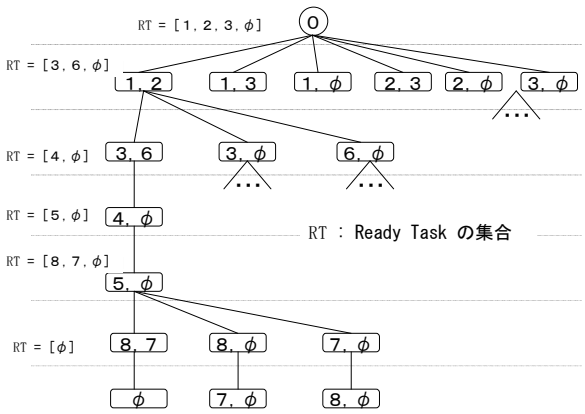

図 3. 1 DF/IHS 法による探索木

DF/IHS 法では, CP/MISF 法のプライオリティリストを使用することで，図 3.1 の最左端の解が CP/MISF 法の解と等しく，精度の高い解候補となる。また，探索木のヒューリスティック的に良い解が集まると推測できる左から右へ，枝切りを行いながら深さ優先探索を行う。

また，DF/IHS 法は強 NP 困難なクラスに属する問題を全探索で解決しようとするため，タスクグラフが数十個以上のタスクを持つと膨大な探索時間を必要とする。

3. 4 枝切り

枝切りとは，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。現在のスケジュール途中結果と，これから最低かかる処理時間の合計（実行可能タスク以降の残存処理時間下限値，以下，これを単に下限値と呼ぶ）が今までに得られている最良のスケジュール長より大きくなると枝切りを行う。図 3.2 では，②の場合に枝切りが発生する。

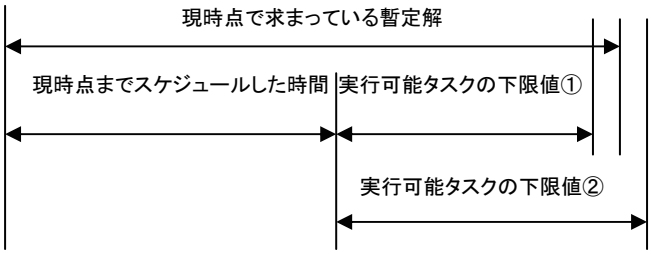

図 3.2 枝切りによる限定操作

枝切りの処理は以下の手順で行われる。

Ⅰ 処理を行ったタスクの中で処理終了時刻が最も遅い時刻となるタスクの終了時間を求める。
Ⅱ 処理を行っていないタスクの下限値の中で最大の下限値を求める。
Ⅲ ⅠとⅡの合計値と暫定解を比較する。
Ⅳ 暫定解の値を超えたら，枝切りを行う。

以上の操作を行うことで，無駄な探索量を削減する。

4 探索空間の削減法

スケジューリングにおける広大な探索空間は問題である。そのため，少しでもスケジューリングする必要のない探索空間を分析して削減を行うことが大切である。そのため，すでに開発された探索空間の削減法である３Ｓ（Shrinking Search Space）法の中で，最適解を逃すことのない２つの手法を取り入れている。

4. 1 必要なアイドルタスクの抽出法[2]

アイドルタスクとは，プロセッサを指定した時間空いたままにするための便宜上のタスクであり，次のタスク

の割り当てを故意に遅らせることにより，スケジュール長の改善を行いたい時に用いられる。アイドルタスクはいつも必要だとは限らないため，解析可能である不必要なアイドルタスクの割り当てを行わない方法である。

4. 2　タスク割り当て制限法[2]

通常，スケジューリングではタスクを各プロセッサに割り当てる組み合わせを，順列を用いてすべて生成してスケジューリングを行っている。しかし，タスク割り当て制限法では，すべての割り当てについて考慮しなくても最適解が得られ，スケジューリングにおける探索範囲を小さくすることが出来る。

5　スケジューリング問題の並列探索解法

スケジューリング問題は，全探索を行うと強 NP 困難なクラスに属す組み合わせ最適化問題であることが知られている。そこで，前述した探索アルゴリズムを並列化する。並列化によって探索空間は複数のプロセッサに分散され，早期に良い解が発見されることで枝切りによる限定操作が増加し，無駄な探索の削減が可能になる。そのため，スケジューリング効率が向上する。しかし，探索に必要な情報の共有に通信時間がかかる可能性があるため，通信のタイミングを考慮し，通信量を抑える必要がある。今回行ったスケジューリングプログラムの並列化の方法を以下に記す。

① プロセッサの中の一つを master，その他のプロセッサを slave と呼ぶ異なる探索方法を行う 2 種類に分類する。

- master の動き

1. 逐次スケジューリングと同じ方法で探索。
2. slave から探索空間の要求があると，要求を受け取った時点で自らがスケジューリングしたプロセッサ，タスク，通信の組み合わせ以降の組み合わせのスケジューリングを行う探索空間を割り当てる(図 5. 1)。

- slave の動き

1. 自らが探索を行う探索範囲を master に要求。
2. 受け取った探索範囲の探索を行い，終了合図がくるまで探索範囲を要求し続ける。

② master と slave は処理開始時刻，タスク，プロセッサ，通信の組み合わせを 1 つ選択するたびに他のプロセッサからの更新された暫定解の通信の有無を確認する。これによって良い暫定解を早期に見つけて枝切りを行い，無駄な探索を削減する。

③ 各プロセッサはタスクグラフを全探索する際の探索木の情報を保有していて，master と slave 間で通信される探索範囲の情報はその探索木の深さと分岐数のみとなる。slave は受け取った情報の深さの分岐より先の部分木を探索することになる。これによって並列探索におけるオーバヘッドとなる通信量を抑制している。

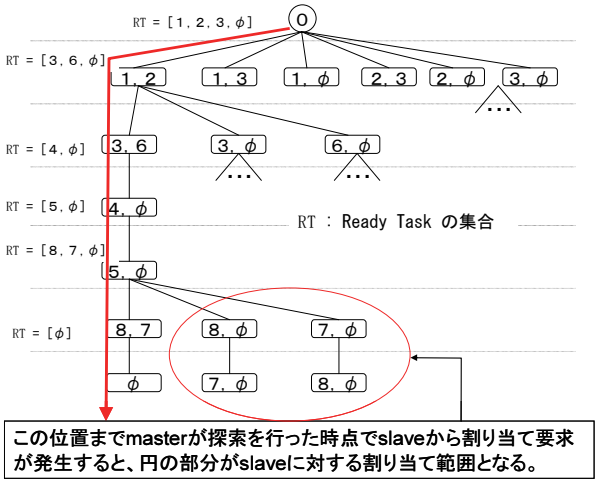

図 5. 1 master の slave に対する探索範囲の割り当て

6　通信を考慮した下限値

タスクスケジューリングでの枝切り操作とプライオリティレベル設定には，各タスクから出口タスクまでの最短距離を使用する。これを下限値と呼ぶ。従来使用する下限値は各タスクの処理時間を元に算出していて，タスク間の通信による遅延は考慮されていない。各タスクの下限値を求める際にタスク間の通信を考慮すると，下限値計算が処理プロセッサ，タスク，通信の組み合わせを考える組み合わせ最適化問題となり，下限値の算出に多くの時間を費やしてしまう。

本研究では，通信を考慮したタスクの下限値を求めて枝切りやプライオリティレベルに使用し，スケジューリング性能を向上することを目標としている。ここでは下限値計算に条件を加えることで計算時間を抑えた下限値の算出方法を説明する。

6. 1　下限値計算の条件

下限値計算では，下限値を求めるタスクが終了してから出口タスクまでの最短経路を求める必要がある。しかし，後続タスクのすべての通信状況を考慮することは困難である。そのため，下限値の計算が複雑化しないような以下の条件を設定した。

① 直接後続タスクの下限値を元に計算する。
② すべての処理装置はアイドル状態とする。
③ 処理装置の数を制限しない。
④ 後続タスク同士の先行関係を考慮しない。
これにより真の最短経路とはいかないが，従来に比べて精度の高い下限値が単純な計算で算出できる。

6. 2 下限値計算

下限値とは，下限値を求めたいタスクの処理後から出口タスク処理までの最低限必要な処理時間を求め，その値と下限値を求めたいタスクの処理時間を合算した値である。

下限値計算で使用する値を以下に記す。

A ： 下限値を求めるタスク
NCPE (Non Communication Processor Element)
： タスク A を処理したプロセッサ
CPE (Communication Processor Element)
： タスク A を処理していないプロセッサ
t ： 任意の直接後続タスク
time(t) ： タスク t の処理時間
com(t) ： タスク A からタスク t への通信時間
lb(t) ： タスク t の下限値
lb’(t) ： lb(t) - time(t)
pass(t) ： タスク A 処理後タスク t を通る出口タスクまでに必要な処理時間
絶対パス長 ： タスク A の直接後続タスクのプロセッサに対する割り当て方ごとの直接後続タスクの pass の最大値

NCPE と CPE の違いはタスク A の処理にある。タスク A を処理したプロセッサはタスク A の直接後続タスクを処理しても通信は必要ない。なぜなら全ての必要なデータはローカルに存在するからである。そのため，NCPE と呼んでいる。タスク A を処理していないプロセッサではタスク A の直接後続タスクを処理する場合，タスク A を処理したプロセッサから必要なデータを受け取らなければならず，プロセッサ間に通信時間が発生する。そのため，CPE と呼んでいる。

6. 2. 1 NCPE で全てのタスクを処理する場合

まず，NCPE にすべ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は，lb’の降順にタスクを処理した時の絶対パス長となる。ここではそれを証明する。

NCPE で全てのタスクが処理される場合，タスク t は lb’が降順の状態でタスク A の処理後 t 番目に処理するタスクとする。

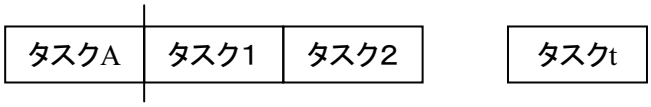

図 6. 1 NCPE におけるタスクの処理位置

また，タスク A の直接後続タスクを NCPE に割り当てた時，自身の pass が絶対パス長となるタスクを x ，x 以前に処理するタスクを w ，x 以後に処理するタスクを y と呼ぶ。

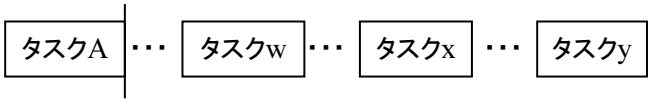

図 6.2 タスク w，x，y の位置関係

タスク A 処理後の絶対パス長は
pass(x)= time(1)+…+time(x)+lb’(x)
という式で求められる。このとき，タスク x を後方で処理すると time(1)～time(x)が増加することになるので
①タスク x をその位置より後方で処理すると絶対パス長は増加する。
②また，以下の２つの式より w と y の処理順を入れ換えると絶対パス長は増加する。
pass(w)=time(1)+…+time(x)+…+time(y)+lb’(w)
lb’(w) > lb’(x)
③また，タスク x の処理を前方で処理すると元のタスク x の順で処理するタスク w が生じるので，タスク x をその位置より前方で処理すると絶対パス長は増加する。
pass(w)=time(1)+…+time(x)+lb’(w)
lb’(x) > lb’(x)
①，②，③より NCPE において lb’の降順にタスクを処理すると絶対パス長が最小となる。

6. 2. 2 全てのプロセッサを使用する場合

次に CPE も使用して処理する場合のタスク A 終了後の最低限必要な処理時間を求める。CPE でタスク t を処理するとタスク t のパスは com(t)+lb(t) となる。

そのため，以下の手順でタスク A 終了後の最低限必要な処理時間を求める。

① NCPE で処理しているタスクの中で com(t)+lb(t)が最小となるタスク t を探索する(図 6.3)。
（ア）NCPE で処理するタスク t が１つなら，そのタスクのパスが求める値となる。

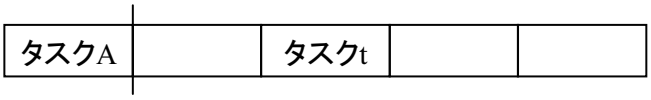

図 6.3 タスク t の選択

② NCPE で処理するタスクの pass の中で最大となる値を探索する。

③ ②の値と com(t)+lb(t)を比較する。

（ア）com(t)+lb(t)が大きいなら②の値が求める値。

（イ）com(t)+lb(t)が小さいなら，タスク t を CPE で処理する(図 6.4)。

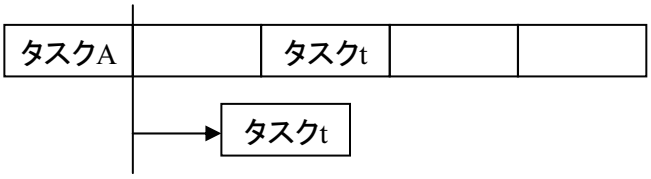

図 6.4 タスク t を処理するプロセッサの選択

④ NCPE で処理するタスクからタスク t を除いて，NCPE で処理するタスクの pass の最大値を求める(図 6.5)。

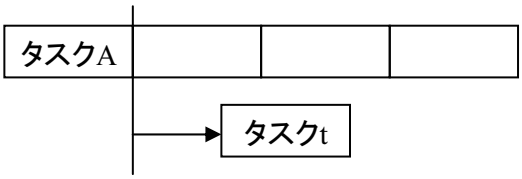

図 6.5 タスク t を NCPE から削除

⑤ ④の値と com(t)+lb(t)を比較。

（ア）com(t)+lb(t)が大きいなら com(t)+lb(t)が求める値。

（イ）com(t)+lb(t)が小さいなら①に戻る。

以上①～④の手順で求めた値にタスク A の処理時間を加えた値がタスク A の下限値となる。

6. 3 下限値の使用方法

求めた下限値は最短経路の近似値として精度が上がっているため，枝切りに使用する。しかし，プライオリティレベルとして使用することはヒューリスティック的であるため，従来の下限値より探索効率が必ず良くなるとは限らない。そこで，２種類の下限値をプライオリティレベルとして使用した探索での初期解が良い解となる下限値をプライオリティレベルとして使用することにした。

7 実行と評価

7. 1 評価方法と実行環境

スケジューリング効率の評価は 2 種類行った。

1 つ目の評価は 2 種類の下限値を用いて評価用タスクグラフを全探索し，最適解を求める際の探索時間を比較した（以下，探索時間評価と呼ぶ)。従来の下限値より通信を考慮した下限値を使用した方が，全探索時間がどのくらい削減されるかでスケジューリング効率の向上を評価する。

2 つ目の評価は 2 種類の下限値を用いて評価用タスクグラフのスケジューリングを行い，設定した時間で求まる暫定解を比較した（以下，暫定解評価と呼ぶ)。この評価では，通信を考慮した下限値を使用した際の実用的な時間内で全探索できないようなタスクグラフに対するスケジューリング効率の変化を評価する狙いがある。従来の下限値より通信を考慮した下限値を使用した方が，設定した時間で求まる暫定解がどのくらい短縮するかでスケジューリング効率の向上を評価する。

使用したマシンの仕様は以下である。

CPU : Intel(R) Xeon(R) X5550 @ 2.67GHz x 2

OS : Linux

RAM : 12GB

これら 2 つの評価は並列探索における相乗効果も期待できるため，上記のマシンで８コアを使用して並列探索を行った。

7. 1. 1 探索時間評価

探索時間評価用タスクグラフ

- 本研究室のランダムタスクグラフジェネレータで生成したタスクグラフ
- タスク数 21
- タスク処理時間 1～30
- 通信時間①1～20 ②1～40 ③1～60
- 後続タスク数 1～4　先行タスク数 1～4
- タスクグラフ 80 個の通信時間を 3 段階設定して，合計 240 個のタスクグラフを生成。

探索時間評価方法

従来の下限値を使用してタスクグラフの 4 プロセッサに対するスケジューリングを行う。その後，今回提案する通信を考慮した下限値を使用して探索時間評価用タスクグラフの 4 プロセッサに対するスケジューリングを行う。以上２つのスケジューリングでの探索時間を比較して，今回提案する通信を考慮した下限値を使用した場合

のスケジューリング効率の向上を評価する。

7. 1. 2　暫定解評価

暫定解評価用タスクグラフ

- 早稲田大学笠原研究室の標準タスクセット[3]
- タスク数 100
- CCR と呼ばれる比を用いて通信時間を付加した。（CCR=通信時間の総和/タスク処理時間総和）[4]
- タスクグラフ 30 個に CCR の値を 1，5，10 と 3 段階設定して，合計 90 個のタスクグラフを生成。

暫定解評価方法

従来の下限値を使用してタスクグラフの 4 プロセッサに対するスケジューリングを 60 秒の探索時間で行う。その後で今回提案する通信を考慮した下限値を使用して暫定解評価用タスクグラフの 4 プロセッサに対するスケジューリングを 60 秒の探索時間で行う。以上２つのスケジューリングで求まった解を比較して，今回提案する通信を考慮した下限値を使用した場合のスケジューリング効率の向上を評価する。

7. 2　実行結果

7. 2. 1　探索時間評価の実行結果

図 7.1，図 7.2，図 7.3 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。

(1) 探索時間評価(通信時間 1～20 のタスクグラフ)結果

ほとんどのタスクグラフにおいて，探索時間の削減率がプラス側に出ている。つまり，通信を考慮した下限値を使用することで探索時間は削減されていることがわかる。全体平均では探索時間が 17.3%削減されていて，90%以上探索時間が削減されているタスクグラフも存在した。

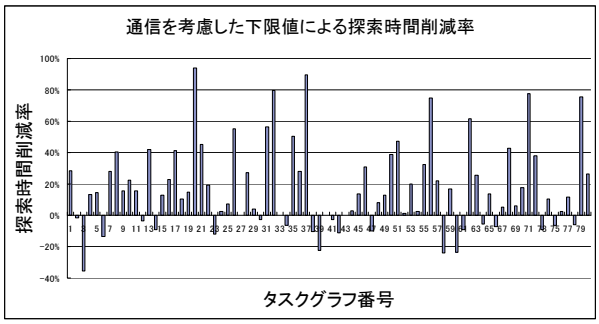

図 7. 1 通信を考慮した下限値による探索時間削減率(1)

(2) 探索時間評価(通信時間 1～40 のタスクグラフ)結果

ほとんどのタスクグラフにおいて，探索時間の削減率がプラス側に出ている。つまり，通信を考慮した下限値を使用することで探索時間は削減されていることがわかる。全体平均では探索時間が 22.1%削減されている。

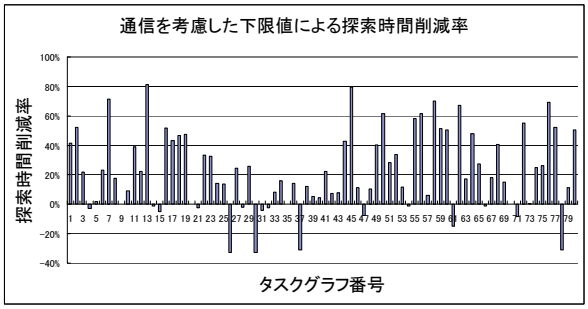

図 7.2 通信を考慮した下限値による探索時間削減率(2)

(3) 探索時間評価(通信時間 1～60 のタスクグラフ)結果

ほとんどのタスクグラフにおいて，探索時間の削減率がプラス側に出ている。つまり，通信を考慮した下限値を使用することで探索時間は削減されていることがわかる。全体平均では探索時間が 22.2%削減されている。

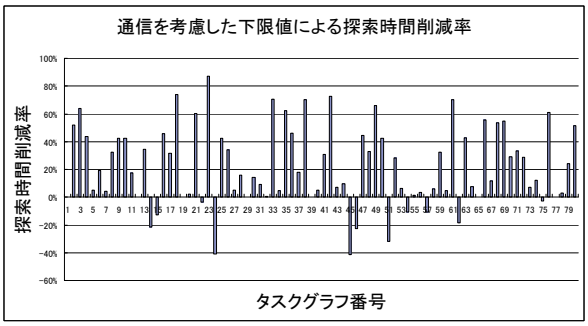

図 7. 3 通信を考慮した下限値による探索時間削減率(3)

7. 2. 2　暫定解評価の実行結果

図 7.4，図 7.5，図 7.6 は横軸がタスクグラフ番号，縦軸が暫定解の短縮率となっている。

(1) 暫定解評価(CCR=1 のタスクグラフ)の結果

ほとんどのタスクグラフにおいて，解の改善率がプラス側に出ている。つまり，通信を考慮した下限値を使用することで解は改善されていることがわかる。全体平均では 1.0%求まる解が改善した。

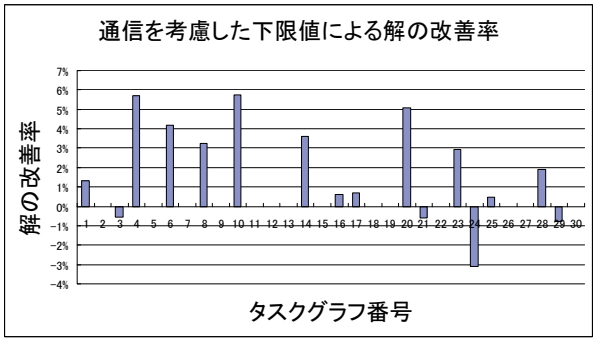

図 7.4　通信を考慮した下限値による解の短縮率(1)

(2) 暫定解評価(CCR=5 のタスクグラフ)の結果

ほとんどのタスクグラフにおいて，解の改善率がプラス側に出ている。つまり，通信を考慮した下限値を使用することで解は改善されていることがわかる。全体平均では 1.3%求まる解が改善した。

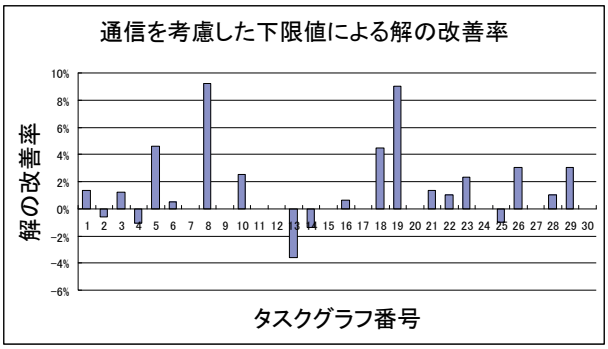

図 7.5　通信を考慮した下限値による解の短縮率(2)

(3) 暫定解評価(CCR=10 のタスクグラフ)の結果

ほとんどのタスクグラフにおいて，解の改善率がプラス側に出ている。つまり，通信を考慮した下限値を使用することで解は改善されていることがわかる。全体平均では 2.8%求まる解が改善した。

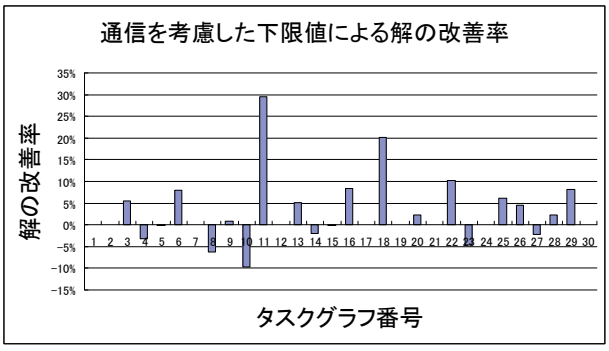

図 7.6　通信を考慮した下限値による解の短縮率(3)

7. 3　評　価

7. 3. 1　探索時間評価と考察

全体を通して 17～20%近い探索時間が削減されている。これは精度の高い下限値を使用することで，良い解の早期発見と枝切りによる探索空間削減の増加につながったことが原因と予測できる。また，探索時間の削減率がタスクグラフに依存していた。これは，通信を考慮して下限値を計算することでの下限値精度の向上率がタスクグラフごとに異なることが原因であると思われる。また，一部のタスクグラフで削減率がマイナスになっていた。これはプライオリティレベルとして使用した際の初期解が良い解である下限値をプライオリティレベルとして使用したことが原因だと思われる。この方法で通信を考慮した下限値がプライオリティレベルとして使用されると，従来の下限値を使用する場合より初期解は良くなるが，暫定解や最適解が先に見つかる保障はない。そのため，従来の下限値を使用した場合より探索時間が延びてしまったと予測している。

7. 3. 2　暫定解評価と考察

解の改善が見られるタスクグラフは限られているが，劇的に悪くなる例は稀であり，通信を考慮した下限値の使用によって解の改善傾向が見られた。解の改善傾向が見られた理由として，通信を考慮した下限値を使用することで，良い解の早期発見と枝切りによる探索空間削減の増加につながったことが原因と予測できる。また，通信を考慮した下限値によって大きなスケジューリング効率向上の効果が出なかった理由には，精度の高い下限値を計算できたとしても，必ず早期に良い解が発見できるとは限らないといったことが挙げられる。今回のスケジューリングアルゴリズムでは，下限値を元にヒューリスティック的に良い解があると思われる順に探索空間を探索するが，実際に良い解が探索空間のどこに存在しているかは不明である。そのため，良い解の早期発見と枝切りによる探索空間の削減の効果が少なくなった可能性が考えられる。

7. 3. 3　全体の評価と考察

最適解を求められる全探索での評価では，制限時間内で行う暫定解評価より探索時間が長く，探索範囲が大きかったため，良い解の早期発見の効果と枝切りによる探索空間削減の効果が大きく，スケジューリング効率の向上に対する効果が大きかったと思われる。また，最適解を求めることが出来ない暫定解の評価では，探索時間と探索範囲が限定されることで，良い解の早期発見の効果と枝切りによる探索空間削減の増加が抑えられてしまい，スケジューリング効率の向上に対する効果が少なくなったと考えられる。全体的には，スケジューリング効率向上の幅がタスクグラフに依存しているが，評価では決められた時間内での暫定解が短縮傾向にあり，全探索時間は減少傾向にある。つまり，精度の高い下限値を使用することで，良い解の早期発見と枝切りによる探索空間削減の増加につながり，スケジューリング効率が向上したと考えられる。

８　終わりに

タスクグラフにはタスク処理時間，通信時間，先行タスク，後続タスクなどにそれぞれ異なった特徴がある。そのため，スケジューリング問題では全てのタスクグラフにおける同様なスケジューリング効率の向上は難しいが，今回提案した通信を考慮した下限値を使用した限定操作でのスケジューリングでは，スケジューリング効率の向上傾向が確認できた。

今回は下限値探索に時間をかけないようにするために複数の条件を設けた。この条件を少なくすることでさらに精度の高い下限値が作成可能である。また，劇的なスケジューリング効率の向上には新たな探索アルゴリズムの考案が必須であると考えている。

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，謝辞を表します。

参考文献

[1] 笠原博徳,「並列処理技術」，コロナ社，1991

[2] 津田耕治,「通信を考慮したタスクスケジューリング問題の探索解法」，成蹊大学大学院工学研究科情報処理専攻修士論文，2004

[3] STG(Standard Task Graph Set), http://www.kasahara.elec.waseda.ac.jp/schedule/index.html　2010 年 1 月

[4] O.Sinnen, “Task Scheduling for Parallel Systems”, Wiley-Interscience (2007)