成蹊大学理工学研究報告
J. Fac. Sci. Tech., Seikei Univ.
Vol.44 No.1 (2007) pp.17-23

通信遅延を考慮したタスクスケジューリングのためのタスク粒度解析

尾高　輝[*1]，甲斐　宗徳[*2]

Task Granularity Analysis for Task Scheduling Taking Account of Communication Overhead

Akira ODAKA[*1], Munenori KAI[*2]

ABSTRACT : Task scheduling problems belong to the class of strong NP-hard combinatorial optimization problem. Taking account of communication overhead into task graphs makes these problems more difficult to solve by any search algorithms because the search spaces become larger. In this paper, we propose a task granularity analysis method in order to make the search spaces narrower and to shorten the search time by task fusion. Some basic patterns and their conditions of partial task graphs to be merged into a single task are proposed. In order to shorten the schedule length as possible, a task duplication method is also proposed. We examine both methods to our search algorithm for randomly generated task graphs and practical task graphs for the applications, such as robot control, sparse matrix multiplication and SPEC fpppp, and obtain good effectiveness.

Keywords : task scheduling, task fusion, task duplication, heuristic algorithm, combinatorial optimization problem

（Received March 26, 2007）

１．はじめに

プロセッサの処理能力は年々向上しているが，ユーザの処理内容はより複雑化，大規模化している。従って単一プロセッサによる処理では時間が掛かりすぎる場合がある。そこで処理速度を向上させるためにマルチプロセッサシステムの研究が進められている。

マルチプロセッサシステムで効率よく高速に処理をさせるためには，マシンが持つ性能を十分に引き出す必要がある。そこで並列実行可能なタスクを抽出し，そのタスク集合をスケジューリングすることが重要なキーポイントとなる。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。この問題に対して，従来の優良なスケジューリング結果を求めるとされているスケジューリング手法がいくつかあるが，これらはタスク間の通信時間を考慮していない。しかし実際には通信時間が無視できないほど，大きくかかってくるケースもある。通信遅延を考慮しない最適スケジュールにそのまま通信を加えても最適解になっているとは限らないので，通信遅延を考慮したスケジューリングは非常に重要になってくる。しかし，タスク間の通信を考慮すると，タスクの組み合わせが多くなり，探索空間は以前よりもさらに広大化することになる。つまりタスク数が少し増えるだけでも探索数は激増してしまう。そこで抽出されたタスクグラフの無駄な並列性をあらかじめ取り除くためにタスク融合を行い，探索空間を削減することで，効率の良い探索を行うことができる。このような理由からスケジューリング前にスケジューリングアルゴリズムに都合の良い粒度を決定することは非常に効果的である。またタスクの粒度解析は，どんなスケジューリング手法の組み合わせに対しても有効である。

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

２．タスクスケジューリング問題

タスクを節点，タスク間の先行制約を有向辺で表すことで，並列計算をモデル化したものがタスクグラフであ

る。タスクグラフは閉路のない重み付き有向グラフである。タスク間の先行制約は，タスク間にデータ依存が存在することにより生じる。タスクの重みはタスクの実行にかかる時間を表す。図 1 にタスクグラフの例を示す。

タスクスケジューリング問題とは，図 1 のようなタスクグラフを能力の等しい m 個のプロセッサに，先行制約のある n 個のタスクを，処理割り込みなしで割り当てるという条件の下で，各タスクを与えられた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最適化問題である。

３．従来のスケジューリング手法

3．1　CP（Critical Path）法，CP/MISF(Critical Path/ Most Immediate Successors First)法 [1)]

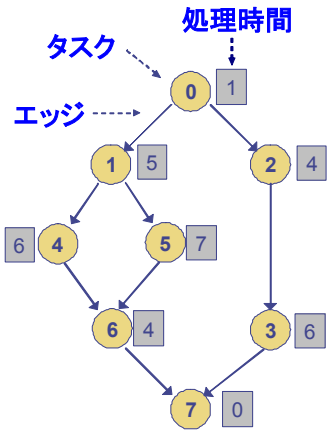

図１　タスクグラフ

CP 法は通信を考慮していないヒューリスティックアルゴリズムで，各タスクから最終タスクまでかかる処理時間の下限値（プライオリティレベル）を求め，これの大きいタスクほど実行優先順位を高くする。このプライオリティレベルを基に，リストスケジューリングを行う。CP/MISF 法は CP 法を改善したもので，同じプライオリティレベルが複数存在する時は，直接後続タスク数の多い方のプライオリティを高くし，先にアイドルプロセッサに割り当てる。

3．2　DF/IHS(Depth First/Implicit Heuristic Search)法 [1)]

DF/IHS 法とは，CP/MISF 法による優れたヒューリスティック効果を取り入れた，分枝限定法(Branch＆Bound)による，探索アルゴリズムである。分枝限定法は，現在求められている暫定解よりも良い解が得られそうな枝を探索し，良い解が得られそうにない部分木は下限値を使用し切り落としていく。

４．計算機モデルの定義

本研究の対象とする計算機モデルは，任意の方法で相互結合された均一のプロセッサにより構成される並列計算機である。プロセッサはそれ自身のプライベートメモリを持ち，共有メモリを持たない。プロセッサ間の通信はすべて相互結合網を通して一台のプロセッサから一台のプロセッサに明示的にメッセージを送ることで行われる。同じプロセッサ上で実行されるタスク間においてメッセージの交換が必要な場合，データはローカルメモリから直接読み込まれる。先行制約を持つ２つのタスクが異なるプロセッサで処理されるとき，プロセッサ間で生じる通信は並列処理に起因するオーバヘッドであり，Oliver らは文献 3) において，図 2 のように取り扱うべきと述べている。図２はタスク A とタスク B が異なったプロセッサに割り当てられた時に生じる通信オーバヘッドを表したもので，図中の各パラメータは以下の通りである。

- T：タスクの処理時間
- e_{AB}：タスク B とタスク B の間で必要とされるデータ授受のための通信時間
- L：通信リンク
- Os：送信準備のオーバヘッド
- Or：受信準備のオーバヘッド
- P：プロセッサ

Os と Or はシステムに依存する値で，e_{AB} は送受信データの量に依存するものである。本研究では簡単のため $Os+e_{AB}+Or$ を通信オーバヘッドとして取り扱うものとし，タスク A とタスク B が同じプロセッサに割り当てられた場合，通信時間は 0 とする。

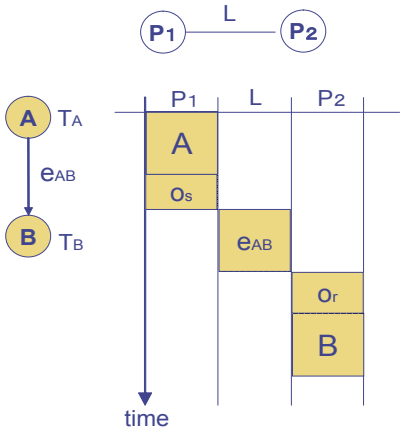

図２　タスク間の通信

５．通信を考慮したスケジューリング

5．1　通信オーバヘッド

タスク間での通信時間は無視できないほど大きいケースがあるが，従来のスケジューリング手法ではタスク間の通信時間を考慮していない。通信遅延を考慮しない最適スケジュールにそのまま通信時間を加えても最適解になっているとは限らないので，本研究では，このような通信遅延を考慮したスケジューリングを行うために DF/IHS 法の拡張を行った。

図１のタスクグラフにおいて，先行制約のあるタスク間でデータ通信が行われる必要があるため，その通信時間が各有向エッジに新しく付加されることになる。本研究では，図３(a)に示すように，１つのプロセッサから他のプロセッサへの通信は同時に行うことができず，図３(b)のように逐次的に通信しなければならないようなプロセッサ間の相互結合網を仮定する。ただし，送信元と受信先のペアが異なる通信は同時に行なうことができるものとする。

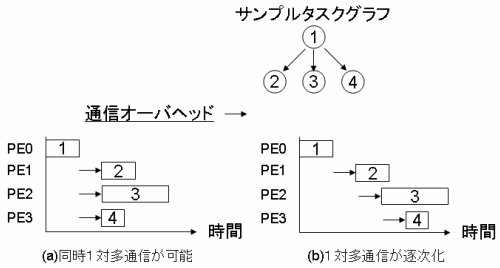

図３　マルチプロセッサ相互結合網と通信オーバヘッド

5．2　3S(Shrinking Search Space)法 [2)]

通信遅延を考慮して全探索を行う際，各深さで割り当てタスクの組み合わせだけでなく，通信によってスケジューリング結果が異なるプロセッサへの割り当てに関する順列も考慮に入れなくてはならない。そのため探索空間が以前よりも広大化することになる。そこで，スケジュール候補をとる必要のない探索空間を分析して除外することにより，実行時間の短縮をはかる。

３S 法においてキーとなる方法は３つある。１つめは，必要なアイドルタスクの抽出法で，これはすべてのプロセッサへの割り当てを考慮した時に，必ずしもアイドルタスクが必要だとは限らないので，不要なアイドルタスクを割り当てない手法である。２つめは，タスク割り当て制限法で，これは各プロセッサへタスクを割り当てたときに，明らかに通信を無意味に行っている割り当てを見つけ，探索を行わない手法である。３つめは，通信における送受信パターンの削減法で，これは各タスクをプロセッサへ割り当てを行うにあたって，通信における送受信パターンが格段に増大するので，効果的にスケジューリングを行うために，割り当てタスクの通信を受け取る時間に注目し，スケジューリング結果が重複する確率の高い探索を行わない手法である。

６．タスク粒度解析によるタスク融合

タスク融合を行うねらいは，タスク数を減らし，タスク割り当ての組み合わせを減らすことで，探索空間を減少させることによりヒューリスティック探索，全探索の両探索においてより早く有用な暫定解を求めることである。またプロセッサ資源の無駄を省くこともできる。

6．1　タスク融合の概念

図４のタスクグラフのように直接先行後続関係にあるタスク，または並列関係にあるタスク間においてタスクの融合はなされる。この処理時間は融合される前の２つのタスクの処理時間の和であり融合後のタスクの先行後続関係は崩れない。

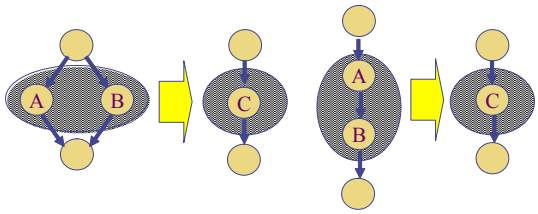

図４　タスク融合の仕方

6．2　通信時間の扱い

図５においてタスク A, B 間の送受信データとタスク A, C 間の送受信データに重複がある可能性があるので，タスク融合により通信時間を融合後の通信時間にどのように反映させるかで３種類の方法を用意した。

- 送受信データに重複がない場合
  - $e_{AD}= e_{AB}+e_{AC}$
- 送受信データが完全に重複している場合
  - $e_{AD}= \max\{e_{AB},e_{AC}\}$
- 送受信データにいくらかの重複がある場合

  例えば $e_{AB}>e_{AC}$ で，20%の重複があると仮定すると
  - $e_{AD}= e_{AB}+(e_{AC}*0.8)$

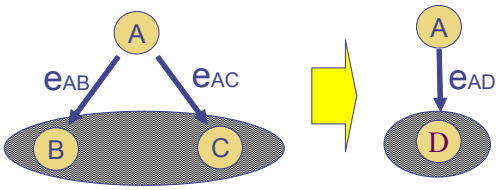

図５　融合時の通信時間

6．3　タスク融合による通信オーバヘッドの軽減

タスク融合によって，複数回に分けて通信されていたデータが１回の通信にまとめられることで，通信回数を削減し，オーバヘッドを軽減することができる。一般的に，通信オーバヘッドは無視できない大きさであるため，スケジューリングモデルにオーバヘッドを考慮することは精度の良い並列プログラムを生成するのに重要である。図６(a)は，３台のプロセッサ PE0，PE 1，PE 2 でタスク A，B，C を含む処理を実行する並列プログラムを模式化したものである。タスク A，B を別のプロセッサで実行するため，PE1 への通信を２回行う。しかし，図 6(b)のように，タスク B を PE 0 で実行することで，通信を１回にまとめることができ，オーバヘッドの軽減につながる。タスク融合により，タスク A，B の処理が完了してから通信を開始することになるため，送信開始時刻が遅くなるが，この例では，受信オーバヘッド Or が１回で済むので，PE 1 で実行する他のタスクの実行時刻を遅れさせることなく，タスク C の開始時刻を早めることができる。

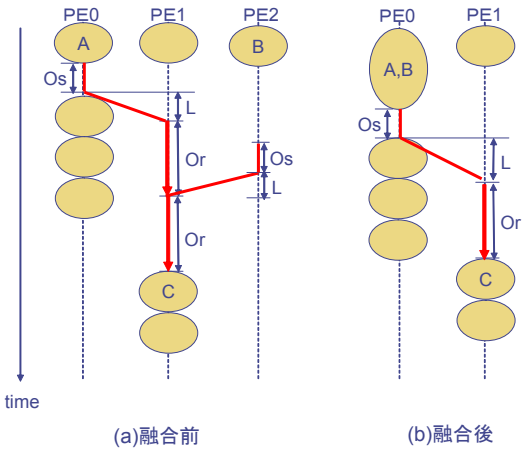

図６　通信の一括化

6．4　ヒューリスティックタスク融合法

本手法では生成されたタスクが細かすぎた場合に，ヒューリスティックなタスク融合手法を用いて並列性をあまり犠牲にすることなく複数のタスクを融合し１つの粗いタスクを生成する。図７にこの手法が適用される基本パターンを示す。

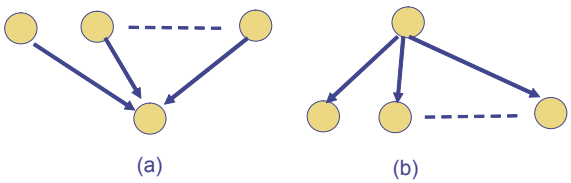

図７　タスク融合の基本パターン

この手法では，タスクグラフ中で図７の様な部分タスクグラフを取り出し，その最小並列処理時間が逐次処理時間より大きい場合にそれらのタスクを融合する。これを融合条件とする。例えば図７(a)の様な部分タスクグラフにおいて，融合条件を満たした場合，図８のように部分タスクグラフの最適なタスク割り当てを見て，同じプロセッサに割当てられているタスク同士を融合する。図７(b)の場合も基本的に図７(a)と同じである。

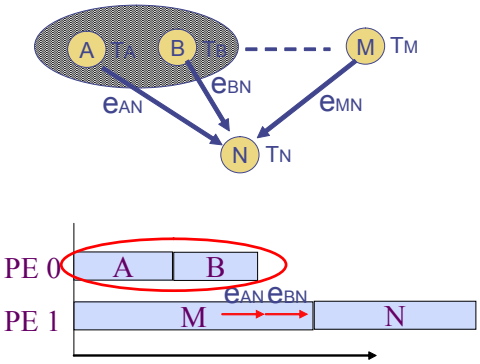

図８　並列関係の融合

この単純な方法を部分タスクグラフに繰り返し適用することにより，図９のような木構造を持つタスクは，タスク間の通信時間に対するタスク処理時間の割合に応じて複数のあるいは１つの粗いタスクに融合される。

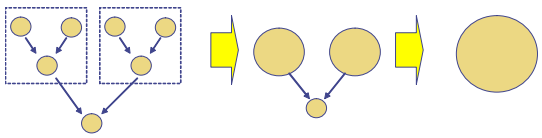

図９　タスク融合の適用例

7．タスク複製

7．1　複製の条件

タスク複製のねらいは，タスクを複製することにより複製元タスクの後続エッジを分散させ，後続タスクの処理開始時間を早めることである。しかし，タスクを複製

することで複製元の先行タスクからの通信が増え，逆にスケジュール長が伸びてしまう可能性がある。そこで本手法では局所的には，その複製によってスケジュール長が伸びない場合のみ，その該当タスクを複製対象タスクとした。タスク複製の条件は，複製対象タスクとその直接後続タスクを部分タスクグラフとし，複製を行った結果複製前よりも部分タスクグラフの最小並列処理時間が短くなる場合にタスクの複製を行う。図 10 の部分タスクグラフでは，タスク 2 を複製することによりタスク 5 の開始を早めることができ，結果としてこの部分タスクグラフのスケジュール長が短くなることがわかる。このような場合にタスク複製を適用する。

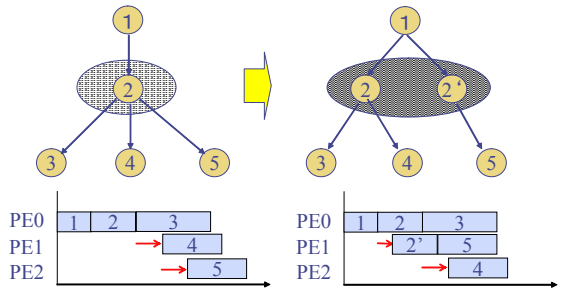

図 10　タスク複製の適用例

7. 2　後続エッジの分散

複製を行った際，後続タスクへのエッジをどのように分散するかについて説明する。小規模なタスクグラフにおける最適解のタスク割り当てを観察すると，プライオリティレベルの高いタスクは通信を行わず，逐次処理されることが多いことがわかる。そこで本手法では，図 11 のように，タスク 1 が複製条件を満たすとき，その複製をタスク 1'とすると，後続タスク 2～5 はプライオリティレベルが高いものから順に，タスク 1 とタスク 1'交互にその後続タスクとなるようにエッジを分散する。

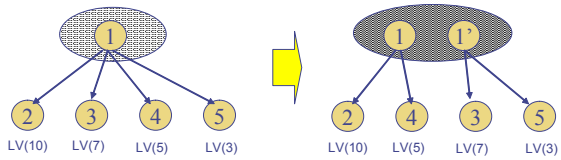

図 11 後続エッジの分散

8．評　価

タスク融合とタスク複製の効果を調べるために，ランダム生成されたタスクグラフといくつかのアプリケーションプログラムのタスクグラフを用いて評価を行った。

［実行環境］

ＣＰＵ:PentiumD(2.79GHz)，メモリ:1GB

［指標］

タスク融合・複製を行う前後において

① スケジュール長(初期解)

② スケジュール長(一定時間探索後の暫定解)

③ タスク数

を比較する。

探索時間については，タスクの規模や通信を考慮している点から，現実的な時間内ですべての探索を終えることは困難であるため，ランダムタスクグラフでは 60 秒，アプリケーションプログラムのタスクグラフでは 300 秒で探索を打ち切り，その時点での暫定解を評価の対象とした。

8. 1　ランダムタスクグラフによる評価

乱数によって様々な形状のタスクグラフを作成したものを評価用タスクグラフとして用いた。乱数の発生方法は一様乱数である。

- サンプルタスクグラフ数：100
- 割り当てプロセッサ数：16

－各タスクグラフのパラメーター－

・各タスクグラフのタスク数：50
・直接先行タスク数：1～3
・直接後続タスク数：1～3
・タスク処理時間：1～100
・通信時間：1～100
・並列度(処理時間の和/クリティカルパス)：1～5

8. 1. 1　スケジュール長(初期解)の比較

ランダム生成された 100 通りのタスクグラフはいずれもタスク数が 50 であるが，6.で述べたタスク融合を適用したところ，平均してタスク数が 17.1%減少した。そしてすべてのタスクグラフについてタスク融合前後におけるスケジューリング初期解の比較を行った。ここで，初期解とは 5.2 で述べた 3S 法が探索の中で最初に求める解候補であり，CP/MISF 法でタスクの割り当て時刻に通信による遅延を考慮しながら決定されたスケジュールと一致するものである。図 12 は各タスクグラフの融合前後のスケジュール長の伸縮率を融合前のスケジュール長を 1 として散布図で表したものである。図から分かるように，融合後のスケジュール長(初期解)と，融合前のスケジュール長(初期解)を比較すると 7 割以上のタスクグラフでスケジュール長が改善され，全体平均でスケジュール長を 5.1%削減することができた。

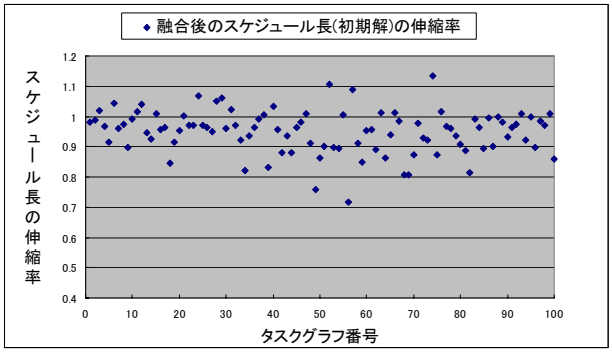

図 12 融合前後のスケジュール長(初期解)の比較

8. 1. 2 スケジュール長 (一定時間探索後の暫定解) の比較

融合前後におけるスケジュール長 (60 秒探索後の暫定解) の比較を行った。図 13 は各タスクグラフの融合前後のスケジュール長の伸縮率を融合前のスケジュール長を 1 として散布図で表したものである。図から分かるように，融合後のスケジュール長(60 秒探索後の暫定解)は，融合前のスケジュール長(60 秒探索後の暫定解)より平均で 5.9%削減することができた。

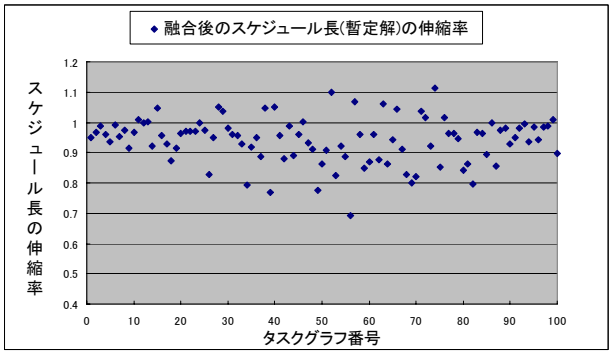

図 13 融合前後のスケジュール長(暫定解)の比較

前述の通り，全探索を行うことは事実上不可能であるので，限られた時間の中で早期に，より優良な解を求めることが重要である。また，全探索をすれば膨大な探索時間がかかる中で，探索の初期段階の実用的な時間内では，できる限り良質の解候補が存在する探索空間を対象として探すことが重要である。そこで融合前後における 60 秒間の探索で初期解からどれだけスケジュール長が短縮されたかを比較すると，融合をしないで探索を行った場合のスケジュール長の短縮率は平均 2.5%であったのに対し，融合を行ってから探索をした場合のスケジュール長の短縮率は平均 3.3%であった。このことからタスク融合によって，スケジュール前に無駄な探索空間を除外したことで早期に優良な解を求めることができたと言える。

8. 1. 3 タスク融合とタスク複製を同時に適用した場合

100 タスクグラフ中３つのタスクグラフでタスク複製が適用された。適用された数が少ないのは，複製によってスケジュール長が悪化することのないように，本手法では複製の条件を厳しくしているからである。タスク複製が行われた３例において，融合のみを行った場合の初期解・暫定解と比較してスケジュール長の変化はなかった。

8. 2 標準タスクセットによる評価

早稲田大学笠原研究室で公開されている標準タスクセット(アプリケーションプログラム) [4]を評価用タスクグラフとして用いた場合の評価を示す。

タスクグラフの概要は以下のようである。

・Robot control(タスク数 88，エッジ数 131)
・Sparse matrix solver(タスク数 96，エッジ数 67)
・SPEC fpppp(タスク数 334，エッジ数 1145)

しかしこれらのタスクグラフには通信時間が含まれていないため，

CCR(communication to computation ratio)
=通信時間の総和／処理時間の総和

をパラメータとし，CCR を 1～3 まで変化させてランダムに通信時間を加えることで各タスクグラフ 30 種類の通信パターンのタスクグラフを生成した。

8. 2. 1 スケジュール長(初期解・暫定解)の比較

30 種類の通信パターンを加えた標準タスクセットにタスク融合を適用したところ，平均してタスク数が 21.7%減少した。そしてすべてのタスクグラフについてタスク融合前後におけるスケジュール長(初期解・300 秒間探索後の暫定解)の比較を行った結果が表 1 である。融合前よりも暫定解が改善されたタスクグラフは全体の約 82%で，最もスケジュール長が良くなったものでは約 32%短縮された。逆に最も悪くなってしまったものではスケジュール長が約 10%伸びていた。平均的には融合によってスケジュール長を十分改善できたと言える。また，Robot control において，ランダムタスクグラフ同様，融合をしないで探索を行った場合よりも，融合を行ってから探索をした方がスケジュール長の短縮率が平均で 0.8%上がった。Sparse matrix solver と SPEC fpppp に関して

表 1 評価結果

	融合前のスケジュール長(初期解)との比較	融合前のスケジュール長(暫定解)との比較	融合前のタスク数との比較
robot	平均4.4%の削減	平均4.9%の削減	平均22.7%の削減
sparse	平均2.9%の削減	平均2.9%の削減	平均26.9%の削減
fpppp	平均10.8%の削減	平均10.8%の削減	平均15.5%の削減

は初期解と暫定解が同じであった。従って，効果の度合いはタスクグラフに依存するが，実アプリケーションにおいてもタスク融合により無駄な探索空間が削減されたことで，早期に優良な解を発見できたことになる。

8. 2. 2　タスク融合とタスク複製を同時に適用した場合

Robot control において 5 種類のタスクグラフにタスク複製が適用された。タスク複製が適用されたタスクグラフに関してはタスク融合のみを行った場合の暫定解よりも，平均で 4.6%スケジュール長が削減された。またタスク融合のみを行った初期解・暫定解と比較して，タスク複製適用により初期解・暫定解が悪くなってしまったタスクグラフはなかった。

９．　終わりに

本研究では，通信遅延を考慮したタスクスケジューリングのための前処理としてタスク粒度解析を行い，並列ログラムに適切な粒度をスケジュール前に決定することを行った。結果，タスク融合を行う前と比較して，スケジューリング時の探索空間が削減され，実用的な時間内でより優良な解を得ることができた。またタスク複製により通信を減らすことでスケジュール長の削減を行うことができた。

今後の課題として，本手法では部分タスクグラフを取り出し，処理時間と通信時間の割合によって局所的に融合・複製の判定を行っているが，さらに精度の良い粒度解析を行うためには数値だけではなく，タスクグラフの形状などを考慮に入れた手法が必要であると考えられる。

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) Oliver Sinnen,Leonel Augusto Sousa, Frode Eika Sandnes, “Toward a Realistic Task Scheduling Model”,IEEE Transactions On Parallel And Distributed Systems, VOL.17, NO. 3, MARCH 2006
4) 標準タスクセット, http://www.kasahara.elec.waseda.ac.jp/schedule/index.html, 2007 年 3 月現在参照可