成蹊大学理工学研究報告
J. Fac. Sci.Tech., Seikei Univ.
Vol.55 No.2 (2018)pp.7-14

タスクグラフの階層的マクロタスク化とそのタスクスケジューリング手法の提案

長谷川 幹*1, 甲斐 宗徳*2

Detecting Hierarchically Structured Macro-tasks within a Task Graph and Its Task Scheduling Method

Motoki HASEGAWA*1, Munenori KAI*2

ABSTRACT : One of the key methods for the optimized parallel processing of any program is task scheduling. This intends to minimize the execution time of the entire program by determining an optimal schedule for allocating all tasks, which are processing units comprising the program, onto available processor elements into pieces. Due to complexity of calculation and being a large-scale problem, the optimization problem is extremely challenging to solve with the current algorithms in a practical amount of time. Authors have studied the method of partially optimizing task graph as a technique of task scheduling for large-scale problems. This paper reports on scheduling method that supports hierarchical macro for task groups.

Keywords : Task scheduling, Combinatorial optimization problems, High performance search method

(Received November 30, 2018)

1．はじめに

マルチコアやマルチプロセッサといった現代の並列コンピューティングにおいて、効率よく高速に処理を行うための方法の１つとしてタスクスケジューリングがある。

タスクスケジューリングは、並列処理環境で計算資源(Processor Element : PE)に最適なタスクの割り当てを決定し処理パフォーマンスを最大限に引き出す技術である。このタスクスケジューリングは、組み合わせ最適化問題の計算複雑度のクラスの中で強NP困難というクラスに属する[1]。この強NP困難に属する問題は、問題の規模が大きくなるほど、最適解を得るのに必要となる探索時間が指数関数的に増大してしまう。従って、大規模なタスク集合でのスケジューリングは、実用的な時間内に最適解を得ることは不可能に近い。

過去の代表的なタスクスケジューリングアルゴリズムである、リストスケジューリング[2]、CP法[3]、CP/MISF法[4]や、 DF/IHS法[4]などは、各マシン間の通信時間を考慮しているものではない。実際の並列実行時には各マシン間でのデータ通信にかかる通信時間による影響が大きい可能性もあり、通信遅延を考慮したスケジューリングアルゴリズムが重要となる。この通信遅延を考慮したスケジューリングは、組み合わせ最適化問題における探索の回数をさらに増やし、より困難なものとする。

このような背景から大規模なタスク集合でも短時間で最適解、またはそれに近い解を求められる通信遅延を考慮したスケジューリングアルゴリズムが必要とされている。

大規模なタスク集合を実用的な時間内にスケジューリングを行う方法として、部分最適化を用いた階層的スケジューリングによるタスクスケジューリングの高速解法について研究を行った。特に、汎用的なプログラムに対応するためにベンチマークプログラムからなるタスクグラフに対して有効な階層的スケジューリングアルゴリズムの構築を行った。

2．タスクスケジューリング

2．1 タスクグラフ

タスクスケジューリングでは、問題を表現するために、

*1：理工学専攻博士前期課程学生
*2：理工学研究科理工学専攻教授 (kai@st.seikei.ac.jp)

タスクをノード、タスク間の先行制約を有向エッジで表した、タスクグラフと呼ぶDAG(Directed Acyclic Graph)を用いる。処理の開始と終了を明確にするため、タスクグラフには、コストが 0 のダミーノードとしてスタートノードとエンドノードが必要に応じて追加されている。図 2.1 は、スタートノードをタスクS、エンドノードをタスクEとしたサンプルタスクグラフである。

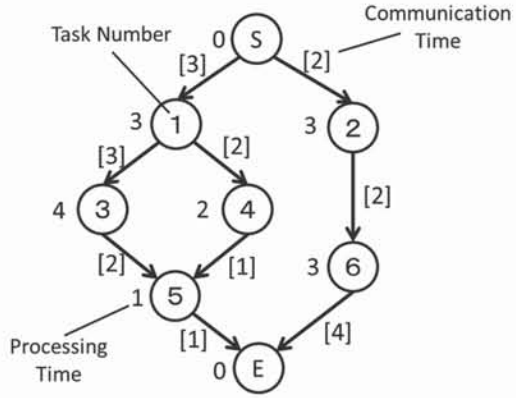

図 2.1　タスクグラフ 1

ノード内の数字はタスク番号(Task Number)、ノード横の数字はタスクの処理コスト(Processing Time)を表している。本研究では先行後続関係にあるタスク同士がそれぞれ別のPEに割り当てられた際に、実際の並列実行時のデータの転送時間として、マシン間の通信遅延を考慮する。エッジ横の角括弧で囲まれた数字はその際にかかる通信コスト(Communication Time)を表している。

2. 2　ガントチャート

タスクスケジューリングの割り当て結果を可視化するために、縦軸にPE、横軸に処理時間を取った時系列チャートとしてガントチャートを用いる。本研究では、この他に縦軸に各PEのデータの送受信のタイミングとしてSend,Recvの 2 つを追加したガントチャートを用いる。図 2.2 は、図 2.1 のタスクグラフ 1 のスタートノードS、タスク 1、タスク 2 の 3 つのタスクを 2 つのPEに割り当てた様子を表したガントチャートである。

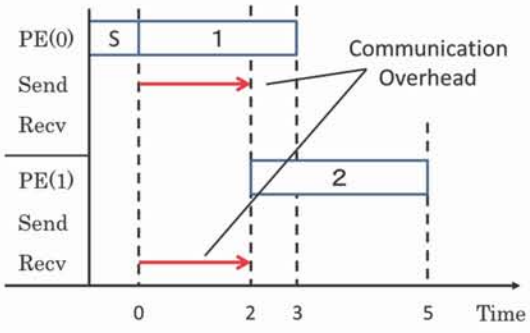

図 2.2　ガントチャート

各タスクは、自身の全ての先行タスクの処理が終了してからでなければ、処理を開始することができない。また、エッジで結ばれたタスクjとその先行タスクであるタスクiが異なるPEに割り当てられた場合は、通信遅延(Communication Overhead)を考慮し、エッジに付加された通信コストC(i,j)を払う必要がある。

2. 3　タスクの下限値とプライオリティレベル

タスクの下限値とは、そのタスクからエンドノードまでに最低でもかかる処理時間の合計値である。本研究では、この下限値の大きさによってタスクの割り当て優先度（プライオリティレベル）を持たせ、探索に利用する。

3．スケジューリングアルゴリズム

組み合わせ最適化問題を解くアルゴリズムは大きく 2 つに分けられる。一つ目は、人間の経験則に基づいて構築され、短時間で近似解を求めることが目的の「ヒューリスティックアルゴリズム」である。二つ目は、問題の最適解を求めることが目的の「最適化アルゴリズム」である。

3. 1　ヒューリスティックアルゴリズム

代表的なヒューリスティックアルゴリズムとして、リストスケジューリング[2]、CP(Critical Path)法[3]、CP/MISF(Most Immediate Successors First)法[4]などがある。

リストスケジューリングは、スケジューリングアルゴリズムの基盤となるアルゴリズムである。実行可能なタスク（レディタスク）を処理が終了しているPE（アイドルPE）に割り当てる。レディタスク数がアイドルPEを上回る時のタスクの割り当て順序は、タスク番号の順番に選択される。

CP法は、レディタスク数がアイドルPE数を上回る場合にプライオリティレベルの高いレディタスクからアイドルPEに割り当てを行う。

CP/MISF法は、CP法においてプライオリティレベルが等しいタスク同士の割り当て順序を後続タスク数で比較して決定する方法である。

3. 2　最適化アルゴリズム

代表的な最適化スケジューリングアルゴリズムであるDF/IHS(Depth First / Implicit Heuristic Search)法[4]は、組み合わせ最適化問題に最も効果のある最適化手法とされている分枝限定法(Branch & Bound)にCP/MISF法のヒューリスティック効果を取り入れた探索アルゴリズムである。

このDF/IHS法は、探索木を構成しながら分枝限定法の限定操作を行い、深さ優先探索によって暫定解を更新しながら全探索を行って問題の最適解を求める。

3. 3　通信遅延を考慮したスケジューリングアルゴリズム

これまでに紹介したCP法、CP/MISF法、DF/IHS法は、全てPE間の通信遅延を考慮しないスケジューリングアルゴリズムである。本研究ではDF/IHS法に、各タスクから限定された範囲の後続タスクまでの通信遅延を考慮した下限値を求めるREDIC(Remaining Distance Including Communication Overhead)法[5]を加えたアルゴリズム[6]を用いる。

4．タスクグラフの階層的探索手法

大規模なタスク集合を実用的な時間内にタスクスケジューリングする方法として、タスクグラフを複数回に分けてスケジューリングする階層的なスケジューリング方法の研究を行った。階層的探索手法は、タスクグラフの中から部分的に最適化可能なタスク群を探索して検出する「部分タスクグラフの検出」と、部分タスクグラフと元の全体タスクグラフを複数回スケジューリングする「階層的スケジューリング」を行う。

以下に、階層的探索手法の手順を示す。

① タスクグラフの中から部分最適化可能なタスク群を検出する。
② 検出したタスク群についてのみスケジューリングを行い、PE への割り当てを決める。
③ 元のタスクグラフの中で、②でスケジューリングしたタスク群を 1 つのタスクとしてみなし、タスクグラフ全体についてスケジューリングを行う。

上記の手順において、①で検出したタスク群を「部分タスクグラフ」、②で部分タスクグラフについて行うスケジューリングを「部分スケジューリング」、部分タスクグラフに対して元のタスクグラフを「全体タスクグラフ」、全体タスクグラフについてのスケジューリングを「全体スケジューリング」、③で全体タスクグラフの中で部分タスクグラフを 1 つのタスクとしてみなすものを「マクロタスク」、とそれぞれ定義する。

4. 1　部分タスクグラフの検出手法

部分タスクグラフとは、「入口ノード」、「出口ノード」、その中間に存在する「中間ノード群」の三要素から構成されるタスク集合が 1 つのタスクグラフとして成り立つものと定義する。図 3.1 では、タスク{1,3,4,5}が部分タスクグラフに該当する。前述の手順①のタスクグラフの中からタスク群を探索し、部分タスクグラフを検出する手法は、上記の部分タスクグラフの定義に該当するタスク群を任意に手動で決定する。

4. 2　階層的スケジューリング

改めて、階層的探索手法の手順を以下に示す。

① タスクグラフから部分タスクグラフを検出
② 部分タスクグラフについて部分スケジューリングを行う
③ 全体タスクグラフについて全体スケジューリングを行う

階層的スケジューリングとは、4.1 で検出した部分タスクグラフに対してのみ行う「部分スケジューリング」と、全体タスクグラフの中で部分タスクグラフを 1 つの「マクロタスク」としてみなして行う「全体スケジューリング」とにタスクスケジューリングを分割して行う方法である。タスク数が少ないスケジューリングに分割することにより、スケジューリングにかかる時間を減少させることが目的である。

次の図 4.1 は、図 2.1 のタスクグラフ 1 から部分タスクグラフを検出した様子である。

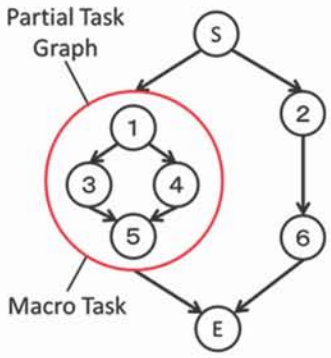

図 4.1　部分タスクグラフとマクロタスク

部分スケジューリングは、部分タスクグラフの内部タスク{1,3,4,5}についてのみ最適化を行うスケジューリングのことを言う。ここで、全体のタスクグラフの中で部分タスクグラフを 1 つの「マクロタスク(M)」としてみる。全体スケジューリングは、マクロタスク(M)を含むタスク{S,M,2,6,E}についてスケジューリングを行う。

4. 2. 1　階層的スケジューリングによるスケジューリング時間の減少

階層的スケジューリングに対して、従来の最適化スケ

ジューリング手法を「フラットスケジューリング」と定義する。

タスクグラフのタスク数をn、部分タスクグラフの内部タスク数をmとすると、上記の階層的スケジューリングは次の条件下 $n > m > 1$ で行われることが前提となる。タスク数nで行われるフラットスケジューリングに対し、階層的スケジューリングは、タスク数mでの部分スケジューリングとタスク数(n−m+1)での全体スケジューリングに分割される。

タスク間の先行制約を考慮しない環境下での、フラットスケジューリングと階層的スケジューリングにおける探索回数の比較を表 4.1 に示す。

表 4.1　スケジューリング方法と探索回数の比較

スケジューリング方法	探索回数
フラットスケジューリング	n!
部分スケジューリング	m!
全体スケジューリング	(n−m+1)!
階層的スケジューリング	(n−m+1)! + m!

表 4.1 のフラットスケジューリングの探索回数と階層的スケジューリングによる探索回数を比較したとき、乗算の方が明らかに大きくなるため、

$n! > (n-m+1)! + m!$

が成り立つ。タスク間の先行制約、分枝限定法における分枝限定操作を考慮しても上記の式の大小関係は成り立つことが分かっており[7]、階層的スケジューリングによって探索回数が削減されることが分かる。よって階層的スケジューリングによってスケジューリング時間を減少させられることが分かる。

4. 3　部分スケジューリング方法

部分スケジューリングは、検出した部分タスクグラフに対して行うスケジューリングである。この方法は、単純にタスク数が少ないタスクスケジューリングであるため、従来の最適化手法を用いてスケジューリングを行った。

4. 4　全体スケジューリング方法

全体スケジューリングは、実体がタスクグラフである「マクロタスク」を割り当て対象に含むため、従来の最適化手法では実現できない。なぜならば、従来の最適化手法は、「1 つのタスクは 1 つの計算資源(PE)にのみ割り当てる」ことが前提条件であった。しかし、マクロタスクは内部に複数のタスクを持ち、そのタスクを並列実行するため、複数のPEに割り当てなければならない。正しく階層的スケジューリングを行うために、タスクが必要とするPE数に対応した割り当てをするようなアルゴリズムを導入した。

4. 4. 1　必要PE数に対応したタスクの割り当て

複数のPEに割り当てるマクロタスクにスケジューラが対応するため、スケジューリングの手順の中で割り当て可能なタスクを選ぶ時に、タスクの必要PE数を見てその数だけ割り当てるようにする。

まず、タスクに自分がいくつのPEに割り当てる必要があるのかという情報を追加した（必要PE数)。そして、割り当て可能なタスクを選択するときに、選択したタスクを格納するキューに必要PE数が 2 以上のときは、その数だけ増やす作業を追加した。これにより、タスクが必要とするPE数に正しく割り当てることが可能になった。

5．階層的スケジューリングアルゴリズムの拡張

5. 1　複数の部分タスクグラフに対応した階層的スケジューリングアルゴリズム

ここまで説明した階層的探索手法では、図 5.1 のような部分タスクグラフ(マクロタスク)同士が続いているようなタスクグラフにおいて、それぞれのマクロタスクの先行後続関係が正しく認識できていない。これは、従来手法ではタスクの先行タスクとなりうるものは 1 つのタスクであることが前提であったためである。しかし、マクロタスク同士に先行後続関係がある場合、ただ単にお互いを先行タスク、後続タスクとするだけでなく、その内部のどのタスクとエッジがつながっていたのかを認識する必要がある。これは、スケジューリングではマクロタスクというタスクのまとまりを扱うが、実際の並列実行時にはその内部のタスク間で通信が行われるからである。

図 5.1 を例にして、タスク{1,2,3,4}からなるマクロタスクをM1、タスク{5,6,7,8}からなるマクロタスクをM2とする。また、M1 とM2 はそれぞれ必要PE数が 2 であるとする。この時、実際の並列実行時に通信が必要なタスクは、タスク 4 とタスク 5 の間である。M1 がPE(0)とPE(1)に割り当てられており、M2 の割り当て先を探索しているとする。ここで、M2 の先行タスクがM1 であるとすると、M2 から見た時にタスク 5 の先行タスクであるタスク 4 がPE(0)とPE(1)のどちらに割り当てられているのかが分からない。そのため、マクロタスクの先行タスクがマクロタスクであった時に探索が正しく行われないという課題があった。

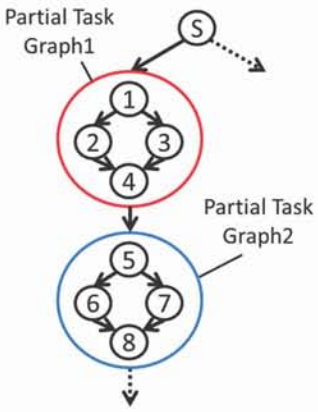

図 5.1　複数の部分タスクグラフ

この問題を解決するために、マクロタスク内部のスタートノードとエンドノードに対して、マクロタスクにする前の先行タスクと後続タスクを記憶させた。図 5.1 では、マクロタスクM1 のエンドノードであるタスク 4 の後続タスクをM2 ではなくタスク 5 とし、マクロタスクのスタートノードであるタスク 5 の先行タスクをM1 ではなくタスク 4 とした。これにより、マクロタスク同士が先行後続関係であったとしても、正しい割り当てが可能となる。

以上のアルゴリズムの導入により、複数の部分タスクグラフを持つタスクグラフに対しても階層的スケジューリングを行うことが可能となる。

5. 2　マクロタスク内のアイドル時間の活用

タスクスケジューリングによって、PEへのタスクの割り当てが求まったとき、PEはすべての時間においてタスク処理を行うわけではなく、タスクの処理と処理の間に空き時間が存在する。この空き時間を、PEのアイドル時間と呼ぶ。図 5.2 の両矢印がその例である。

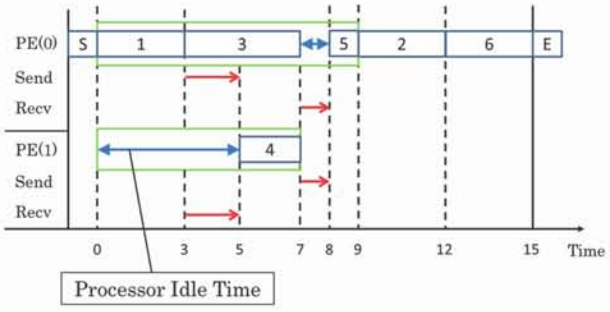

図 5.2　PEのアイドル時間

図 5.2 では、PE(1)の時間 0 から時間 5 がアイドル時間となっているが、実際には、このアイドル時間にタスク 2 を処理可能である。このように、階層的スケジューリングによって最適解を求めるためには、マクロタスク内に存在するアイドル時間をマクロタスク外のタスクの割り当てに活用する必要がある。

図 5.3 は、PE(1)のアイドル時間へタスク 2 を割り当てた結果である。このように、アイドル時間へタスクを割り当てることによって、問題の最適解が求まることがある。

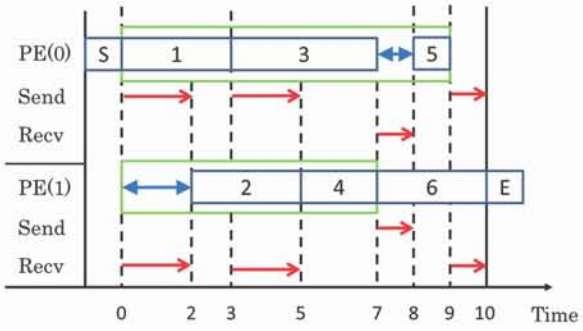

図 5.3　アイドル時間へのタスクの割り当て

しかし、全てのアイドル時間に対してどのタスクが割り当て可能かどうか全探索することは、スケジューリング時間を増やすことになってしまう。そこで、アイドル時間への割り当て探索を有効かつ限定的にするために、探索を行う条件をPE側とタスク側に対して 2 つの制約を考案した。

5. 2. 1　PEのタスク占有率による制約

PEのタスク占有率とは、マクロタスクの割り当てられたPEの処理時間の内、実際にタスク処理を行っている時間の割合である。

あるPEに割り当てられたマクロタスクの実行開始時間をt_1、実行終了時間をt_2 とすると、時間t_1 からt_2 の内、実際にタスクが割り当てられている時間の割合をそのPEのタスク占有率とする。例えば、図 5.2 のPE(1)のタスク占有率は、8÷9×100≒88.9%、PE(2)のタスク占有率は、2÷7×100≒28.6%である。

アイドル時間への割り当て探索によるスケジューリング時間の増加を最低限にするために、探索を試みる条件として、「マクロタスクが割り当てられたPEのタスク占有率が 50%である」ことを階層的スケジューリングアルゴリズムに適用した。

5. 2. 2　タスクのプライオリティレベルによる制約

アイドル時間への割り当て探索を行う際のタスク側の条件として、アイドル時間の前後に割り当てられているタスクのプライオリティレベルの範囲に収まるプライオリティレベルを持つタスクのみを探索対象にした。

図 5.2 では、PE(1)におけるアイドル時間は時間{0-5}

である。ここで、アイドル時間の後ろに割り当てられているタスク 4 のプライオリティレベルを参照し、タスク 4 のプライオリティレベルよりも高い値を持つタスクのみを割り当ての探索対象とする。

図 5.4 は、図 2.1 のタスクグラフの各タスクのプライオリティレベルをCP/MISF法[4]によって求め、左から昇順に並べたリストである。図 5.4 より、タスク 4 よりも高いプライオリティレベルであるのはタスク 2,6 の内、タスク 2 のみであるため、タスク 2 のみをアイドル時間への割り当て探索対象とし、タスク 6 は探索の対象外とする。

S	1	2	3	4	6	5	E

図 5.4　CP/MISF法[4]によるプライオリティリスト

5.2.1 と上記のプライオリティレベルの制約により、アイドル時間への割り当て探索を必要最低限にすることにより、不要な探索の増加を防ぐことが可能になる。

6．評価実験

これまでに述べた階層的探索手法を実装した階層的スケジューラ(Hierarchical Scheduler : HS)と従来のDF/IHS法によって最適解を求める最適化スケジューラ(Optimize Scheduler : OS)とで比較実験を行った。実験環境を以下に示す。

- CPU : Intel® Xeon®X5560 @ 2.8 GHz
- OS : Cent OS 5.6
- RAM : 19GB

実験に使用したタスクグラフは、以下の 2 種類のベンチマークプログラムを基に、本研究室がタスク処理コスト・通信エッジコストを算出し、生成した 2 種類のタスクグラフとする。

・Mi Bench Version1.0 : basicmath _large.c

・NAS Parallel Benchmarks : IS(size 'S')

上記に加えて、早稲田大学笠原研究室によって公開されているタスク数 88 個の標準タスクセット[9]の計 3 種類を実験の対象とする。

この 3 種類のタスクグラフに対してタスクの処理コスト、エッジの通信コストを以下の条件において 100 パターンランダム生成した。

- タスク処理コスト：最大 20、最小 5
- エッジ通信コスト：最大 10、最小 1

また、スケジューリング時間の上限を 60 分とし、60 分時点で探索が完了していない場合は、その時点での暫定解を求まった解とした。求まった解（求解値）、探索回数、スケジューリング時間（求解時間）を両スケジューラにおいて比較した。

タスクグラフ	タスク数	求解値比率			探索回数比率			求解時間比率		
		平均	最大	最小	平均	最大	最小	平均	最大	最小
basicmath	51	0.941	1.043	0.852	0.734	0.972	0.525197	1.000	1.000	1.00000
NPB IS(size 'S')	43	0.980	1.042	0.729	0.631	2.023	0.000002	0.525	1.845	0.00001
標準タスクセット	88	0.969	1.050	0.837	0.853	2.434	0.000378	0.776	1.000	0.00069

表 6.1　実験結果

表 6.1 は、OSの結果を基準の 1.0 としてHSによる 3 項目の結果を比率で示している。

求解値では、3 種類のタスクグラフ全てで比率の平均値が 1.0 を下回っており、HSによってOSの求解値よりも良い解を探索できたことが分かる。

探索回数においても、3 種類全てで比率の平均値が 1.0 を下回っており、HSはOSよりも少ない探索回数で探索を完了したことが分かる。

求解時間について、basicmathベンチマークの実験では、100 個全てで求解時間の上限 60 分では探索が完了しなかったため、平均値・最大値・最小値いずれも 1.0 となっている。一方、NPB IS(size 'S')では、求解時間が平均で約 47%減少した。

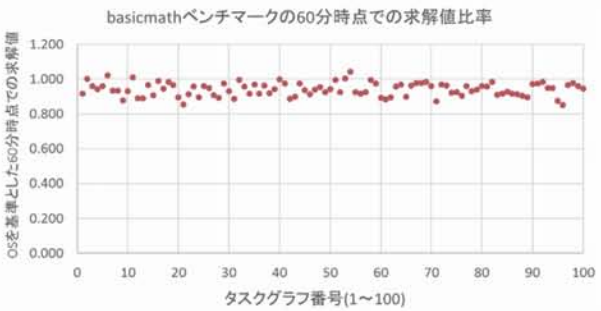

図 6.1　basicmathベンチマーク結果の求解値比率

図 6.1 は、basicmathベンチマークに対する実験の求解値の比率を示した散布図である。OSの求解値を基準の 1.0 とし、各タスクグラフのHSによる求解値の比率を表している。

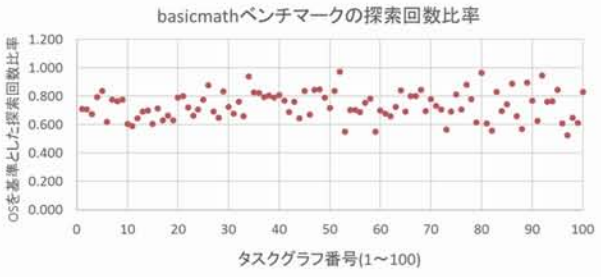

図 6.2　basicmathベンチマーク結果の探索回数比率

図 6.2 は、basicmathベンチマーク実験の探索回数の比率を示した散布図である。求解値と同様にOSの探索回数を基準の 1.0 としてHSの探索回数の比率を表している。

2 つの散布図から、basicmathベンチマークに対して階層的スケジューリングアルゴリズムが特に有効であったと分かる。

探索回数については、100 個全てのタスクグラフにおいてOSよりもHSの探索回数の方が少ない結果であった。

7．おわりに

大規模なタスク集合に対してより高速に最適解または近似解を求めるための手法として、階層的スケジューリング手法の研究を行った。これまでの階層的スケジューリング手法では、ベンチマークプログラムからなるタスクグラフのような大規模なタスク集合や、1 つのタスクグラフに複数の部分タスクグラフが存在するようなタスク集合に対しては、効果を発揮することができなかった。そこで、複数の部分タスクグラフへの対応や、PEのアイドル時間の活用法の構築によって、多種多様なタスクグラフに対して有効な階層的スケジューリングアルゴリズムの構築を行った。

評価実験では、ベンチマークプログラム 2 種と標準タスクセットを元にしたタスクグラフに対して、階層的スケジューラと最適化スケジューラで実験を行った。結果から探索回数について、basicmathでは約 27%減少、NPB IS(size 'S')では約 37%減少、標準タスクセットでは約 15%減少した。求解時間については、NPB IS(size 'S')で約 47%減少、標準タスクセットで約 23%減少した結果を得た。

今後の課題について、評価実験の結果から、タスク数 43 個のNPB IS(size 'S')の探索回数の減少率よりも、タスク数 88 個の標準タスクセットの探索回数の減少率の方が少なかった。理想では、タスク数が増えるほど、階層的スケジューリングによって探索回数が減少した方が望ましいが、逆に減少率が低い結果となった。このことから、アイドル時間への割り当て探索手法に対して更なる最適化が必要であると考える。

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] Edward G. Coffman, "Computer and Job-shop Scheduling Theory", John Wiley and Sons, 1976.

[3] T.C. Hu, "Parallel sequencing and assembly line problems", Operations Research, Novem-ber/December 1961 Vol.9 No. 6 pp.841-848,1961

[4] H. Kasahara and S. Narita, "Practial Multiprocessor Scheduling Algorithm for Efficient Parallel Processing", IEEE Transactions on Computers, Vol. 33, No. 11, pp. 1023-1029, November. 1985

[5] Masahiko Utsunomiya, Ryuji Shioda, Munenori Kai, "Heuristic search based on branch and bound method for task scheduling considering communication overhead", Proc. of IEEE Pacific Rim Conference on Communications, Computers and Signal Processing, pp.256-261, 2011.

[6] 渋谷知則，栗田浩一，甲斐宗徳，" 通信遅延を考慮したタスクスケジューリング問題ための並列分枝限定法とその評価"，FIT2014(第 13 回科学技術フォーラム)，第 1 分冊, pp.149-154，2014

[7] 柴山修 "部分スケジューリングを活用した全体タスクスケジューリング問題の高速解法" 成蹊大学理工学部卒業論文，2015

[8] 渋谷知則,"通信遅延を考慮したタスクスケジューリング問題の並列解法-タスクグラフの部分最適化に基づく探索優先順位の改良-" 成蹊大学理工学研究科 修士論文，2016

[9] http://www.kasahara.elec.waseda.ac.jp/schedule/