成蹊大学理工学研究報告
J. Fac. Sci.Tech., Seikei Univ.
Vol.58 No.1 (2021)pp.5-10

Deep Learning を用いた階層的な部分タスクグラフ検出法の構築

田邑 大雅[*1]，甲斐 宗徳[*2]

Construction of Hierarchical Partial Task Graph Detection Method Using Deep Learning

Taiga TAMURA[*1], Munenori KAI[*2]

ABSTRACT：Since task scheduling problems belong to a class of the strong NP-hard combinatorial optimization problem, the required scheduling time increases exponentially as the number of tasks increases. Therefore, we find some small subtask graphs that can be optimally scheduled in the overall task graph, and solve them individually as a scheduling problem. Then, each subtask graph can be treated as one macro task in the whole task graph. This reduces the apparent number of tasks in the overall task graph, reduces the scale of the task graph, and significantly reduces the search time that depends on the number of tasks. We call this hierarchical scheduling. However, this partial task graph detection has a drawback that it becomes a combinatorial optimization problem by itself. Therefore, in this paper, we construct and evaluate a method for detecting partial task graphs using deep learning.

Keywords：Task scheduling, Deep Learning, Convolutional Neural Network

(Received June 18, 2021)

1．はじめに

プロセッサの処理能力向上の問題において，半導体の集積度の限界により，CPUの処理能力の向上に限界が近づいていることから，複数のプロセッサを用いて演算処理を行うマルチプロセッサシステムが活発に利用されている．このマルチプロセッサシステムなどの並列コンピューティングにおいて，効率よくかつ高速に処理を行うためには，タスクスケジューリングが重要なポイントと考えられる．しかし，タスクスケジューリングは，強NP困難な最適化問題であるため，問題の規模が大きくなると実用的な時間内でスケジューリングを終えることが難しくなるという問題点が存在する．現在，この課題を解決しタスクスケジューリングを有効活用できるような研究が進められている．当研究室では，複数回にわたってスケジューリングを行う階層的スケジューリング[1]によるスケジューリング時間短縮の研究が進められてきた．しかし，この階層的スケジューリングにおける部分タスクグラフの検出には，それ自体が組み合わせ最適化問題になり得るという問題点があり，アルゴリズムの改善による時間短縮には限界を感じてきた．そこで，Deep Learningを部分タスクグラフ検出に適用して処理の高速化を図ることを目的とする．

2．タスクスケジューリングとタスクグラフ

タスクスケジューリング問題では，問題を可視化するために，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．Fig. 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．ノード内の数字はタスク番号，ノード横の数字はタスクの処理コストを表している．エッジ横の角括弧内の数字は，先行後続関係にあるタスク同士がそれぞれ別

[*1]：理工学部情報科学科学生
[*2]：理工学専攻教授 (kai@st.seikei.ac.jp)

のPEに割り当てられた際に，データの転送時間としてかかるマシン間の通信コストを表している．

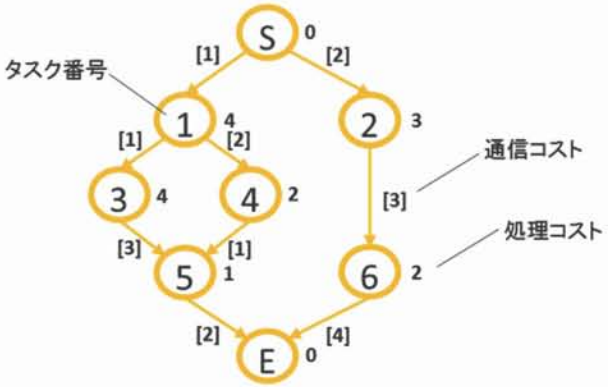


Fig. 2.1　タスクグラフ

また，タスクスケジューリングでは，スケジューリングの結果どのような割り当てになったかを可視化するため，縦軸にPE，横軸に処理時間を取った時系列チャートであるガントチャートを用いる．当研究室では，これの縦軸に各PEのデータの送受信のタイミングとしてSendとRecvの 2 つを追加したものを使用している．Fig. 2.2 は Fig. 2.1 のタスクグラフのスタートノードS，タスク 1，タスク 2 の 3 つのタスクを 2 つのPEに割り当てた際の様子を表したガントチャートである．

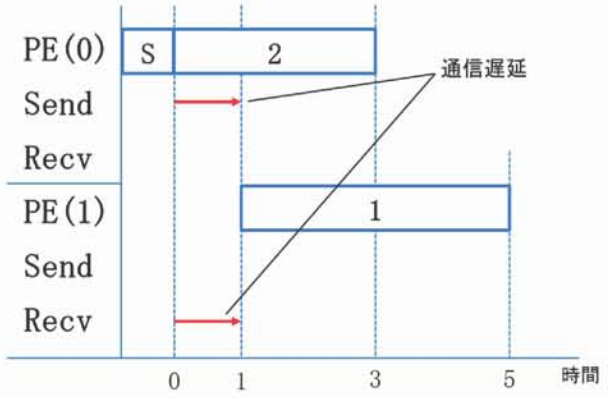


Fig. 2.2　ガントチャート

3．階層的スケジューリング

階層的スケジューリングとは，従来のように一度に全てのタスクでスケジューリングを行うのではなく，複数回にわたって階層的にスケジューリングを行うという手法である．タスクグラフ内のタスクの集合(部分タスクグラフ)を検出し，部分タスクグラフのみでスケジューリングを行った後，それを 1 つのタスク(マクロタスク)として全体でスケジューリングすることで，タスク数を減らしたスケジューリングを可能とする手法である．これにより少ないタスク数でのスケジューリングを行うことになり，処理時間の短縮に繋がる．Fig. 3.1 はタスクグラフにおける部分タスクグラフとマクロタスクを表したものである．

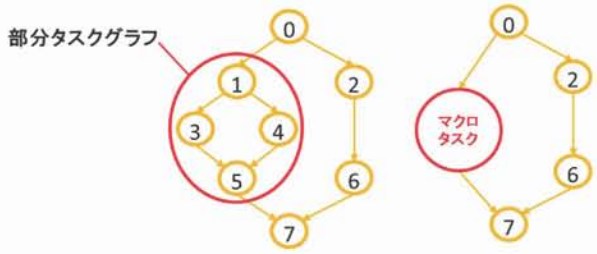


Fig. 3.1　部分タスクグラフとマクロタスク

4．Deep Learningの導入

部分タスクグラフになるノードの組み合わせを求めることは，それ自体が組み合わせ最適化問題となり得るため，これを求めるアルゴリズムの開発には限界を感じた．そこで処理の高速化を図るため，階層的スケジューリングにおける部分タスクグラフの検出にDeep Learningを適用する手法の研究を行った．

Deep Learningとは，多層のニューラルネットワークによる機械学習の手法の 1 つである．問題となる入力データと，その解答となる出力データを大量に与えると，それらから自動で特徴や法則を見出し学習する．これを利用するには，特徴や法則を見つけることをある程度期待できるデータを与える必要がある．またDeep Learningには，一度学習を済ませてしまえば，解を求めるのに必要な時間が短いという性質を持つため，階層的スケジューリングにおける部分タスクグラフの検出にかかる探索時間の問題を解決するのに適している．学習に用いるニューラルネットワークのモデルは，入力データを二次元配列で表現することにより，画像認識の分野で実績を上げている畳み込みニューラルネットワーク(Convolutional Neural Network：CNN)を用いる．

4．1　学習させるデータの作成

Deep Learningを利用するためには，学習させるデータの形式を定める必要がある．本研究においては，部分タスクグラフの検出にこれを利用するため，入力データにはタスクグラフ全体の情報，出力データにはタスクグラフの中に存在する部分タスクグラフの情報をそれぞれ与え，学習させる．これにより，タスクグラフの情報を入力すると，タスクグラフの中に存在する部分タスクグラフの情報が出力されることを目指すものとする．

4．1．1　入力データの形式

入力データは，タスクグラフの情報を二次元配列で表現する．入力データに格納する情報は，各タスクの処理コスト，各タスク間の通信コスト，各タスク間のエッジの繋がり，の 3 要素とする．二次元配列を用いて，その行と列をそれぞれ先行タスク番号と後続タスク番号とし，各要素にタスク間の通信コストを格納する．また、対角要素には各タスクの実行時間を格納する．Fig. 4.1 は入力データの形式を表したものである．

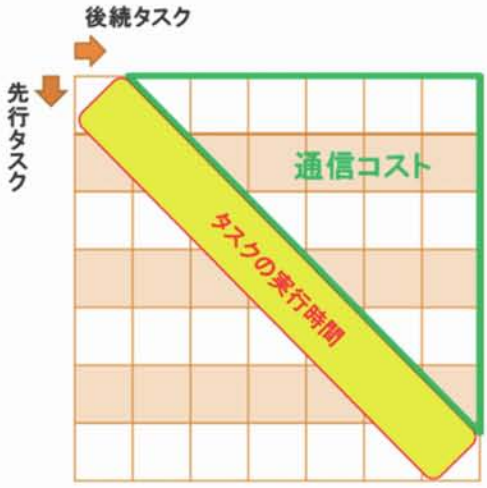


Fig. 4.1　入力データ形式

4．1．2　出力データの形式

出力データは，タスクグラフ内に含まれる部分タスクグラフの情報を一次元配列で表現する．出力データに格納する情報は，マクロタスク番号，内包するタスク番号，の 2 要素とする．要素番号をタスク番号と見た配列の各要素に識別値を格納した。

識別値とは，タスクグラフ内に部分タスクグラフが複数存在する場合を考慮し，各タスクがどの部分タスクグラフに所属するのかを識別できるように割り振る値である．また，そのタスクがいずれの部分タスクグラフにも所属しない場合，識別値は 0 とすることでこれを表現する．Fig. 4.2 は出力データの形式を表したものである．

0	1	2	3	4	5	6
0	1	1	0	1	1	0

部分タスクグラフ所属情報

Fig. 4.2　出力データ形式 1

4．2　タスクグラフの結合による教師データ作成

Deep Learningによる学習に用いるデータは，精度の高い学習を期待できるようにするため，教師データとして用意する学習データに変換するタスクグラフは，含まれる部分タスクグラフの情報が正確なものにしたい．そこで，あらかじめ含まれる部分タスクグラフの情報が正確なタスクグラフを作成し，それを形式変換して教師データとする．含まれる部分タスクグラフの情報が正確なタスクグラフは，タスクグラフに別のタスクグラフを挿入，結合することで実現する．Fig. 4.3 はタスクグラフの挿入，結合による目的のタスクグラフ作成の様子である．

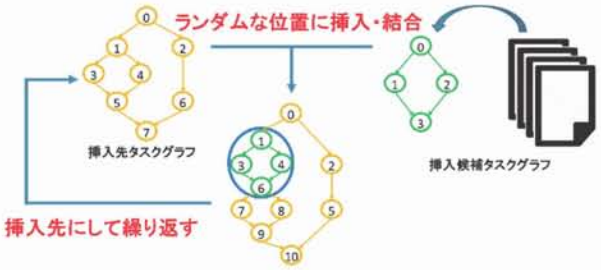
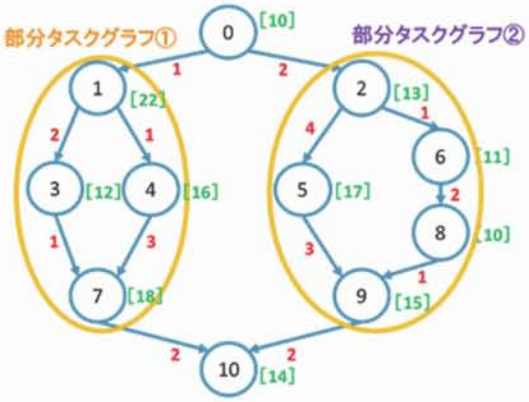


Fig. 4.3　タスクグラフの結合

上記の操作をタスクの総数が必要な数に達するまで繰り返すことで，学習データ作成に必要なタスクグラフを用意した．また，挿入候補となるタスクグラフは複数用意することで作成されるタスクグラフの種類を増やした．Fig. 4.4 はタスクグラフとそれから作成された教師データの一例である．



入力データ

10	1	2	0	0	0	0	0	0	0	0
0	22	0	2	1	0	0	0	0	0	0
0	0	13	0	0	4	1	0	0	0	0
0	0	0	12	0	0	0	1	0	0	0
0	0	0	0	16	0	0	3	0	0	0
0	0	0	0	0	17	0	0	0	3	0
0	0	0	0	0	0	11	0	2	0	0
0	0	0	0	0	0	0	18	0	0	2
0	0	0	0	0	0	0	0	10	1	0
0	0	0	0	0	0	0	0	0	15	2
0	0	0	0	0	0	0	0	0	0	14

出力データ

0	1	2	1	1	2	2	1	2	2	0

Fig. 4.4　教師データの一例

4．3　学習モデルの設計

本研究では，CNNを用いた学習モデルの構築を行った．CNNは，画像認識の分野で実績を上げており，入力データの形式を画像と同じように二次元配列で表現することにより，これを用いることを可能とする．CNNとは，Deep Learningにおいて研究されているノードの接続方法の 1 つであり，畳み込み層とプーリング層で構成されるニューラルネットワークである．畳み込み層とプーリング層では，決められたサイズの領域にフィルタ処理をかけて次の層への対応付けをする．

Fig. 4.5 は，本研究で使用した，CNNを用いて構築した学習モデルである．入力をフィルタサイズの異なる 4 つの畳み込み層で畳み込み，それらを結合する一連の処理を「Layer Unit」と定義する（Fig. 4.5 では「L」と表記）．Layer Unitを 3 並列に接続したものを 2 つと，Layer Unit を 1 つ直列に接続する．Layer Unitの接続パターンは，いくつかのパターンで実験を行い，学習にかかる時間とのバランスを考慮した上で決定した．また，プーリング層ではMax Poolingを使用した．

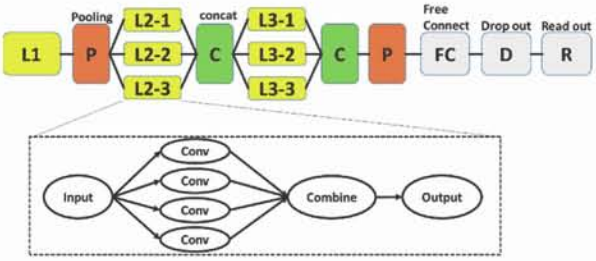


Fig. 4.5　CNNによる学習モデル

5．学習結果

4.1 と 4.2 で作成したデータセットを 4.3 で構築したモデルで学習する．今回は，タスク数 12，20，24 のタスクグラフを結合によりそれぞれ 50,000 個用意し，それらをまとめて形式を変換することにより，3 種類の教師データを用意する．

5．1　学習精度

ステップ数を 8000 に変更し，用意した 3 種類すべてのデータセットを教師データとして，それぞれ学習を行った．Fig. 5.1 は，その時の学習の精度の遷移を表したものである．タスク数 12 のデータセットでは，90%以上の精度を確保することができた．

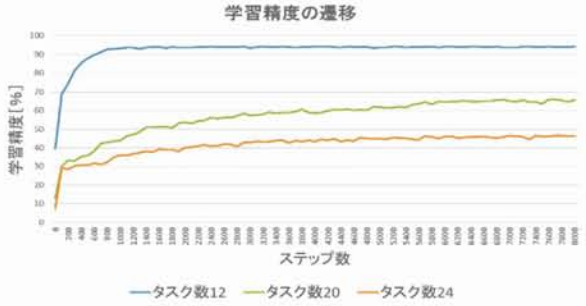


Fig. 5.1　学習精度の遷移

5．2　所属する部分タスクグラフの誤検出

学習したモデルを使用し，タスクグラフから部分タスクグラフの検出を行ったところ，Fig. 5.2 に示すように複数の部分タスクグラフに所属する可能性があるタスクについて，そのいずれにも所属しないとして検出される形の誤検出が確認された．

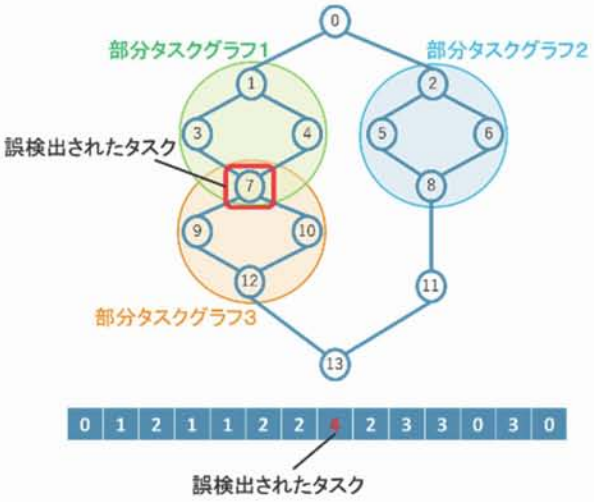


Fig. 5.2　飛び地の形の誤検出

また，Fig. 5.3 に示すようなタスクグラフ内のエッジで分岐した先のタスク同士が同一の部分タスクグラフに所属するとして検出される誤検出も確認された．

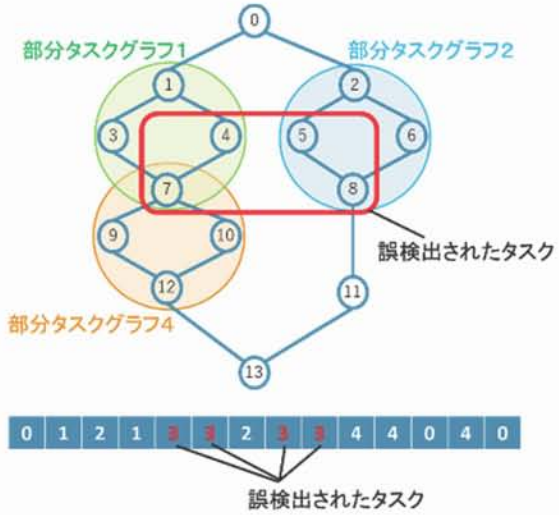


Fig. 5.3　分岐先のタスクの誤検出

6．階層的な部分タスクグラフの検出法

誤検出の問題は，与える教師データからの学習の難度が高いことが原因のひとつだと考えられる．そこで，一度に全ての部分タスクグラフを検出するのではなく，階層的スケジューリングにおける部分タスクグラフの検出と同様に，その階層に存在する部分タスクグラフのみを検出する手法に切り替えた．Fig. 6.1 は階層的な部分タスクグラフの検出法を表したものである．

Fig. 6.1　階層的な部分タスクグラフの検出法

6．1　入力データ形式の修正

検出の目的を変更するにあたり，Deep Learningの学習に用いる学習データの形式にも変更を加える．階層的な部分タスクグラフの検出を行うにあたって，入力データには新たに前の階層までに部分タスクグラフに所属したタスクの情報を追加する必要がある．そこで，Fig. 6.2 に示すように，対角要素に格納された値の内，前の階層までに部分タスクグラフに所属したタスクに該当するものについてその値を-1 に変更した．

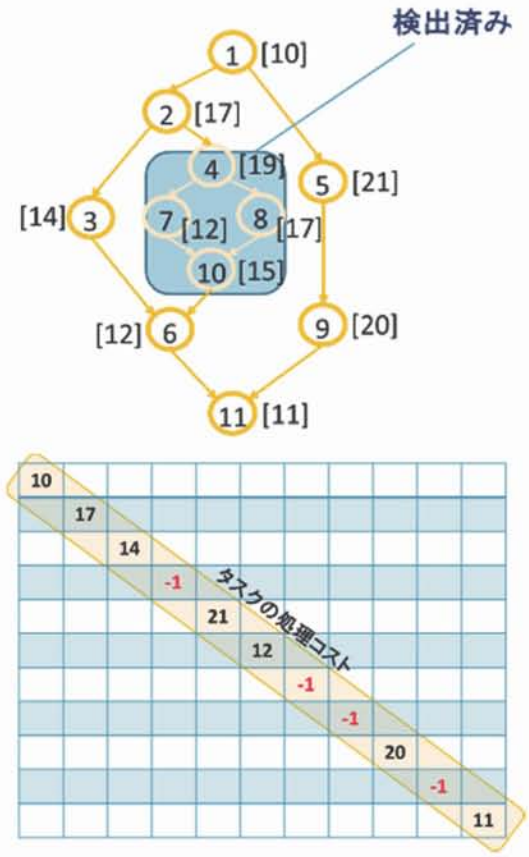


Fig. 6.2 修正後の入力データ形式

6．2　出力データ形式の修正

変更にあたり出力データに必要な情報は，その階層における部分タスクグラフの情報のみとなるため，複数の識別値ではなく，Fig. 6.3 に示すように，部分タスクグラフに所属するか否かを 0or1 で表現して格納した．

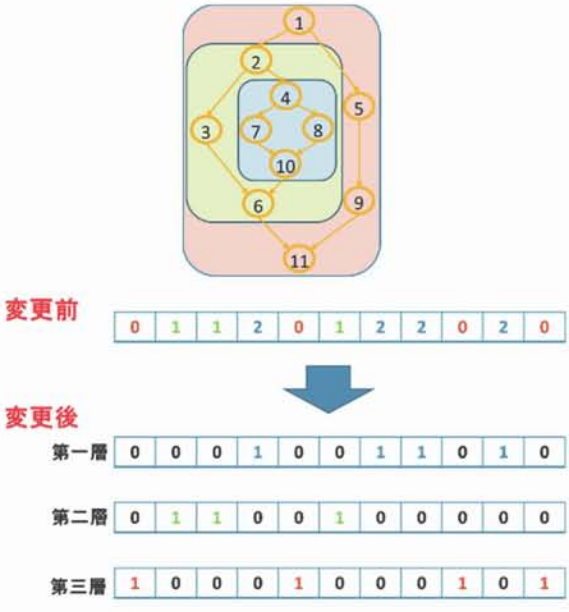


Fig. 6.3　修正後の出力データ形式

7．学習精度と評価

Fig. 7.1 はタスク数 20 のタスクグラフ 50,000 個分を 1 つのデータセットとし，それを学習した変更前と変更後の学習精度の遷移の比較を表したものである．階層的な部分タスクグラフの検出のために学習する変更後の方が変更前の学習に比べて学習精度が高水準で安定するまでにかかるステップ数が少ないことがわかる．

Fig. 7.1 変更の前後での学習精度の遷移の比較

また，学習が完了したものに入力データを与え，その部分タスクグラフ検出の精度を確認した．Fig. 7.2 は学習が完了したものに部分タスクグラフが 3 つ階層的に存在するタスクグラフを入力データとして与え，その部分タスクグラフ検出の結果を確認した例である．

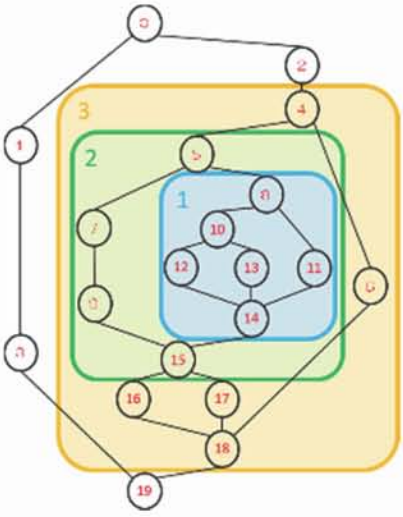


Fig. 7.2　部分タスクグラフ検出の例

8．おわりに

部分タスクグラフのDeep Learningを用いた検出においても，複数の層に分け，より小さな部分タスクグラフから検出し，検出済みのものは 1 つのマクロタスクにまとめてさらに上位層の部分タスクグラフを検出することが効果的であることが分かった．

今回，Deep Learningを適用するにあたって，その学習に用いる教師データは 1 つのタスクグラフから複数の入出力データを作成し，50,000 個分を 1 つのデータセットとしている．そのため，変更前と比べて 1 つのデータセットに含まれるタスクグラフの種類は減少している．従って，学習させるタスクグラフの種類を増やし，より大量のデータセットでの学習を試みることは今後の課題のひとつとして考えられる．また，現状の仕様では部分タスクグラフの検出にとどまっているため，実際のタスクグラフから部分タスクグラフをマクロタスク化し，スケジューリングを行う手段は別途進めている階層的スケジューリングと連携させてこの効果を確認する必要がある．また，タスクスケジューリングのGUIツールと連携し，検出された部分タスクグラフをGUIツール上で視覚化して確認できるようにすると便利になると考えられる．

参考文献

[1] 長谷川 幹，澁谷 知則，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のための階層的最適化による高速解法”，FIT2016(第 15 回情報科学技術フォーラム)，第 1 分冊，pp.191-196，2016