成蹊大学理工学研究報告
J. Fac. Sci.Tech., Seikei Univ.
Vol.58 No.1 (2021)pp.11-16

分枝限定法と Deep Learning を組み合わせた 並列タスクスケジューリング解法の開発

小納 惇平[*1]，甲斐 宗徳[*2]

Development of A Parallel Task Scheduling Solver that Combines A Branch-and-bound Method And Deep Learning

Jumpei KONO[*1], Munenori KAI[*2]

ABSTRACT：Since the task scheduling problem belongs to the strong NP-hard combinatorial optimization problem, the search time for the optimum solution becomes enormous due to the increase in the scale of the problem. Deep Learning can be applied to this difficult problem. Deep Learning has the advantage that the required time to find a solution is short once learning is completed, but it has the disadvantage that the optimum solution is not always found. Therefore, in this paper, we prototype and evaluate a method for speeding up to find the optimal solution by scheduling that combines the search method based on branch-and-bound method and deep learning.

Keywords：Task Scheduling, Deep Learning, Branch and bound, Convolutional Neural Network

(Received June 18, 2021)

1．はじめに

並列処理環境において、処理を効率的に高速化する手法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、並列環境下において、タスクという処理単位を、計算資源(Processor Element：PE)に最適に割り当て、処理時間を最小化する技術である。しかし、このタスクスケジューリング問題は強NP困難な組み合わせ最適化問題に属する[1]。そのため、問題の規模が大きくなるほど最適解の探索に膨大な時間を要してしまい、現実的な時間で最適解を得られない可能性がある。

タスクスケジューリング問題を解くためのアルゴリズムは大きく 2 つに分類される。1 つ目は、人間の経験則に基づき短時間で近似解を求めるヒューリスティックスケジューリングアルゴリズムである。2 つ目は、最適解が見つかるまで探索を行う最適化スケジューリングアルゴリズムである。代表的なヒューリスティックスケジューリングアルゴリズムにはリストスケジューリングなど[2][3][4]が挙げられる。代表的な最適化アルゴリズムとしてはDF/IHS法[4]が挙げられる。

DF/IHS法は、分枝限定法（Branch & Bound、以下B&Bと記す）に基づく方式で、各タスクのプライオリティレベル（そのタスクからタスクグラフの出口タスクまでの経路長の下限値）を用いて限定操作を行いながら探索を進めていく。このとき、探索における暫定解の値がよい値であればあるほど、早い段階で限定操作を行うことができるため、探索回数を削減することができる。筆者等の研究では、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手法を加えたアルゴリズムを用いており、さらに計算複雑度が上がることになる。

このようにアルゴリズムの構築が難しく、求解時間が膨大になる問題に対して、Deep Learningの適用が考えられる。Deep Learningの長所として、解を求めるまでにかかる時間が短い点があげられる。短所としては、Deep Learningは最適解を出力するとは限らないという点があげられる。Deep Learningによって短時間で解を出力して探索解法に伝え、暫定解の更新が行われれば限定操作を早い段階で行うことができ、探索時間の短縮につながる

[*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)

と考えられる。そこで本研究ではB&BとDeep Learningを組み合わせる並列タスクスケジューリングの研究を行い、探索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

2．1　タスクスケジューリングとは

タスクスケジューリングとは並列処理環境下においてプロセッサにタスクを最適に割り当て、処理性能を最大限に引き上げる技術である。タスクとは処理を行う対象を分割した単位を表している。各タスクは独立して処理を行うことはできるが、タスク間にはデータの依存関係が存在し、実行順序の制約(先行制約)を所持している。本研究では依存関係のあるタスク同士が異なるPEへ割り当てられた際、先行タスクのデータを後続タスクへ送る転送時間として通信時間を考慮する。

2．2　タスクグラフ

タスクフラフとは、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DAG(Directed Acyclic Graph)で表現することができる。処理の開始と終了を明確にするために、スタートノードとエンドノードと呼ばれる処理コスト 0 のダミーノードを追加する。図 2.1 はスタートノードをS、エンドノードをEとしたサンプルタスクグラフである。

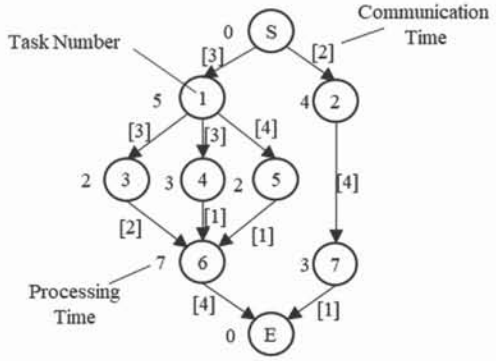

図 2.1　サンプルタスクグラフ

ノード内の数字はタスク番号(Task Number)、ノードの左側の数字は処理コスト(Processing Time)、各エッジに添えられた角括弧で囲まれた数字はタスク間の通信コスト(Communication Time)を表している。タスク間の通信コストは、並列処理において異なるPEに割り当てられた際に生じる、データの通信時間のことを表している。

2．3　ガントチャート

タスクグラフに対してスケジューリングを行った際、各PEにどのようにタスクを割り当てたかを可視化するために、ガントチャートを用いる。これは縦軸にPE、横軸に処理時間を取った時系列チャートである。本研究ではそれに加え、縦軸に各PEのデータ送受信のタイミングを表すSendとRecvの 2 つを追加している。図 2.2 のガントチャートは、図 2.1 のサンプルグラフにおけるスタートノードS、タスク 1、タスク 2 を割り当てた様子を表している。

各タスクは、自身のすべ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。タスクiと、エッジでつながれたその先行タスクjが異なるPEに割り当てられたならば、エッジに付与された通信コストC(i,j)を算入する必要がある。

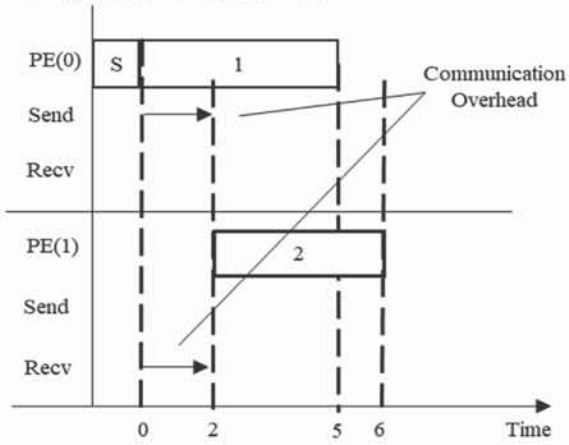

図 2.2　ガントチャート

3．Deep Learning

Deep Learningは十分なデータを用いることで、人間の力無しで機械が自動的に特徴を抽出する、ディープニューラルネットワーク(DNN)を用いた学習のことである。ディープニューラルネットワークとは、ニューラルネットワークと呼ばれる、人間や動物の脳神経回路をモデルとしたアルゴリズムを多層構造化したものである。

Deep Learningは大量のデータに対して学習を行い、その学習をもとに推論を行う。Deep Learningを用いる利点は主に 2 つある。1 つ目は一度学習を行えば推論にかかる時間が短く済む点である。2 つ目は学習から推論までDeep Learning自身が行うため、タスクスケジューリング問題のアルゴリズムを考える必要がない点である。一方、Deep Learningを用いるときの問題点は、Deep Learningの出力はあくまで推論であるため、与えられた問題に対して最適解を返すとは限らない点である。そのため最適解を求める場合には、最終的に別の手段で確かめる必要がある。

4．タスクスケジューリングへのDeep Learningの導入

4．1　タスクスケジューリングにおけるDeep Learningの役割

本研究でのタスクスケジューリングにおけるDeep Learningの役割は、B&Bに基づく探索手法の補助をすることである。

Deep Learningを用いたスケジューリングは、タスクグラフを入力した際にすべてのスケジューリング結果を一度で出力するのではなく、アイドルになったプロセッサにタスクを割り当てるイベントごとに、どのタスクを割り当てるべきかを出力させるようにしている。イベント時間は、いずれかのタスクを実行しているプロセッサの終了時間を比較し、最も早く終了するプロセッサの処理終了時間とした。このタスクの選択をスタートタスクからエンドタスクまで繰り返し行うことによりスケジューリング結果を得ることができる。

4．2　タスクスケジューリングへのDeep Learningの導入方法

タスクスケジューリングへのDeep Learningの導入の流れを以下に示す。

① ランダムなタスクフラフを大量に生成し、生成したタスクグラフの最適解をB&Bに基づく探索手法を用いて求める。
② ①で得られた最適解のスケジューリング結果をもとに教師データを作成する。
③ ②で作成した教師データを用いてDeep Learningが学習を行う。
④ 学習が完了したら、新しく生成されたタスクグラフに対してスケジューリングを行う
⑤ B＆Bの初期解を④で得られたスケジューリング結果で更新し、探索を開始する。

4．3　Deep Learningの教師データ

本研究におけるDeep Learningの教師データのうち、入力データはタスクグラフの情報とスケジューリング情報、出力データは割り当てるべきタスクの情報とした。

4．3．1　タスクグラフ情報の行列化

本研究におけるタスクグラフの情報はタスク数×タスク数の行列で表現した。行を先行タスク、列を後続タスクとする行列で表現する。行列の各要素には、先行制約に基づく通信情報を入れる。本研究における通信情報はタスク間の通信コストとする。また、行列の対角要素は各タスクにおけるタスクの処理コストとしている。これを基に図 4.1 のようなグラフは図 4.2 のような行列で表現できる。

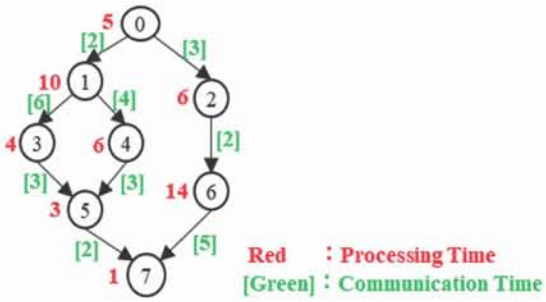

図 4.1　行列化前のタスクグラフ

Successor task

Predecessor task	T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]
T[0]	5	2	3	0	0	0	0	0
T[1]	0	10	0	6	4	0	0	0
T[2]	0	0	6	0	0	0	2	0
T[3]	0	0	0	4	0	3	0	0
T[4]	0	0	0	0	6	3	0	0
T[5]	0	0	0	0	0	3	0	2
T[6]	0	0	0	0	0	0	14	5
T[7]	0	0	0	0	0	0	0	1

図 4.2　行列化後のタスクグラフ

タスクiを先行タスク、タスクjを後続タスクとする通信における通信コストは､行列における[i,j]に入力される。例えば、タスク 0 を先行タスク、タスク 1 を後続タスクとする通信における通信コスト 2 は、行列における[0,1]に入力されている。また、タスクiの処理コストは行列の[i,i]に入力される。例えばタスク 0 の処理コスト 5 は行列の[0,0]に入力され､タスク 1 の処理コスト 10 は[1,1]に入力されている。また、先行制約がないところには 0 を入力している。

このようにすべてのタスクの処理コストとタスク間の通信コストを入力することにより、タスクグラフを行列で表現している。

4．3．2　スケジューリング情報の行列化

タスクグラフの情報は 1 つのタスクグラフに対して 1 つ生成される。一方、スケジューリング情報は、イベント時間ごとにタスクの割り当てが変わるため、1 つのタスクグラフに対して複数生成される。

スケジューリング情報には最初は何も割り当てられてい

ないという情報が入っている。その後、タスクの割り当てが行われるたびにスケジューリング情報の更新を行っていく。スケジューリング情報の更新の流れを図 4.3 に示す。

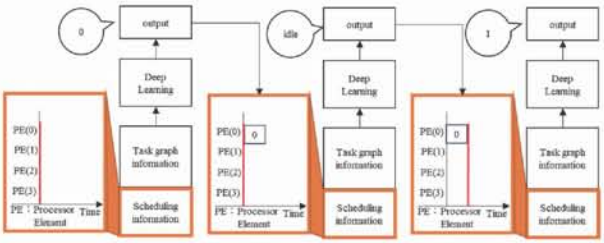

図 4.3　スケジューリング情報更新の流れ

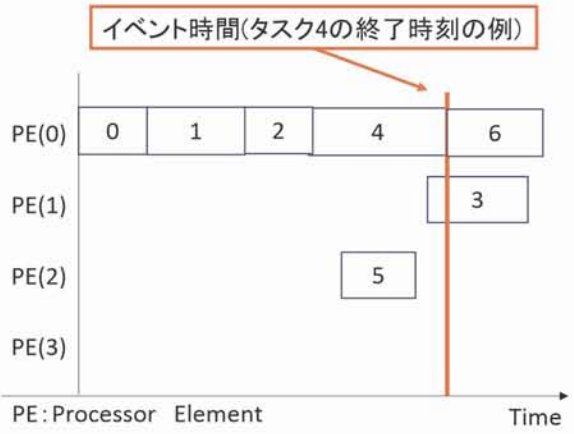

図 4.4　ガントチャートとイベント時間

図 4.3 のスケジューリング情報のガントチャートにおける赤線部分はイベント時間を表している。図 4.4 でガントチャートを示し、その時のスケジューリング情報を図 4.5 に示す。

	T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]
PE[0]	**1**	**2**	**3**	0	**4**	0	**-7**	**0**
PE[1]	0	0	0	**-6**	0	0	0	**0**
PE[2]	0	0	0	0	0	**5**	0	**1**
PE[3]	0	0	0	0	0	0	0	**0**

図 4.5　図 4.4 のガントチャートの赤線時点のスケジューリング情報

スケジューリング情報は行をプロセッサ数、列をタスク数とするプロセッサ数×タスク数の行列で表現した。各行と列はそれぞれのプロセッサとタスクとみなす。行列の各要素には、そのイベント時間での各プロセッサにおける各タスクの割り当て状況が入力されている。本研究では、割り当てられていなければ 0、割り当てられていればその割り当て順序が入っている。また、各要素は実行済みであれば正の数、実行中であれば負の数とする。

図 4.4 においてタスク 0 はPE0 で最初に割り当てられているため、行列の[0,1]に 1 を入力している。また、タスク 0 はPE1 では処理を行われていないため 0 が入力されている。また、エンドタスクに該当する最後の列の要素はどのプロセッサの割り当てを行うかの情報を入力する。具体的には割り当てを行うプロセッサは 1、それ以外は 0 とする。

4. 3. 3　割り当てるべきタスク情報のベクトル化

出力データである、割り当てるべきタスク情報はタスク数にアイドルタスクを加えた、タスク数＋1 の配列とした。配列の各要素には、それぞれのタスクを割り当てるべき確率が入っている。教師データとして作成する際は、割り当てるタスクを 1 それ以外は 0 で表している。例えば、タスク 6 を割り当てる際には図 4.6 のような配列になる。

T[0]	T[1]	T[2]	T[3]	T[4]	T[5]	T[6]	T[7]	T[idle]
0	0	0	0	0	0	1	0	0

図 4.6　タスク 6 を割り当てる際の割り当て得るべきタスク情報

4. 3. 4　畳み込みニューラルネットワーク(CNN)の適用

前節で述べたように入力データが行列で表現できるため、行列形式データを学習する画像認識の分野で効果を発揮している畳み込みニューラルネットワーク(CNN)を適用することにした。4 つの畳み込み層に入力を行い結合していく流れを行い(図 4.7)、それを横に 3 つ縦に 3 つつなげるというモデルを採用した。モデルを図 4.8 に示す。

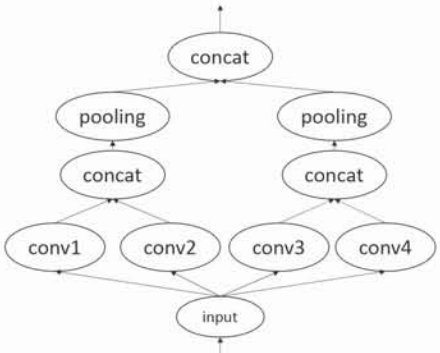

図 4.7　conv_4 レイヤー

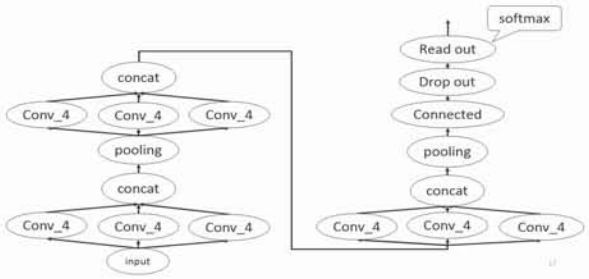

図 4.8　Deep Learningのモデル

5．並列スケジューリング解法

タスク数の多いタスクグラフに対してスケジューリングを行うためにB&Bによる全探索解法とDeep Learningによる解法を同時に連携させて進める並列スケジューリング解法を用いる。並列スケジューリング解法の流れを以下に示す。

① 最適解の求まるサイズのタスクグラフでB&Bによる探索を行って教師データを作成し、それをDeep Learningに学習させる
② 探索したいタスクグラフに対してDeep Learningでスケジューリングを行い、その結果を初期解としてB&Bによる探索を開始する。
③ B&Bによる探索において暫定解の更新が行われた場合、その暫定解を教師データに加え学習を行わせる。
④ 再びDeep Learningにスケジューリングを行わせ、暫定解よりもいい結果を出力したらB&Bの暫定解の更新を行い、探索を進行させる。
⑤ ③と④を繰り返し行う

②において、新たに生成されたタスクグラフに対してDeep Learningを用いたスケジューリングの流れを以下に示す。

A) 新たに作成したタスクグラフからタスクグラフの情報とスケジューリング情報を作成する。
B) イベント時間において、スケジューリング情報の更新を行う。
C) A)で作成した情報をDeep Learningに入力し、実行するべきタスクを出力させる。
D) 割り当てを行ったタスクがアイドルタスクでない場合、プロセッサの処理終了時刻を割り当てたタスクの処理終了時間に更新する。
E) 未割当のタスクが存在するならB)に戻る。
F) すべてのタスクの割り当てが完了したら、最後に実行が完了するタスクの処理終了時間をDeep Learningを用いて求めたスケジュール長とする。

④において、Deep Learningの解がB&Bの暫定解の更新を行った場合、B&Bの探索において限定操作が早い段階で行われるようになるため、探索時間を短縮できると考えられる。

6．評価

今回の実験は以下のようなタスクグラフで実験を行った。

- タスク数：20
- 各ノードが持つ最大の直接先行タスク数：3
- 各ノードが持つ最大の直接後続タスク数：3
- 各ノードの最大処理コスト：30
- 各ノードの最小処理コスト：5
- 各エッジの最大通信遅延コスト：10
- 各エッジの最小通信遅延コスト：5
- 作成するタスクグラフが持つ最大並列度：4
- 作成するタスクグラフが持つ最小並列度：1

また、割り当てるプロセッサ数は 4 で固定して実験を行った。処理コストや通信時間、先行後続関係がランダムなタスクグラフを 682 個作成し、最適解を探索し合計で 35265 個の教師データを作成した。そのデータをもとにDeep Learningに学習を行わせたところ学習精度は約71%となった。

新しく 100 個のタスクグラフを作成し、Deep Learningを用いたスケジューリングを行った。スケジューリング結果と探索の初期解を比較したグラフを図 6.1 に示す。

図 6.1　スケジュール長の比較

今回の実験では、スケジューリングを行った 100 タスクグラフのうち 6 タスクグラフにおいて、従来手法のB&Bにおける探索の初期解よりDeep Learningを用いたスケジューリングの解がいい解を出力した。しかし、ほとんどのタスクにおいて、従来手法のB&Bにおける探索の初期解の方が良い解を出力するという結果になった。

次に、Deep Learningを用いたスケジューリングの解が従来手法のB&Bにおける探索の初期解よりもいい解を出力したタスクグラフのうち、最も探索時間が長かった

タスクグラフ 59 に対して並列スケジューリングを行った。B&Bの探索における解の更新回数とその時点での探索の解を図 6.2 に示す。

分枝限定法の探索		
探索時間[h]	解の更新回数	探索の解
-	-	(188)
0〜1	5	165
2〜3	3	160
3〜4	2	158
16〜17	2	156

図 6.2　分枝限定法の暫定解の更新

解の更新が行われ、新しく教師データが得られた際にDeep Learningで学習を行い、再びスケジューリング解を出力させた。その時の解とB&Bの探索の解を比較し、より良い値で更新した際の暫定解を図 6.3 に示す。

探索開始からの経過時間[h]	Deep Learningを用いたスケジューリングの解	暫定解
0	173	173
1	187	165
3	208	160
4	166	158
17	170	156

図 6.3　Deep Learningによるスケジューリング解

このタスクグラフは 100 時間探索したところ、探索が終了しなかった。100 時間の探索のうち暫定解の更新が行われたのはほとんどが探索開始から 4 時間以内であった。また、Deep Learningを用いたスケジューリングの解が暫定解の更新に使われたのは初期解のときのみであった。

7．おわりに

いくつかのタスクグラフでB&Bの初期解より良い解を出力することができた。

並列スケジューリングにおいてDeep Learningを用いたスケジューリングによる解が暫定解を更新するまでに至らなかったが、最初に求めた解よりも良い解を導出できた点に関してはよかったと言える。

タスク選択の際に同じスケジュール長になる場合でもいずれかのPEを選んだ時に不正解になるのが学習精度を落とす原因の一つになっている可能性はある。

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] 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