

MDDの拡張による線形順序拡大集合の省領域表現とスケジューリング問題への応用

三宅 郁人¹ 瀧本 英二¹ 畑埜 晃平^{1,2}

概要：順序制約付きスケジューリング問題に対し、与えられた順序制約を満たす全順序からなる集合（線形順序拡大集合）を圧縮表現した多分岐決定図（Multi-valued Decision Diagram, MDD）を用いて、そのMDDのサイズに比例した時間で最適解を求める手法が提案されている。しかし、この手法には、順序制約の数が少ないほどMDDのサイズが指数爆発を起こしやすいという欠点がある。本研究では、線形順序拡大集合上に同値関係を導入し、その同値関係により定義される同値類集合を表すMDD表現法を提案するとともに、このMDDを用いても、そのサイズに比例した時間で最適解を求められることを示す。提案手法で得られるMDDのサイズは、従来手法で得られるMDDのサイズを超えることはなく、指数的に改善される場合がある。人工データを用いた実験において、提案手法は、順序制約数がある一定値以下になると、従来手法とは逆に、制約数が少ないほど計算量が急激に改善する性質を持つことが示された。

キーワード：組合せ最適化、スケジューリング問題、MDD

1. 概要

順序制約付きスケジューリング問題とは、各ジョブの処理時間、および、一部のジョブ対における順序制約が与えられたとき、実行可能解集合の中で各ジョブの待ち時間と実行時間の合計（フロータイム）を最小にするジョブの実行順序（スケジューリング）を求める問題である。ここで、順序制約はジョブ集合上の半順序関係とみなすことができ、実行可能解とは、与えられた半順序関係に矛盾しない全順序、すなわち線形順序拡大（linear extension）のことをいう。順序制約付きスケジューリング問題は一般にNP困難 [4, 5] で、2 近似多項式時間アルゴリズムが知られている [1–3, 6, 8]。一方、厳密解法においても多分岐決定図（Multi-valued Decision Diagram, MDD）と呼ばれる有向非巡回グラフを用いた手法が提案されている [7]。それは、与えられた半順序関係から、その線形順序拡大集合を表現するMDDを効率的に構築し、さらに、スケジューリング問題をMDD上の最短路問題に帰着させ線形時間で解くというものである。従って、計算量は構築されたMDDのサイズに依存することになるが、順序制約の数が少ないほど、MDDのサイズが指数爆発を起こしやすいという欠点があ

った。

本研究では、線形順序拡大集合上に同値関係を導入し、その同値関係によって定義される同値類の代表元のみからなる集合をMDDで表現することにより、線形順序拡大集合全体を、より省領域で表現する手法を提案する。このMDDは、与えられた制約に、いくつかの制約を追加することによって得られる半順序集合の線形順序拡大集合を表現するものとなっているため、その構築に [7] の手法を用いることができる。また、代表元集合を表すMDDのサイズは、線形順序拡大集合全体を表すMDDのサイズを超えることはなく、指数的に改善される場合が存在する。さらに、新しいMDD表現を用いても、スケジューリング問題は、そのMDD上の最短路問題に帰着可能であり、したがって効率的に最適解を求めることができることを示す。

実際、人工データによる実験において、提案手法が既存手法に劣ることはなく、順序制約が少ない場合は既存手法を凌駕する結果を得ている。また、提案手法は既存手法とは逆に、順序制約数がある一定以下になると、順序制約が少ないほどより計算量が急激に改善する性質を持つ。

2. 準備

ジョブの集合を $[n] := \{1, 2, \dots, n\}$ で表すとし、 S_n を $[n]$ 上の順列ベクトルの集合とする。例えば、 $S_3 = \{(1, 2, 3), (1, 3, 2), (2, 1, 3), (2, 3, 1), (3, 1, 2), (3, 2, 1)\}$ 。順

¹ 九州大学
Kyushu University

² 理研 AIP
RIKEN AIP

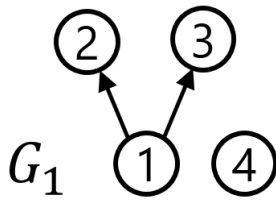


図 1 制約グラフの例. 制約 G_1 を満たす線形順序拡大の逆順序の集合は, $S_{G_1}^{-1} = \{(2, 3, 1, 4), (2, 3, 4, 1), (2, 4, 3, 1), (3, 2, 1, 4), (3, 2, 4, 1), (3, 4, 2, 1), (4, 2, 3, 1), (4, 3, 2, 1)\}$.

列 $\pi \in S_n$ に対し, π_i は π の第 i 成分を指す. $\pi_i = j$ のとき $\pi_j^{-1} = i$ と記し, 順列 $\pi \in S_n$ に対する逆順列 $\pi^{-1} \in S_n$ を, $\pi^{-1} = (\pi_1^{-1}, \dots, \pi_n^{-1})$ と定義する. 順列 π の各成分 π_i をジョブ i の優先度と解釈することにより, π は, π_i の大きい順にジョブ i を処理せよというジョブの実行順序を表す. すなわち, 順列 π が指定するジョブの実行順序は, $\pi_n^{-1} \rightarrow \pi_{n-1}^{-1} \rightarrow \dots \rightarrow \pi_1^{-1}$ となる.

各ジョブ $i \in [n]$ の処理時間を w_i とすると, 実行順序 π のフロータイムは, $\sum_{i=1}^n \pi_i w_i = \pi \cdot w$ で表される. 例えば, $n = 4$ で $1 \rightarrow 4 \rightarrow 2 \rightarrow 3$ の順にジョブが実行されるとする. この実行順序を表す順列は $\pi = (4, 2, 1, 3)$ である. 一方, ジョブ 1, 4, 2, 3 の待ち時間と処理時間の和は, それぞれ $w_1, w_1 + w_4, w_1 + w_4 + w_2, w_1 + w_4 + w_2 + w_3$ となるが, それらの合計 (フロータイム) は, 確かに $4w_1 + 3w_4 + 2w_2 + w_3 = (4, 2, 1, 3) \cdot (w_1, w_2, w_3, w_4) = \pi \cdot w$ となっている.

順序制約は, $[n]$ 上の半順序関係を表す有向非循環グラフ (DAG) $G = ([n], E)$ として与えられるとする. ただし, 有向辺 $(i, j) \in E$ は, 「ジョブ i をジョブ j より先に処理しなければならない」という制約を表す. したがって, 順序制約 G をみたす順列ベクトル (G の線形順序拡大) の集合 S_G および, その逆順列の集合 S_G^{-1} は, それぞれ以下のよう

$$S_G = \{\pi \in S_n \mid \forall (i, j) \in E, \pi_i > \pi_j\},$$

$$S_G^{-1} = \{\pi^{-1} \mid \pi \in S_G\}.$$

図 1 に例を示す.

以上の準備のもとに, 順序制約付きスケジューリング問題を次のように定義する.

入力: DAG $G = ([n], E)$, $w \in \mathbb{R}^n$

出力: $\pi = \operatorname{argmin}_{\pi \in S_G} \pi \cdot w$

3. 既存手法

Matsumoto らは, 順序制約 G の線形順序拡大集合 S_G を表現するデータ構造 π -MDD を用いて, 順序制約付きスケジューリング問題の厳密解を求める手法を提案した [7]. 本節では, その概要を述べる.

π -MDD は, 各順列 $\pi^{-1} \in S_G^{-1}$ の各成分を連結することによって得られる長さ n の文字列集合を受理する状態数

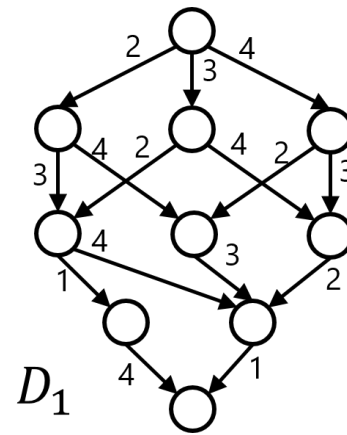


図 2 図 1 の順序制約 G_1 を満たす線形順序拡大集合 S_G を表現する π -MDD D_1 .

最小の決定性有限オートマトンとして定義される. (ただし, 非受理状態と非受理状態への遷移は省略.) 図 2 に例を示す. π -MDD は, 1 つの根 (開始状態) と 1 つの葉 (受理状態) を持ち, 各辺 e がジョブ $l(e) \in [n]$ をラベルとして持つ DAG であること, および, 根から葉までの各経路 $(e_1, e_2, \dots, e_n)$ と S_G^{-1} の各要素 $(l(e_1), l(e_2), \dots, l(e_n))$ が 1 対 1 対応することに注意されたい. π -MDD のサイズを, D を DAG とみなしたときの辺の数と定義し, $|D|$ と記す.

Matsumoto らの手法は, 以下の 2 つの手順からなる.

手順 1 与えられた制約グラフ $G = ([n], E)$ から, 線形順序拡大集合 S_G^{-1} を表す π -MDD $D = (V_D, E_D)$ を構築する.

手順 2 π -MDD D の各辺 (状態遷移) $e \in E_D$ に, 重み $v_e = dw_{l(e)}$ を割り当てる. ここで, d は辺 e の深さである. 次に, 得られた辺重みつきグラフの根から葉までの最短路 $(e_1, e_2, \dots, e_n)$ を求め, その最短路が表す受理系列 $(l(e_1), l(e_2), \dots, l(e_n)) = (\pi^*)^{-1} \in S_G^{-1}$ から, 最適解として π^* を出力する.

アルゴリズムの正当性は, 以下の観察により容易に確かめられる. 手順 2 の重みづけにより, 根から葉までの任意の経路 $(e_1, e_2, \dots, e_n)$ に対し, その経路に対応する順列を $\pi^{-1} = (l(e_1), l(e_2), \dots, l(e_n))$ とすると, その経路の長さ $\sum_{d=1}^n v_{e_d} = \sum_{d=1}^n dw_{l(e_d)} = \sum_{d=1}^n dw_{\pi_d^{-1}} = \sum_{i=1}^n \pi_i w_i$ が π のフロータイムを与える.

Matsumoto らの手法の性能を定理として示す前に, いくつかの準備を行う.

順序制約 G が定義する半順序を $<_G$ で表す. すなわち, 任意の $u, v \in [n]$ に対し, $v \leq_G u$ と, G において頂点 u と v を結ぶ道が存在することが等価である. 集合 $L \subseteq [n]$ が順序制約 G の下方集合 (lower set) であるとは, 任意の $u, v \in [n]$ に対し, $u \in L$ かつ $v <_G u$ ならば $v \in L$ を満たすときをいう. G の下方集合の族を $\mathcal{L}_G \subseteq 2^{[n]}$ と記す. 下

方集合族と線形順序拡大の関係について、束論においてよく知られた以下の命題が成り立つ。

命題 3.1. 半順序集合 $(\mathcal{L}_G, \supseteq)$ は、最小元 $[n]$ と最大元 $\emptyset$ を持つ分配束であり、各極大鎖 $L_1 = [n] \supseteq L_2 \supseteq \dots \supseteq L_{n+1} = \emptyset$ と G の線形順序拡大 $\pi = (\pi_1, \pi_2, \dots, \pi_n)$ が一対一対応し、任意の $1 \leq i \leq n$ に対し、 $L_i = L_{i+1} \cup \{\pi_i^{-1}\}$ が成り立つ。

この命題より、以下の系を得る。

系 3.1. 制約グラフ G の線形順序拡大集合 S_G を表す π -MDD は、半順序 $(\mathcal{L}_G, \supseteq)$ のハッセ図と同型である。

すなわち、 π -MDD の各状態は下方集合 $L \in \mathcal{L}_G$ と一対一対応し（初期状態は $[n]$ 、受理状態は $\emptyset$ に対応）、任意の $L, L' \in \mathcal{L}$ に対し、 $L = L' \cup \{u\}$ のとき、状態 L から L' に向かってラベル u の辺がある。

いよいよ、Matsumoto らの手法の性能を以下に示す。

定理 3.1 ([7]). 与えられた制約グラフ $G = ([n], E)$ に対し、制約付きスケジューリング問題の厳密解は、 $O(n|\mathcal{L}_G|)$ 時間で求めることができる。

4. 提案手法

本節では、本研究における提案手法について述べる。

Matsumoto らの手法では、制約が少ないほど G の下方集合 $\mathcal{L}_G$ の数が大きくなる傾向にあるため、定理 3.1 で示唆されるように、求解に必要な時間が大きくなるという問題がある。特に、制約がまったくない場合が最悪で、下方集合は最大 ($|\mathcal{L}_G| = 2^{[n]}$) となる。一方、制約がまったくない場合は、処理時間 w_i の小さい順にジョブを実行するのが最適であるため、ソートにより最適解を簡単に求めることができる。したがって、 S_G において制約がないとみなせるジョブの部分集合 A を見出すことができれば、 A 上の最適順序はソートによって求められるので、 A 上の $|A|!$ 通りの順序を 1 つの順序に集約した順序集合を現す π -MDD を構築すれば十分であることが分かる。本手法では、このようにして、 π -MDD のサイズの削減を図る。

4.1 同値関係

まず準備として、以下を定義する。

定義 4.1 (入力集合, 出力集合). 制約グラフ $G = ([n], E)$ における各頂点 $v \in [n]$ に対して、 v の入力集合 I_v と出力集合 O_v を以下のように定義する。

$$I_v = \{v' \in [n] \mid (v', v) \in E\},$$

$$O_v = \{v' \in [n] \mid (v, v') \in E\}.$$

これを用いて、ジョブ集合 $[n]$ 上の同値関係を定義する。

定義 4.2 (ジョブ集合上の同値関係). 制約グラフ $G = ([n], E)$ において、ジョブ $i, j \in V$ が同値であるとは、 $I_i = I_j$ かつ $O_i = O_j$ が成り立つことをいい、 $i \simeq_G j$ と書く。ただし、以降文脈から明らかなときは $i \simeq j$ と表記する。

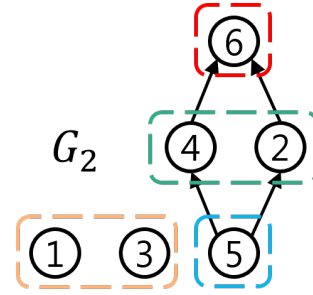


図 3 制約グラフ G_2 の同値類への分割（点線枠は同値類を表している。）

この同値関係の定義により、ジョブ集合 $[n]$ は同値類に分割される。例を図 3 に示す。以降では、制約グラフ G を固定し、 G 上の同値関係に基づく $[n]$ の同値類への分割を

$$[n] = A_1 \cup A_2 \cup \dots \cup A_N$$

とおき、各 A_j をジョブ同値類と呼ぶ。

また、順序ベクトル π 、ジョブの部分集合 $A \subseteq [n]$ 上の任意の置換 $\delta: A \rightarrow A$ に対し、 $\pi \circ \delta$ を、任意の $i \in [n]$ に対し、

$$(\pi \circ \delta)_i = \begin{cases} \pi_{\delta(i)} & i \in A \text{ のとき,} \\ \pi_i & i \notin A \text{ のとき} \end{cases}$$

を満たす順序と定義する。これを用いて G の線形順序拡大集合 S_G 上の同値関係を定義する。

定義 4.3 (線形順序拡大集合上の同値関係). 順序 $\pi, \pi' \in S_G$ が制約グラフ $G = (V, E)$ について同値であるとは、各ジョブ同値類 A_j に対し、 A_j 上の置換 $\delta_{A_j}: A_j \rightarrow A_j$ が存在して、

$$\pi' = \pi \circ \delta_{A_1} \circ \delta_{A_2} \circ \dots \circ \delta_{A_N}$$

が成り立つことをいう。順序 π と π' が制約グラフ G について同値であるとき、 $\pi \simeq_G \pi'$ と書く。ただし、以降文脈から明らかなときは $\pi \simeq \pi'$ と表記する。

順序の同値関係の例を図 4 に示す。

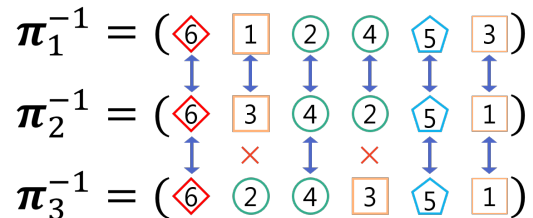


図 4 制約グラフ G_2 について π_1 と π_2 は同値だが π_2 と π_3 は同値でない。（矢印はノード同士が同値であることを表している。）

また、 S_G 上の同値関係について、以下の補題が成り立つ。

補題 4.1. 任意の順序 $\pi, \pi' \in S_n$ に対し、 $\pi \in S_G$ かつ $\pi' \simeq \pi$ ならば、 $\pi' \in S_G$ 。

証明. $a, b \in [n]$ に対し、ジョブ集合 $[n]$ 上の互換 $\tau^{(a,b)}: [n] \rightarrow [n]$ を以下で定義する。

$$\tau_i^{(a,b)} = \begin{cases} b & i = a \text{ のとき,} \\ a & i = b \text{ のとき,} \\ i & \text{それ以外.} \end{cases}$$

一般に, $[n]$ 上の置換はいくつかの $[n]$ 上の互換の合成として表せるので, 次の命題が成立することを示せばよい.

[命題] 任意の $\pi \in S_G$, 任意の $a, b \in [n]$ に対し, $a \simeq b$ ならば, $\pi \circ \tau^{(a,b)} \in S_G$.

以降では, この命題を背理法により証明する. すなわち, $\pi \in S_G$, $a \simeq b$ なのに $\pi' = \pi \circ \tau^{(a,b)} \notin S_G$ と仮定して矛盾を導く.

$\pi' \notin S_G$ より, ある $k, l \in [n]$ が存在して,

$$\pi'_k < \pi'_l \wedge (k, l) \in E$$

が成り立つ. 一方, $\pi \in S_G$ より, $\pi_k > \pi_l$. すなわち, π と π' で k, l 番目の値の大小関係が逆転していることから, $I \equiv \{a, b\} \cap \{k, l\} = \emptyset$ はあり得ない. 以下, I に関する場合分けを行う.

(i) $I = \{k, l\}$ のとき.

$(k, l) \in E$ なので, $(a, b) \in E$ または $(b, a) \in E$ のどちらかが成り立つ. しかし, これは $a \simeq b$ に反している. なぜなら, ジョブ集合上の同値関係の定義より, $a \simeq b$ が成り立つためには, a と b の間に辺があつてはならないからである.

(ii) $I = \{k\}$ のとき.

一般性を失わずに, $a = k$, $b \notin \{k, l\}$ と仮定する. $(a, l) = (k, l) \in E$ および $a \simeq b$ より, $(b, l) \in E$, つまり $\pi_b > \pi_l$. 従つて, $\pi'_k = \pi'_a = \pi_b > \pi_l = \pi'_l$ が成り立つが, これは $\pi'_k < \pi'_l$ に反する.

(iii) $I = \{l\}$ のとき.

一般性を失わずに, $a = l$, $b \notin \{k, l\}$ と仮定する. $(k, a) = (k, l) \in E$ および $a \simeq b$ より, $(k, b) \in E$, つまり $\pi_b > \pi_k$. 従つて, $\pi'_l = \pi'_a = \pi_b > \pi_k = \pi'_k$ が成り立つが, これは $\pi'_k < \pi'_l$ に反する.

以上より, すべての場合において矛盾が生じるので, 題意が示された. $\square$

S_G 上の同値関係に基づいて, 順列 $\pi \in S_G$ の同値類 $[\pi]$ の代表元 $\tilde{\pi}$ を, $\tilde{\pi} \simeq \pi$, かつ, 任意のジョブ同値類 A_j , および任意の $k, l \in A_j$ ($k < l$) に対し, $\tilde{\pi}_k > \tilde{\pi}_l$ を満たす順列と定義する. すなわち, 各ジョブ同値類を構成するジョブ集合に対し, ジョブ番号に基づく全順序制約を追加することによって得られる制約グラフを $\tilde{G}$ とすると, その線形順序拡大集合が代表元 $\tilde{\pi}$ の集合に一致する.

4.2 提案手法

提案手法では, 以下の手順で最適解を求める.

- (1) ジョブ同値類 $A_1, A_2, \dots, A_N$ を求める.
- (2) $S_{\tilde{G}} = \{\tilde{\pi} \mid \pi \in S_G\}$ を満たす制約グラフ $\tilde{G}$ を構築する.
- (3) Matsumoto らの手法を用いて, $S_{\tilde{G}}$ を表す π -MDD $\tilde{D}$ を構築する.
- (4) $\tilde{D}$ の各辺に適切に重みを割り当て, 最短経路問題を解くことで最適解を求める.

以下, 各手順について詳しく見ていく.

手順 1 では, G の各頂点 $v \in [n]$ に対し, まず, v の入出力集合 $I_v = \{i_1, i_2, \dots, i_{|I_v|}\}$ ($i_1 < i_2 < \dots < i_{|I_v|}$) と $O_v = \{o_1, o_2, \dots, o_{|O_v|}\}$ ($o_1 < o_2 < \dots < o_{|O_v|}$) を連結し, 文字列 $s_v = i_1 i_2 \dots i_{|I_v|} o_1 o_2 \dots o_{|O_v|}$ を作る. 次に, 文字列集合 $\{s_v \mid v \in [n]\}$ をソートする. このとき, 文字列長が短いほど順位が小さく, 文字列長が同じときは辞書式順序でソートする. すると, $v \simeq v' \Leftrightarrow s_v = s_{v'}$ であるため, 同値なジョブに対応する文字列は, ソートした系列上では連続的に現れる. これを検出することで, 同値類による分割を求める. 制約グラフ $G = ([n], E)$ の辺の数を $m = |E|$ とおくと, 手順 1 の計算量は $O(n + m)$ 時間である.

手順 2 では, 各 $j \in [N]$ に対し, ジョブ同値類 A_j の要素を $v_1 < v_2 < \dots < v_{|A_j|}$ とすると, 全ての $1 \leq i \leq |A_j| - 1$ に対して有向辺 (v_i, v_{i+1}) を追加することで, 制約グラフ $\tilde{G}$ を得る (図 5 参照). 手順 2 の計算量は $O(n)$ 時間である.

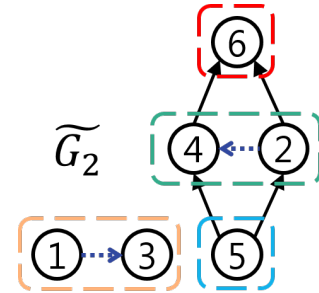


図 5 手順 2 にしたがって, 図 3 の制約グラフ G_2 に有向辺を追加して得られる制約グラフ $\tilde{G}_2$

手順 3 は, Matsumoto らの手法を用いることにより, $O(n|\mathcal{L}_{\tilde{G}}|)$ 時間で $S_{\tilde{G}}$ を表す π -MDD $\tilde{D}$ が得られる.

手順 4 では, まず, ジョブ同値類ごとに, 与えられた処理時間ベクトル w をソートする. そのために, n 次元の配列 $C[\cdot] = \{1, 2, \dots, n\}$ を用意し, 以下の性質を満たすように並べ替える (図 6 参照).

- (1) 任意の $i \in [n]$ について, $i \simeq C[i]$,
- (2) $i \simeq j$ を満たす任意の $i, j \in [n]$ に対し, $i < j \Rightarrow w_{C[i]} \leq$

i	1	2	3	4	5	6
$C[i]$	1	4	3	2	5	6

図 6 制約グラフ G_2 , 処理時間ベクトルが $w_1 \leq w_5 \leq w_3 \leq w_4 \leq w_2 \leq w_6$ であるときの配列 C

$w_{C[j]}$.

次に, π -MDD $\tilde{D}$ の各辺 $e \in E_{\tilde{D}}$ に, 重み $dw_{C[l(e)]}$ を割り当て, 根から葉までの最短路 $(e_1, \dots, e_n)$ を求める. ここで, d は辺 e の深さ, $l(e)$ は辺 e のラベルである. このとき, $\pi^{*-1} \equiv (C[l(e_1)], \dots, C[l(e_n)]) \in S_G^{-1}$ が成り立ち, 最適解として π^* を出力する. 手順 4 の計算量は, $O(n \log n + n|\mathcal{L}_{\tilde{G}}|)$ である.

以上のことを, 主定理としてまとめる.

定理 4.1. 提案手法は, 順序制約付きスケジューリング問題の厳密解を $O(n \log n + m + n|\mathcal{L}_{\tilde{G}}|)$ 時間で出力する.

以下, 提案手法の妥当性の証明を行う.

証明. まず, 手順 2 において得られる制約グラフ $\tilde{G}$ の線形順序拡大集合 $S_{\tilde{G}}$ が, $S_{\tilde{G}} = \{\tilde{\pi} \mid \pi \in S_G\}$ を満たすことを示す.

$S_{\tilde{G}} \subseteq \{\tilde{\pi} \mid \pi \in S_G\}$ の証明:

$\pi \in S_G$ を任意に選ぶ. $\tilde{G}$ の構成法より, $\pi \in S_G$ であることは明らか. また, π は各ジョブ同値類 A_j に対して追加された制約を満たすので, 任意のジョブ $k, l \in A_j$ に対して, $k < l$ ならば $\pi_k > \pi_l$ をみたす. これは, $\pi = \tilde{\pi}$ であることを意味する.

$S_{\tilde{G}} \supseteq \{\tilde{\pi} \mid \pi \in S_G\}$ の証明:

$\pi \in S_G$ を任意に選ぶ. 代表元の定義より, $[\pi]$ の代表元 $\tilde{\pi} \simeq \pi$ は, 元の制約 G だけでなく, 各ジョブ同値類 A_j に対して追加された制約を満たすので, 制約 $\tilde{G}$ も満たす. したがって, $\tilde{\pi} \in S_{\tilde{G}}$.

次に, 手順 4 の正当性を示す.

補題 4.1 より, 任意の順列 $\pi \in S_{\tilde{G}}$ は, 任意の同値類上の任意の置換を行ってもなお制約グラフ $\tilde{G}$ をみたす. ここで, 任意の同値類 $A = \{v_1, v_2, \dots, v_{|A|}\} \subseteq [n]$ に含まれる要素の並び順で最もフロータイムを小さくするのは, フロータイムが順列ベクトル π と処理時間のベクトル w の内積で表されることから, 明らかに任意の $i, j (1 \leq i, j \leq |A|, i \neq j)$ に対して $\pi_{v_i} > \pi_{v_j}$ のとき $w_{v_i} < w_{v_j}$ となるような並び順である. したがって, 手順 4 の前半では, 任意の同値類 $A \subseteq [n]$ に対し, ジョブ $i, j \in A (i < j)$ が $\pi_i > \pi_j$ をみたすようなジョブ番号の系列を, ジョブ $i, j \in A (i < j)$ が $w_i \leq w_j$ をみたすようなジョブ番号の系列へ変換する規則を配列 C として作成している. したがって, 配列 C による変換は, 代表元 $\tilde{\pi} \in S_{\tilde{G}}$ を引数とする, 以下の関数 $F: S_{\tilde{G}} \rightarrow S_G$ の役割を果たすとみなせる

$$F(\tilde{\pi}) = \operatorname{argmin}_{\pi \in [\tilde{\pi}]} \pi \cdot w.$$

よって, π -MDD $\tilde{D}$ の各辺ラベル $l(e)$ を $C[l(e)]$ に置換し

てできる π -MDD $\tilde{D}_C$ では, 任意のパスが, そのパスが表す順列が属する同値類の中で最もフロータイムを小さくする順列を表している.

手順 4 の後半は, π -MDD $\tilde{D}_C$ 上の最短経路問題を解いているとみなすことができ, 以下をみたす順列 π^* のインバースが求められる

$$\pi^* = \operatorname{argmin}_{\pi \in \{F(\tilde{\pi}) \mid \tilde{\pi} \in S_{\tilde{G}}\}} \pi \cdot w.$$

したがって, さらにそのインバースを求めれば最適解が得られる. $\square$

最後に, 提案手法によって得られる π -MDD のサイズは, 既存手法によって得られる π -MDD のサイズを超えないことを示す.

定理 4.2. G の線形順序拡大集合 S_G を表す π -MDD を D , $\tilde{G}$ の線形順序拡大集合 $S_{\tilde{G}}$ を表す π -MDD を $\tilde{D}$ とすると, $|\tilde{D}| \leq |D|$.

証明. $\tilde{G}$ は, G に制約を追加して得られた半順序を表すので, $\tilde{G}$ の任意の下方集合は, G の下方集合でもある. したがって, $\mathcal{L}_{\tilde{G}} \subseteq \mathcal{L}_G$ が成り立つ. 系 3.1 より, $\tilde{D}$ は半順序集合 $(\mathcal{L}_{\tilde{G}}, \supseteq)$ のハッセ図と同型であるが, これは明らかに, 半順序集合 $(\mathcal{L}_G, \supseteq)$ のドメインを $\mathcal{L}_{\tilde{G}}$ に制限したもので, $\tilde{D}$ は, D の状態の部分集合 $\mathcal{L}_{\tilde{G}}$ によって誘導される D の部分グラフとみなすことができる. $\square$

5. 実験

順序制約付きスケジューリング問題を, 既存手法と提案手法で解き, その性能を比較する.

5.1 設定

実験で用いる制約グラフ $G = ([n], E)$ として, Erdős-Rényi ランダムグラフ $G_{n,p}$ を用いた. すなわち, $G_{n,p}$ は, 各 2 頂点 $i, j \in [n] (i < j)$ に対し, 確率 $0 \leq p \leq 1$ で有向辺 (i, j) を引くことによって得られる. また, 処理時間ベクトル w は, $[0, 1]^n$ 上の一様分布に従って選んだ. $n = 20$ とし, 各 $p \in \{0, 0.01, 0.02, \dots, 1\}$ に対してランダムグラフを 25 個作成し, それぞれの手法の実行時間と構築された π -MDD のサイズ (辺の数) の平均を記録した. なお実装は C++ で行い, メモリ 16GB を持つ Intel(R) Core(TM) i7-3770S 3.10GHz の CPU で実行した.

5.2 結果および考察

制約グラフの有向辺の生成確率 p に対する各手法の実行時間 (対数軸) を図 7 に, 構築された π -MDD のサイズ (対数軸) を図 8 にそれぞれ示す.

これらの結果から, 既存手法は順序制約が少なくなるほど構築される π -MDD のサイズが急激に大きくなり, 実行時間も膨大になることが分かる. それに対し, 提案手法で

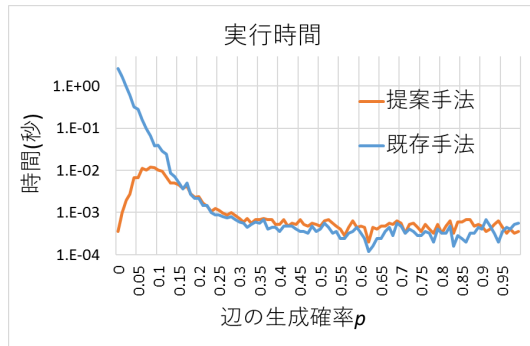


図 7 提案手法と Matsumoto らの手法の実行時間の比較

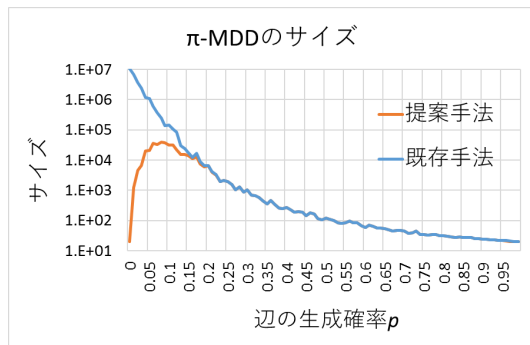


図 8 提案手法と Matsumoto らの手法の π -MDD のサイズの比較

は、順序制約がある一定値より少なくなると、構築される π -MDD のサイズがむしろ減少に転じ、それに応じて実行時間も小さくなっている。これは、順序制約が少ないと、ジョブ集合上にサイズの大きい同値類が現れる傾向が強くなり、したがって、これらのジョブ間に強制的に全順序制約を導入して得られるグラフ $\bar{G}$ の幅が小さくなるためと考えられる。

6. 今後の課題

現在の手法では、限られた場合にしか効率的に解を求められないので、より多くの場合に適用できるような手法を探したい。また、人工データだけではなく、実データを用いた実験も行いたい。それから、重みが複数与えられ、それぞれの重みに対する最適解を返すような設定も考えてみたい。

参考文献

- [1] Chekuri, C. and Motwani, R.: Precedence Constrained Scheduling to Minimize Sum of Weighted Completion Times on a Single Machine, *Discrete Applied Mathematics*, Vol. 98, No. 1-2, pp. 29–38 (1999).
- [2] Chudak, F. A. and Hochbaum, D. S.: A half-integral linear programming relaxation for scheduling precedence-constrained jobs on a single machine, *Oper. Res. Lett.*, Vol. 25, No. 5, pp. 199–204 (1999).
- [3] Hall, L. A., Schulz, A. S., Shmoys, D. B. and Wein, J.: Scheduling to Minimize Average Completion Time: Off-Line and On-Line Approximation Algorithms, *Math. Oper. Res.*, Vol. 22, No. 3, pp. 513–544 (1997).

- [4] Lawler, E. L.: *On Sequencing jobs to minimize weighted completion time subject to precedence constraints* (1978).
- [5] Lenstra, J. K. and Kan, A. H. G. R.: Complexity of Scheduling under Precedence Constraints, *Operations Research*, Vol. 26, No. 1, pp. 22–35 (1978).
- [6] Margot, F., Queyranne, M. and Wang, Y.: Decompositions, Network Flows, and a Precedence Constrained Single-Machine Scheduling Problem, *Operations Research*, Vol. 51, No. 6, pp. 981–992 (2003).
- [7] Matsumoto, K., Hatano, K. and Takimoto, E.: Decision Diagrams for solving a job scheduling problem under precedence constraints, *SEA 2018* (2018).
- [8] Schulz, A. S.: Scheduling to Minimize Total Weighted Completion Time: Performance Guarantees of LP-Based Heuristics and Lower Bounds, *Integer Programming and Combinatorial Optimization, 5th International IPCO Conference, Vancouver, British Columbia, Canada, June 3-5, 1996, Proceedings*, pp. 301–315 (1996).