ジョブショップスケジューリング

2019 年 10 月 25 日

1　問題設定

n 個の仕事（job）$J_1, \ldots, J_n$ を m 台の機械（machine）$M_1, \ldots, M_m$ 上で処理することを考える．各仕事を処理する機械の順序は仕事ごとに予め与えられており，技術的順序（technological sequence）と呼ばれる．各機械 M_r 上での仕事 J_j の処理のことを作業（operation）と呼び，$O_{j,r}$ と表す．各作業 $O_{j,r}$ は処理時間（processing time）$p_{j,r}$ をかけて機械 M_r 上で中断なく処理される．ここで各機械は全て異なり，同時に二つ以上の作業を処理することができないとする．全ての仕事を完成させるまでの時間を総作業時間（makespan）と呼び L で表す．この時，$n \times m$（総作業時間最小）一般ジョブショップスケジューリング問題（$n \times m$ *minimum-makespan* general job-shop scheduling problem）（JSSP）とは，各仕事の技術的順序と，各作業の処理時間が与えられて，L を最小にするような機械上での仕事の処理順序を全て決定することである．

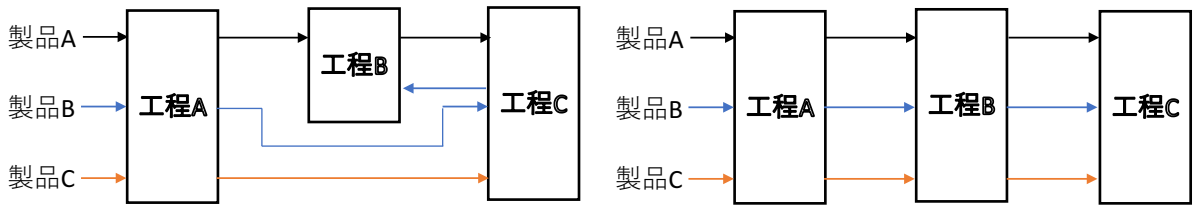
2　スケジューリングの分類

ジョブショップスケジューリング　製品を生産するのに必要な処理工程の順序が製品ごとに決まっている．

フローショップスケジューリング　全ての製品が同一な工程順序．

オープンショップスケジューリング　ジョブの一部，あるいは全部の工程順序が任意．

リエントラント性　同一の生産設備を 2 回以上使用する場合，リエントラント（re-entrant）性があるという．リエントラントジョブショップスケジューリング問題などという．



(a) ジョブショップ型　　(b) フローショップ型

図 1　スケジューリングの分類

3　背景技術

3.1　クリティカルブロック

JSSP の解の表現方法として図 2 のような選択グラフ（disjunctive graph）がよく用いられる（※注：$O_{j,r}$ の r は機械を表し，技術的順序ではないため，行列の要素とは対応しないことに注意）.

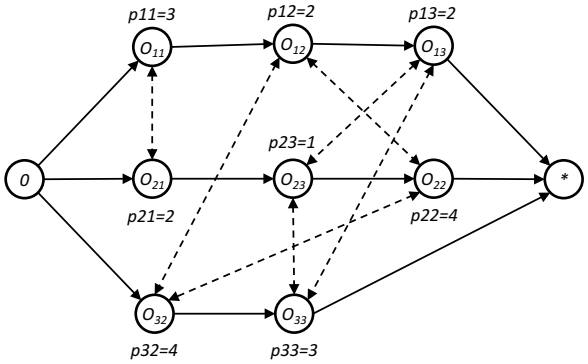


図 2　選択グラフ

JSSP は選択グラフ $G = (V, C \cup D)$ を用いて次のように表現される.

- V は節点の集合であり，作業に対応する節点及び二つの特殊節点：ソース（0），シンク（*）からなる
- C は節点間を結ぶ有向弧（conjunctive arc）の集合で，技術的順序を表す
- D は選択弧（disjunctive arc）の集合で，同一機械上の作業の対を表す

各作業 $O_{j,r}$ の処理時間 $p_{j,r}$ は各節点に付与された重み $p_{j,i}$ によって表す．完全なスケジュールは同一機械上の全ての作業について処理順序を全て決定することによって得られる．選択グラフモデルにおいて，このことは D の全ての無向（選択）弧を有向弧に変えることに対応する．この時 D より得られた有向弧の集合を「選択」（selection）S と呼ぶ．ある選択 S が実行可能スケジュールを表現していることと，有向グラフ $G(S) = (V, C \cup S)$ がサイクルを持たない（acyclic）こととは同値である．この時 s は「完全選択」（complete selection）と呼ばれる．対応するスケジュールの総作業時間は 0 から $*$ に至る最も長い重みつきパスの長さによって与えられる．このパスはクリティカルパス P と呼ばれる．クリティカルパス上の作業列 B は，次の性質が成り立つ時クリティカルブロック（または単にブロック）と呼ばれる.

- B の全ての作業は同一機械上にある
- B の最初（最後）の作業に先行（後続）する同一機械上の作業は（もし存在すれば）P 上にない

3.2　クリティカルブロック近傍

一つの解 S から一回の遷移によって到達可能な全ての解の集合を近傍 $N(S)$ と呼ぶ．JSSP において，例えばクリティカルブロック内の作業をそのブロックの一番先頭，もしくは最後へ移動させる遷移が用いられる．この遷移を用いた近傍をクリティカルブロック近傍（CB 近傍）と呼ぶ．CB 近傍 $N^C(S)$ を次のように

定義する．

B_j をクリティカルパス P 上，特定機械上の連続する作業 $u_{i,j}$ 全体からなる，j 番目のブロックとする．すなわち $B_j = (u_{1,j}, \ldots, u_{i,j}, \ldots, u_{l_j,j})$，ただし，$l_j$ は B_j 内の作業の個数である．この時 P は $(0, B_1, \ldots, B_j, \ldots, B_k, *)$ のように表現される．$l_j > 1$ なるブロック B_j に対して，次の二つの遷移操作 S^b 及び S^a を考える．

- $S^b(S, u_{i,j})(1 < i \leq l_j)$ は，S のブロック B_j 内の作業 $u_{i,j}$ を B_j の先頭に移動することによって得られる選択
- $S^a(S, u_{i,j})(1 \leq i < l_j)$ は，S のブロック B_j 内の作業 $u_{i,j}$ を B_j の最後に移動することによって得られる選択

S^b もしくは S^a によって得られる選択はそれぞれ「前候補」（before candidate）もしくは「後候補」（after candidate）と呼ばれる．S^b 及び S^a によって S より得られる全ての選択の集合は次のように表される．

$$N'^C(S) = \left\{S^b(S, u_{i,j}) | l_j > 1, i = 2, \ldots, l_j, j = 1, \ldots, k\right\} \left\{S^a(S, u_{i,j}) | l_j > 1, i = 1, \ldots, l_j - 1, j = 1, \ldots, k\right\} \tag{1}$$

$N'^C(S)$ の要素である選択の内のいくつかはサイクルを持つ可能性がある．その場合対応するスケジュールは実行可能ではない．したがって，クリティカルブロック近傍 $N^C(S)$ はそれらを除いて次のように与えられる．

$$N^C(S) = \left\{S' \in N'^C(S) | S' \text{は完全選択}\right\} \tag{2}$$

仮に S が長さ 2 以上（$l_j > 1$）のブロックを一つも持たない場合 S は最適解である．なぜならこの時 P の弧は全て C の元からなり，$L(S)$ は P 上の全作業の加工時間の和であり，これは L の下界に等しい．そうでない場合，S は少なくとも一つは長さ 2 以上のブロックを持ち，$N'^C(S)$ は決して空ではない．さらに $N'^C(S)$ は実行可能でないスケジュールを含む可能性があるが，$N^C(S)$ は決して空にならない．なぜならブロック内の連続する二つの作業の処理順序の入れ替えによって決してサイクルは生じず，したがって $S^b(S, u_{2,j})$ 及び $S^a(S, u_{l_j-1,j})$ に対応する少なくとも二つのスケジュールは常に実行可能である．

3.3 GT アルゴリズム

1 つの完全選択から対応するスケジュールを生成する際，各作業を可能な限り早く加工して一意に得られるスケジュールをセミアクティブスケジュールと呼ぶ．セミアクティブスケジュールにおいて，ある作業の処理順序を，他の作業の処理開始時刻を遅らせることなく，繰り上げて早められる場合がある（これを left shift と呼ぶ）．どの作業もこれ以上 left shift できないスケジュールをアクティブスケジュールと呼ぶ．セミアクティブスケジュールのアクティブ化はしばしば解全体の改善につながる．最適スケジュールは明らかにアクティブスケジュールであるので，探索空間を初めからアクティブスケジュールに制限して最適化することで解法の効率化が期待できる．アクティブスケジュールを生成するアルゴリズムとしては，以下の Giffler and Thompson の GT アルゴリズムが有名である．

1. 技術的順序上，次に処理可能な作業の全体を G とし，G 中，最早完了時刻が最も小さい作業を O^* とする．すなわち，$O^* = \arg\min\{O \in G | EC(O)\}$．$O^*$ を処理する機械を M^* とする．M^* が複数存在する場合，任意の一つを選択する．
2. 機械 M^* を利用する，最早開始時刻 $ES(O)$ が最早完了時刻 $EC(O^*)$ よりも小さい作業の集合（コン

フリクト集合）$C[M^*, i]$ を求める．ただし，M^* 上すでに i-1 個の作業が処理されているものとし，$C[M^*, i] = \{O \in G | ES(O) < EC(O^*)\}$.

3. M^* 上 i 番目に処理すべき作業を $C[M^*, i]$ より一つ選び O とする．
4. O の作業開始時刻を $ES(O)$ と設定する
5. 1 に戻る（O^* が空になれば終了）

3.3.1 例

GT アルゴリズムによりアクティブスケジュールを生成する例を図 3 に示す．ここで T_E は最早開始時刻（earliest node time）である．

- (a)
  1. $G = \{O_{1,1}, O_{2,1}, O_{3,2}\}$
     $EC(O_{1,1}) = 3, EC(O_{2,1}) = 2, EC(O_{3,2}) = 4 \therefore O^* = O_{2,1}, M^* = M_1$
  2. $ES(O_{1,1}) = 0, ES(O_{2,1}) = 0 \therefore C[M_1, 1] = \{O_{1,1}, O_{2,1}\}$
  3. $O = O_{1,1}$
- (b)
  1. $G = \{O_{1,2}, O_{2,1}, O_{3,2}\}$
     $EC(O_{1,2}) = 5, EC(O_{2,1}) = 5, EC(O_{3,2}) = 4 \therefore O^* = \{O_{1,2}, O_{2,1}\} \rightarrow O_{1,2}, M^* = M_2$
  2. $ES(O_{1,2}) = 3, ES(O_{3,2}) = 0 \therefore C[M_2, 1] = \{O_{1,2}, O_{3,2}\}$
  3. $O = O_{1,2}$
- (c)
  1. $G = \{O_{1,3}, O_{2,1}, O_{3,2}\}$
     $EC(O_{1,3}) = 7, EC(O_{2,1}) = 5, EC(O_{3,2}) = 9 \therefore O^* = O_{2,1}, M^* = M_1$
  2. $ES(O_{2,1}) = 3 \therefore C[M_1, 2] = \{O_{2,1}\}$
  3. $O = O_{2,1}$
- (d)
  1. $G = \{O_{1,3}, O_{2,3}, O_{3,2}\}$
     $EC(O_{1,3}) = 7, EC(O_{2,3}) = 6, EC(O_{3,2}) = 9 \therefore O^* = O_{2,3}, M^* = M_3$
  2. $ES(O_{1,3}) = 5, ES(O_{2,3}) = 5 \therefore C[M_3, 1] = \{O_{1,3}, O_{2,3}\}$
  3. $O = O_{2,3}$
- (e)
  1. $G = \{O_{1,3}, O_{2,2}, O_{3,2}\}$
     $EC(O_{1,3}) = 8, EC(O_{2,2}) = 10, EC(O_{3,2}) = 9 \therefore O^* = O_{1,3}, M^* = M_3$
  2. $ES(O_{1,3}) = 6 \therefore C[M_3, 2] = \{O_{1,3}\}$
  3. $O = O_{1,3}$
- (f)
  1. $G = \{O_{2,2}, O_{3,2}\}$
     $EC(O_{2,2}) = 10, EC(O_{3,2}) = 9 \therefore O^* = O_{3,2}, M^* = M_2$
  2. $ES(O_{2,2}) = 6, Es(O_{3,2}) = 5 \therefore C[M_2, 2] = \{O_{2,2}, O_{3,2}\}$
  3. $O = O_{3,2}$

- (g)
  1. $G = \{O_{2,2}, O_{3,3}\}$
     $EC(O_{2,2}) = 13, EC(O_{3,3}) = 12 \therefore O^* = O_{3,3}, M^* = M_3$
  2. $ES(O_{3,3}) = 9 \therefore C[M_3, 3] = \{O_{3,3}\}$
  3. $O = O_{3,3}$

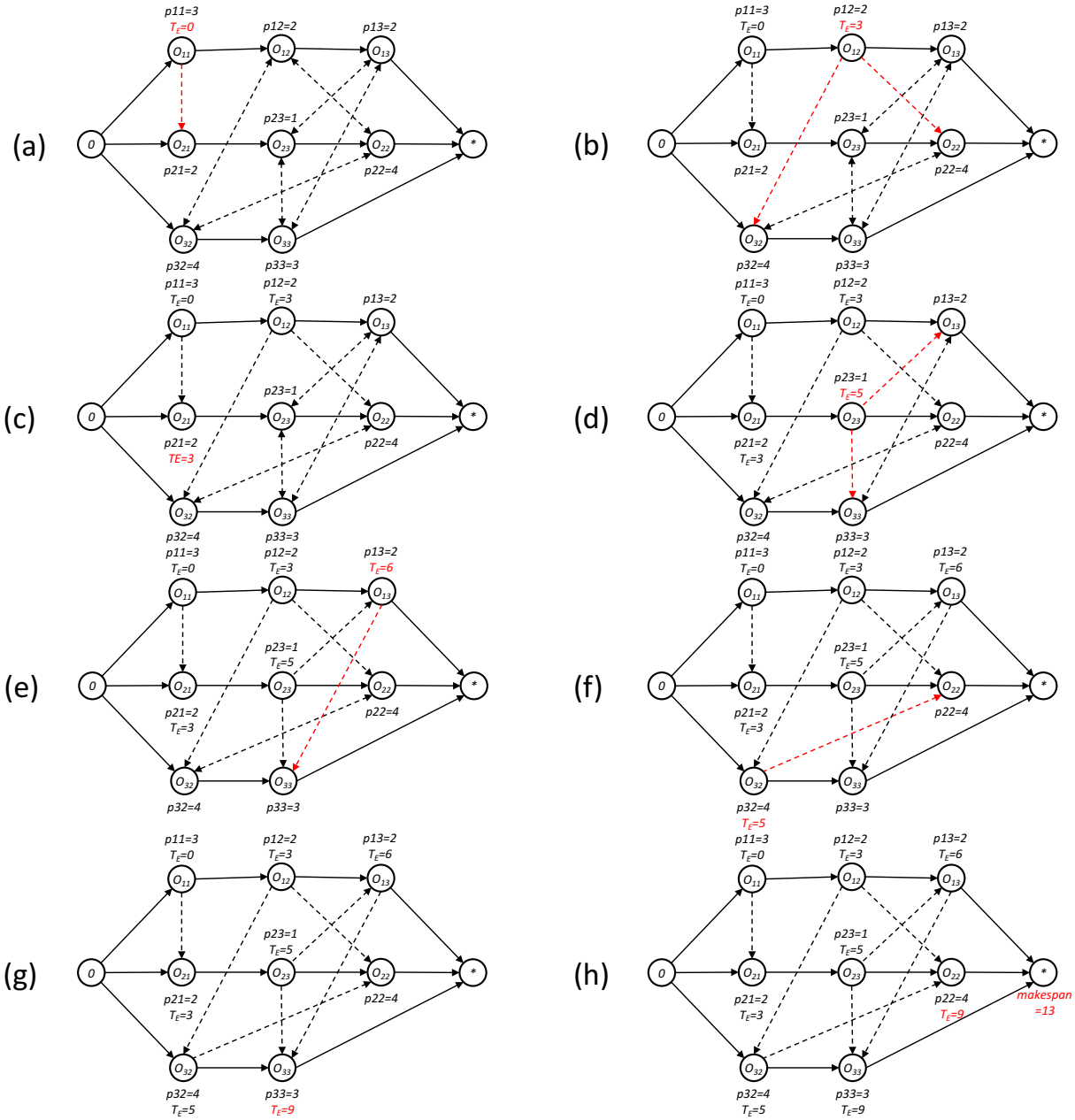


図 3　GT アルゴリズムによるアクティブスケジュール生成例

参考文献

[1] 若宮 利治，片山 謙吾，成久 洋之, ”スケジュール問題に対する有効な交叉法について”, 岡山理科大学紀要. A, 自然科学, vol. 34, pp. 307-318, 1998.

[2] 山田 武士，中野 良平, ”遺伝的局所探索法によるジョブショップスケジューリング問題の解法”, 情報処理学会論文誌, vol. 38, no. 6, pp. 1126-1138, 1997.

[3] Moonen, M. and Janssens, G.K., ”A Giffler-Thompson Focused Genetic Algorithm for the Static Job-shop Scheduling Problem,” *ITEO research paper series,* 2004,

[4] Kazi Shah Nawaz Ripon and Jim Torresen ”Integrated job shop scheduling and layout planning: a hybrid evolutionary method for optimizing multiple objectives”, *Evolving Systems,* vol. 5, no. 2, pp. 121-132, 2014.

[5] Takeshi Yamada and Ryohei Nakano, ”Chapter 7: Job-shop scheduling”, *Genetic Algorithms in Engineering Systems*, pp. 134—160, 1997.