数理解析研究所講究録
1068巻 1998年 216-224

資源制約付きスケジューリング問題の定式化と近似解法

京都大学大学院工学研究科　野々部 宏司 (Koji NONOBE) *
京都大学大学院情報学研究科　茨木 俊秀 (Toshihide IBARAKI) †

Abstract

RCSP (Resource Constrained Scheduling Problem) can formulate many scheduling problems including jobshop and flowshop. However, there are some complex scheduling problems encountered in practical applications, which the traditional RCSP can not deal with. In this paper, we extend the definition of RCSP, and develop a tabu search based heuristic algorithm with some elaborations in representation of solutions, and in construction of search space and neighborhood. Computational experiments are reported for benchmarks of jobshop and RCSPs, and also for some problems from real applications. These results indicate the effectiveness and usefulness of our approach.

1 はじめに

資源制約付きスケジューリング問題 (Resource Constrained Scheduling Problem, RCSP) は, フローショップ問題やジョブショップ問題をはじめ, 多くのスケジューリング問題を定式化できるという汎用性の高さから, 最近改めて注目を集めている. しかし現実の応用分野では, スケジュールの評価基準が複雑かつ多様である, 平日と休日とでは使用できる資源の量が異なる, 段取り替え作業を伴うなど, 従来の RCSP の枠組では扱えない問題も少なくない. そこで本論文では, より広範なスケジューリング問題を定式化できるよう RCSP を拡張する.

次に, 拡張 RCSP に対しタブー探索に基づく近似解法を提案し, ジョブショップ問題や RCSP のベンチマーク問題, さらに企業におけるスケジューリング問題に対して計算実験を行った. その結果, 多数の RCSP の問題例に対して, これまでの最良解を更新することに成功し, 現実問題に対しても実用的な解を得ることができた.

2 定式化

本論文では, RCSP を以下のように定式化する. 資源集合 $\mathcal{R} = \mathcal{R}^{\mathrm{re}} \cup \mathcal{R}^{\mathrm{non}}$, 作業集合 $\mathcal{J} = \{1, 2, \cdots, J\}$, 処理モード集合 $\mathcal{M} = \mathcal{M}_1 \cup \mathcal{M}_2 \cup \cdots \cup \mathcal{M}_J$ が与えられている. 資源集合 $\mathcal{R}$ は, 再生可能 (renewable) 資源集合 $\mathcal{R}^{\mathrm{re}}$ と再生不可能 (nonrenewable) 資源集合 $\mathcal{R}^{\mathrm{non}}$ に分割される. 再生可能資源 $r \in \mathcal{R}^{\mathrm{re}}$ は, 各単位時間 $[t-1, t)$ $(t = 1, 2, \cdots)$ に利用できる量 K_{rt}^{re} がそれまでのスケジュールに依らないのに対し, 再生不可能資源 $r \in \mathcal{R}^{\mathrm{non}}$ は, 各単位時間当たりではなく, スケジュール全体を通して利用できる量 K_r^{non} が決まっている. 例えば, 機械, 人, 作業場所など

*Department of Applied Mathematics and Physics, Graduate School of Engineering, Kyoto University, Kyoto, 606-8501, Japan. E-mail: nonobe@kuamp.kyoto-u.ac.jp
†Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto, 606-8501, Japan. E-mail: ibaraki@kuamp.kyoto-u.ac.jp

が再生可能資源, 原材料などが再生不可能資源に相当する. また, 処理モードとは作業の処理方法を表すものであり, 各処理モードに対して, 処理時間および処理に必要な資源の量が決っている. 各作業 j はそれに対応する処理モード集合 $\mathcal{M}_j$ に属するいずれか一つの処理モードで処理され, 作業 j が処理モード m_j で処理されるとき, 処理時間を p_{m_j}, 作業の処理に必要な資源集合を $\mathcal{R}_{m_j} = \mathcal{R}^{\mathrm{re}}_{m_j} \cup \mathcal{R}^{\mathrm{non}}_{m_j}$ とする. また, 処理に必要な資源量は, 再生可能資源 $r \in \mathcal{R}^{\mathrm{re}}_{m_j}$ に対しては, 処理開始後各単位時間ごとに $k^{\mathrm{re}}_{m_j ru}$ $(u = 1, 2, \cdots, p_{m_j})$, 再生不可能資源 $r \in \mathcal{R}^{\mathrm{non}}_{m_j}$ に対しては, $k^{\mathrm{non}}_{m_j r}$ (作業の処理開始時刻には依存しない) であるとする.

スケジュールは, 各作業 j の処理モード $m_j \in \mathcal{M}_j$, および開始時刻 s_j (完了時刻 $c_j = s_j + p_{m_j}$) に基づいて, $(\boldsymbol{m}, \boldsymbol{s}) = ((m_j \mid j \in \mathcal{J}), (s_j \mid j \in \mathcal{J}))$ で表される. また, 0-1 変数 x_{jm_j} を,

$$x_{jm_j} = \begin{cases} 1, & \text{作業 } j \text{ が処理モード } m_j\ (\in \mathcal{M}_j) \text{ で処理される,} \\ 0, & \text{その他,} \end{cases}$$

と定義する. ここで, スケジュールに課せられる制約は以下のように分類される.

1. 資源制約

再生可能資源制約:

$$\sum_{j \in \mathcal{J}_{rt}} k^{\mathrm{re}}_{m_j r(t-s_j)} \le K^{\mathrm{re}}_{rt}, \qquad r \in \mathcal{R}^{\mathrm{re}}, t = 1, 2, \cdots, T. \tag{1}$$

ここで, $\mathcal{J}_{rt} = \{j \mid r \in \mathcal{R}^{\mathrm{re}}_{m_j}, s_j + 1 \le t \le c_j\}$ (すなわち, 資源 r を必要とし, 時間 $[t-1, t)$ に処理中である作業の集合), T は最大完了時刻 (makespan) $\max_j c_j$ の上界値である.

再生不可能資源制約:

$$\sum_{j \in \mathcal{J}} \sum_{r \in \mathcal{R}^{\mathrm{non}}_{m_j}} k^{\mathrm{non}}_{m_j r} x_{jm_j} \le K^{\mathrm{non}}_r, \quad r \in \mathcal{R}^{\mathrm{non}}. \tag{2}$$

例: 機械

機械 r が同時に処理することのできる作業の数を h とすると, 機械 r は,

$$\begin{aligned} &K^{\mathrm{re}}_{rt} = h, \quad t = 1, 2, \cdots, T, \\ &k^{\mathrm{re}}_{m_j ru} = 1, \quad r \in \mathcal{R}^{\mathrm{re}}_{m_j},\ u = 1, 2, \cdots, p_{m_j} \end{aligned} \tag{3}$$

を満たす再生可能資源 r として扱うことができる. □

2. 先行制約:

$$c_{j_1} \le s_{j_2}, \qquad j_1 \prec j_2. \tag{4}$$

ここで, $\prec$ は作業集合 $\mathcal{J}$ に対して予め与えられている先行関係 (半順序) である.

直前先行制約:

作業 j_1, j_2 $(j_1 \prec j_2)$ に対して,

$$\begin{aligned} &c_{j_1} \le s_{j_2}, \\ &s_{j_2} \le s_{j'}, \quad j' \in \{j' \mid c_{j_1} \le s_{j'}\} \end{aligned} \tag{5}$$

を直前先行制約 $j_1 \prec\prec j_2$ と定義し, さらに, 作業を再生可能資源 r 上に限定した制約

$$r \in \mathcal{R}^{\mathrm{re}}_{m_{j_1}}, r \in \mathcal{R}^{\mathrm{re}}_{m_{j_2}} \quad \Rightarrow \quad \begin{array}{l} c_{j_1} \leq s_{j_2}, \\ s_{j_2} \leq s_{j'}, \quad j' \in \{j' \mid r \in R^{\mathrm{re}}_{m_{j'}},\ c_{j_1} \leq s_{j'}\} \end{array} \tag{6}$$

を資源 r 上における直前先行制約 $j_1 \prec\prec_r j_2$ と定義する.

3. その他の制約:

0-1 変数 x_{jm_j}, 開始時刻 s_j, 完了時刻 c_j, 処理時間 $p_j = p_{m_j}$ に関する上記以外の制約.

現実のスケジューリング問題では, その評価基準は, 最大完了時刻最小, 総納期遅れ時間最小など, 状況に応じて変化するため, 予め目的関数を定めておくことは好ましくない. さらに, より複雑な評価基準を用いることも多い. このため, 本定式化ではスケジュールの評価を制約を用いて行なう. すなわち, 上述の制約を, 必ず満たさなくてはならない制約 (絶対制約) $C^h_1, C^h_2, \cdots, C^h_{L_h}$ と満たすことが望ましいが必ずしも満たさなくてもよい制約 (考慮制約) $C^s_1, C^s_2, \cdots, C^s_{L_s}$ とに分類し, 絶対制約を満たす (実行可能である) 範囲で, 考慮制約の満足度を最大にするという意味でスケジュールを最適化するのである. これを実現するために, 各考慮制約 C^s_l に対して, それが満たされる (満たされない) ならば 0 (正の値) をとるようなペナルティ関数 P_l, および制約 C^s_l の重要度を示す重み w_l (> 0) を導入する. これらを用いて, 全体としての重みつきペナルティ関数 $P = \sum_{l=1}^{L_s} w_l P_l$ を目的関数とし, 最小化する. 各考慮制約に対するペナルティ関数は, 例えば, (左辺) $\leq$ (右辺) の形の不等式に対しては, $P_l = \max\{0, (\text{左辺}) - (\text{右辺})\}$ などとすればよい.

例: 最大完了時刻最小化

作業 sink の完了時刻 c_{sink} に対する等式制約

$$c_{\mathrm{sink}} = 0 \tag{7}$$

を考慮制約とし, c_{sink} をペナルティ関数とする. ここで作業 sink は, 全ての作業に後続する処理時間 0 の仮想的な作業である. 制約 (7) は, $(p_j = 0\ (j \in \mathcal{J})$ でない限り) 満たされることはあり得ないが, 制約 (7) に関するペナルティを減らすことは, 最大完了時刻を小さくすることを意味する. □

例 1. ジョブショップ問題

ジョブショップ問題とは, それぞれ幾つかの作業からなる N 個の仕事と R 台の機械に対して, 最大完了時刻を最小化するスケジュールを求める問題である. ただし, 各仕事を構成する作業間にはそれらの処理順序を定める先行関係が定義されており, また, 各作業を処理する機械は予め定められている.

ジョブショップ問題は, 各機械 r に対して, $h = 1$ として (3) を満たす再生可能資源 r を導入し, 資源制約 (同一機械上で同時に複数の作業の処理ができないことに対応) と先行制約を絶対制約, 等式制約 (7) を考慮制約とすれば, 容易に RCSP に定式化できる. なお, 各作業 j は唯一の処理モードを持ち, その選択の問題は生じない. □

3 段取り替え作業

現実の問題では, ある機械 r 上で作業 j_1 から 作業 j_2 に処理が移行する際, j_1 と j_2 に依存した段取り替え作業が必要であることが多い. 本定式化においては, 以下のようにして段取り替え作業を扱うことができる. ただし, 機械 r 上で複数の作業を同時に処理することはできないものとする.

- 作業 j_2 に必要な, 機械 r 上の段取り替え作業 $\tilde{j}_2$ も通常の作業と見なす.
- 機械 r に対応する再生可能資源 r ($h=1$ として (3) を満たす) を導入する.
- 資源 r 上における直前先行制約 $\tilde{j}_2 \prec\prec_r j_2$ を加える.
- 作業 $\tilde{j}_2$ は処理モード集合 $\mathcal{M}_{\tilde{j}_2}$ を持ち, 資源 r 上で直前に処理される作業 j_1 に応じて, 処理モード $m_{\tilde{j}_2}$ が一意に定まるよう制約を記述する.

例 2. 企業における現実問題 (1)

この例は, 段取り替え作業を伴うスケジューリング問題であり, J種の製品 $\mathcal{J}=\{1,2,\cdots,J\}$ を機械 $r=1,2,\cdots,R$ で処理する際の T 日間のスケジュールを求める問題である. ここで,

- 各製品 j が処理される機械は予め決められている (すなわち, $\mathcal{J}$の分割 $\mathcal{J}=\cup_{r=1}^{R}\mathcal{J}_r$が与えられている).
- 各製品 j に対して, 処理日数 p_j は分かっている.
- 各機械において, 異なる製品への移行には q 日間の段取り替え作業が必要である.
- 各機械は, ある一つの製品を処理しているか, 段取り替え作業を行っているかのどちらかである. これを可能にするため,

$$\sum_{j\in\mathcal{J}_r} p_j + q(|\mathcal{J}_r|-1) = T, \qquad r=1,2,\cdots,R \tag{8}$$

が成立するものとする.

このとき, 各日 t において段取り替え作業を行なっている機械の台数が上限 k_tを越えないように各機械 r のスケジュール ($|\mathcal{J}_r|$ 個の製品の処理順序) を決定することが求められている.

この問題は, 以下のように RCSP に定式化できる:

- 各製品 j, および j に必要な段取り替え作業 $\tilde{j}$ を, それぞれ, 唯一の処理モードを持つ作業 j と処理モード集合 $\mathcal{M}_{\tilde{j}}=\{\text{dummy}, m_{\tilde{j}}\}$ を持つ作業 $\tilde{j}$ に対応させる. ここで, dummy は処理時間 0 の仮想的な処理モードである.
- 資源は, 各機械に対応する R 個の再生可能資源 $r_1,r_2,\cdots,r_R$, および段取り替え作業に対応する再生可能資源 r' の計 $R+1$ 個である. ここで, 資源 $r=r_1,r_2,\cdots,r_R$ は $h=1$ として (3) を満たす. また, 資源 r' の供給量 $K^{\mathrm{re}}_{r't}$ は各日における段取り替え作業数の上限 k_t $(t=1,2,\cdots,T)$, 必要量は各段取り替え作業 $\tilde{j}$ に対し, $k^{\mathrm{re}}_{m_{\tilde{j}}r'u}=1$ $(u=1,2,\cdots,q)$ である.

- 制約は, (i) 資源制約 (1), (ii) 各段取り替え作業 $\tilde{j}$ に対して, 資源 r ($\in \mathcal{R}_j^{\mathrm{re}}$) 上における直前先行制約 $\tilde{j} \prec\prec_r j$, (iii) 段取り替え作業 $\tilde{j}$ の処理モードに関する制約 (段取り替え作業 $\tilde{j}$ が資源 r ($\in \mathcal{R}_j^{\mathrm{re}}$) 上で最初に処理される作業であれば dummy, さもなくば $m_{\tilde{j}}$), (iv) J個の不等式制約

$$c_j \leq T, \qquad j \in \mathcal{J} \tag{9}$$

である. ここで, (9) が成立すれば, 制約 (8) によって, 各機械は遊休時間を持たないことになる.

この問題では, 常に実行可能スケジュールが存在するとは限らない. また, 実際に求められているのが R 機械上の T 日間のスケジュールであるため, 機械に関する資源制約 (1), 直前先行制約, および 制約 (9) を破るわけにはいかない. そこで, それらを絶対制約として扱い, 残った段取り替え作業に関する資源制約を考慮制約として定式化するのが好ましいと考えられる. □

例 3. 企業における現実問題 (2)

例 2 において, 段取り替え作業にかかるコストが高い場合は, 各製品 j の処理日数 p_j を変更してでも, 各日における段取り替え作業数を上限以下にしたいという状況も考えられる. そこで, 次のように考える.

- 各製品 j の処理日数は, 要求日数 p_j に対して, $p_j - 1, p_j, p_j + 1$ のいずれかであればよい.
- 全ての制約を満たすスケジュールが存在するならば, できるだけ各製品の処理日数を要求通りにすることが望ましい.

この問題は, 例 2 に以下の変更を加えることで対応できる:

- 各作業 j に対して, 処理モード集合 $\mathcal{M}_j = \{m_j^-, m_j^0, m_j^+\}$ を導入し, それぞれ, 処理日数が 1 日不足, 要求通り, 1 日過剰の場合に対応させる.
- 2 種類の制約

$$\sum_{j \in \left\{ j | r \in \mathcal{R}_{m_j}^{\mathrm{re}} \right\}} -x_{jm_j^-} + x_{jm_j^+} = 0, \qquad r = r_1, r_2, \cdots, r_R \tag{10}$$

$$\sum_{j \in \mathcal{J}} x_{jm_j^-} + x_{jm_j^+} = 0 \tag{11}$$

を追加する.

ここで, 製品 j の処理日数が 1 日不足 (1 日過剰) であるとき, またそのときに限り, $x_{jm_j^-} = 1$ ($x_{jm_j^+} = 1$) である. (10) は (9) と同様, T 日間のスケジュールを得るために必要であり, 絶対制約である. 一方 (11) は, 製品 j の処理日数が要求通り p_j となるようにする制約であり, 考慮制約である. □

4 アルゴリズム

これまでに, RCSP に対して多くの近似解法が提案されている. 代表的なものには, 優先規則 (priority rule) に基づいてスケジュールを作成していくものなどがあるが [4], 近年, 局所探索に基づく近似解法も幾つか提案されている [1, 6]. 大規模な問題に対する局所探索法 (およびその変形) の有用性は RCSP のみならず, 多くの組合せ最適化問題に対して示されている. 本論文でも, 局所探索法を基本とした近似解法であり, メタ解法の一つであるタブー探索を用いる.

4.1 探索空間

2章で述べた通り, 解は各作業 j の処理モード, および開始時刻を表すベクトル $(\boldsymbol{m}, \boldsymbol{s})$ で表される. しかし, 一般に, ある実行可能解 (全ての絶対制約を満たす解) $(\boldsymbol{m}, \boldsymbol{s})$ から, 一つの作業 j の開始時刻 s_j を変化させて得られる解 $(\boldsymbol{m}, \boldsymbol{s}')$ は, 実行可能であるとは限らない. そのため, $(\boldsymbol{m}, \boldsymbol{s}')$ から, 再び実行可能解 $(\boldsymbol{m}, \boldsymbol{s}'')$ を得るためには, 他の作業の開始時刻の変更を含めた複数の局所的変更が必要となる. したがって, 局所的変更の定義を拡張して, 一回の変更で実行可能解 $(\boldsymbol{m}, \boldsymbol{s}'')$ が得られるとすれば, より効果的な局所探索が期待できる. そこで本論文では, 絶対制約のうち, 再生可能資源制約 (1), 先行制約 (4) および直前先行制約 (5)(6) の全てを満たす解 (以下, 疑似実行可能解と呼ぶ) に限定して局所探索を行うことを考える. そのために, 全作業の順列 $\boldsymbol{\pi} = (\pi_1, \pi_2, \cdots, \pi_J)$ を用意し, $\boldsymbol{\pi}$ に従って開始時刻 s_{π_i} を前から順に決定していくことによりスケジュールを構成する. 以下, 簡単化のため直前先行制約はないものと仮定して, その手順を説明する. 各作業 π_i の開始時刻 s_{π_i} を, $s_{\pi_1}, s_{\pi_2}, \cdots, s_{\pi_{i-1}}$ はすでに決定しているという状況で, 疑似実行可能解が得られるように最早時刻に定めていくのである.

CONSTRUCT
入力: $(\boldsymbol{m}, \boldsymbol{\pi})$, 出力: $(\boldsymbol{m}, \boldsymbol{s})$.

ステップ 0 (**初期設定**) : $i := 1$.

ステップ i : 作業 π_i に対し, 疑似実行可能となる最早開始時刻を計算し s_{π_i} とする.
$i < J$ ならば $i := i + 1$ としてステップ i へ. さもなければ終了.

順列 $\boldsymbol{\pi}$ から, CONSTRUCT により生成されるスケジュールが疑似実行可能となるためには,

$$\pi_{i_2} \not\prec \pi_{i_1}, \qquad 1 \le i_1 < i_2 \le J, \tag{12}$$

すなわち, $\boldsymbol{\pi}$ が先行関係を定義する半順序 $\prec$ に対するトポロジカル整列であればよい. 条件 (12) を満たす順列 $\boldsymbol{\pi}$ の集合を Π と記す. 本論文のタブー探索では, 探索空間として

$$\left\{(\boldsymbol{m}, \boldsymbol{\pi}) \mid \pi_j \in \mathcal{M}_j,\ \boldsymbol{\pi} \in \Pi\right\} \tag{13}$$

を用い, 資源制約 (1), 先行制約 (4), 直前先行制約 (5)(6) 以外の絶対制約 C_l^h は, 十分大きな重みを持つ考慮制約として扱い, それらを考慮に入れたペナルティ関数 P を目的関数として用いる. もちろん, 探索の途中, 絶対制約 C_l^h を満たさない解を訪問することもあるが, 最終的に得られる暫定解 (探索中に見つかった最良解) は C_l^h を満たしていると期待できる.

ここで, 探索空間を (13) とすることで探索効率を高めている反面, 問題例によっては最適スケジュールを求めることが原理的に不可能である場合も存在することに注意する必要がある.

CONSTRUCT の計算量は, 簡単化のため, 再生可能資源の供給量および必要量が時間によらず一定, すなわち, $K^{\mathrm{re}}_{rt} = K^{\mathrm{re}}_{r}$ $(t = 1, 2, \cdots, T,\ r \in \mathcal{R}^{\mathrm{re}})$, $k^{\mathrm{re}}_{m_j ru} = k^{\mathrm{re}}_{m_j r}$ $(u = 1, 2, \cdots, p_{m_j}, r \in \mathcal{R}^{\mathrm{re}}_{m_j})$ を仮定すれば, $O(\sum_{j \in \mathcal{J}}(J_j R_j \log R_j) + P)$ であることが示せる. ここで,

$$J_j = \max_{\substack{m_j \in \mathcal{M}_j \\ r \in \mathcal{R}^{\mathrm{re}}_{m_j}}} \left| \left\{ j' \mid \exists m_{j'} \in \mathcal{M}_{j'}, r \in \mathcal{R}^{\mathrm{re}}_{m_{j'}} \right\} \right|, \quad R_j = \max_{m_j \in \mathcal{M}_j} |\mathcal{R}^{\mathrm{re}}_{m_j}|.$$

また, P は絶対制約である先行制約の数である.

4.2 近傍

現在の解 $(\boldsymbol{m}, \boldsymbol{\pi})$ に対して,

(i) ある作業 j の処理モード m_j を他の処理モード m'_j $(\in \mathcal{M}_j)$ に変更する,

(ii) 順列 $\boldsymbol{\pi}$ において, ある作業 π_{i_2} を作業 π_{i_1} の直前に移す,

という局所的な変更によって $(\boldsymbol{m}, \boldsymbol{\pi})$ の近傍 $N(\boldsymbol{m}, \boldsymbol{\pi})$ を定義する. ただし (ii) においては, 新しい順列 $\boldsymbol{\pi}'$が $\boldsymbol{\pi}' \in \Pi$ を満たすように, 例えば $i_1 < i_2$ であれば $\pi_{i_1} \not\prec \pi_{i_2}$ でなくてはならず, さらに, $i_1 < i' < i_2$, $\pi_{i'} \prec \pi_{i_2}$ となる作業 $\pi_{i'}$ が存在する場合, $\pi_{i'}$ も π_{i_1} より前に移動する.

なお, 局所探索中, 近傍 $N(\boldsymbol{m}, \boldsymbol{\pi})$ 内の解全てを候補に探索を行うことは, 各解の評価に時間がかかることを考慮すると非効率的である. そこで, 現在の解 $(\boldsymbol{m}, \boldsymbol{\pi})$ が満足していない制約 C_l に注目し, そのような C_l の少なくとも一つを満たす効果があると思われる局所的変更のみを試みる. 具体的には, 制約 C_l に関係する作業 j に限定して (i) を, また, 制約 C_l に, 開始時刻 s_j, あるいは終了時刻 c_j が関係している作業 j に限定して ($\pi_{i_2} = j$ として) (ii) を適用する. さらに, ある作業 j' の開始時刻を早めることが求められている場合, 以下のような変更を試みることも効果的である: 解 $(\boldsymbol{m}, \boldsymbol{\pi})$ に対して,

$$E_P = \{(j_1, j_2) \mid j_1 \prec j_2 \text{ かつ } c_{j_1} = s_{j_2}\},$$

$$E_R = \{(j_1, j_2) \mid \text{作業 } j_2 \text{ の開始時刻が, 作業 } j_1 \text{ との資源の競合により遅れた}\}$$

とする. このとき, $\boldsymbol{\pi} \in \Pi$ であること, CONSTRUCT では, $\pi_1, \pi_2, \cdots, \pi_J$ の順に開始時刻が決定していくことから, $j_1 = \pi_{i_1}$, $j_2 = \pi_{i_2}$ とすれば, $(j_1, j_2) \in E_P \cup E_R \Rightarrow i_1 < i_2$ である. 有向グラフ $(\mathcal{J}, E_P \cup E_R)$ において, 上述の作業 j' に至る有向路上にあり, $(j_1, j_2) \in E_R \setminus E_P$ である枝 (作業の組) に対して (ii) を適用する.

4.3 タブー探索

上述の探索空間, 近傍を用いてタブー探索 [3] を行う. タブーリストには, 局所的変更 (i) に対しては作業 j, (ii) に対しては作業 j_2 を保持する. また, 本アルゴリズムでは, タブー探索の基本的要素に加え, 文献 [7] と同様の方法でパラメータ tabu tenure の自動調節を行っている.

5 計算実験

前章で述べたアルゴリズムを用いて, ジョブショップ問題, RCSP のベンチマーク問題, および現実のスケジューリング問題に対して計算実験を行った. なお, アルゴリズムは C 言語で記述し, 計算実験は全てワークステーション Sun Ultra2 (300MHz) 上で行った.

5.1 ジョブショップ問題

ジョブショップ問題のベンチマーク問題として有名な, Fisher & Thompson の問題例 ft10 (10 仕事 10 機械, 100 作業) と ft20 (20 仕事 5 機械, 100 作業)[2] を用いた. これらの問題例に対してはともに最適値が分かっている. 問題例 ft10, ft20 に対して, 1 回 300 秒の探索をそれぞれ 30 回ずつ行った結果を表 1 に示す.

表 1: ジョブショップ問題に対する計算結果.

問題例	最適値	計算実験で得られた値		
		最小値	平均値	最大値
ft10	930	930	943.1	954
ft20	1165	1182	1194.6	1209

ft10 に対しては最適値が, ft20 に対しても最適値からの誤差 1.6% 程度の近似解を得ることができた. 本アプローチがジョブショップ問題を含め, より広範な問題を定式化することのできる RCSP に対するものであることを考慮すれば, 本解法の性能は満足できるものであると言える.

5.2 PSPLIB

PSPLIB[5] (Project Scheduling Problem LIBrary) には, RCSP のベンチマーク問題が数多く用意されており, 問題例やこれまでの最良値などがホームページ[1] から入手できる. 計算実験では, その中から j60*_*.sm, j90*_*.sm, j120*_*.sm, j30*_*.mm の 4 タイプ, 計 2110 個の問題例を用いた. 各タイプのサイズ等は以下の通りである. また, 4 タイプとも先行制約付きであり, 最大完了時刻を最小化することが目的である.

問題例のタイプ	$\lvert\mathcal{J}\rvert$	$\lvert\mathcal{M}_j\rvert$	$\lvert\mathcal{R}^{\mathrm{re}}\rvert$	$\lvert\mathcal{R}^{\mathrm{non}}\rvert$
j60*_*.sm	60	1	4	0
j90*_*.sm	90	1	4	0
j120*_*.sm	120	1	4	0
j30*_*.mm	30	3	2	2

j60*_*.sm と j30*_*.mm に対しては 1 回 30 秒, j90*_*.sm と j120*_*.sm に対しては 1 回 60 秒の探索を, 各 5 回ずつ行った. その結果を表 2 に示す. 多数の問題例に対してこれまでの最良値を更新することができた.

5.3 現実問題

2章の例 3 で述べた問題に対して計算実験を行った. ここでは $J = 77$, $R = 19$, $T = 30$, $q = 1$ の問題例を用いた. 各機械 r 上で処理される製品の数 $|\mathcal{J}_r|$ は 2 ~ 7 のいずれかであり, 要求処理日数 p_j $(j \in \mathcal{J})$ の最小値は 1 日, 最大値は 25 日である. また, 各日の段取り替え作業数の上限 k_t は (第 1 日を月曜日として) 平日は 3 回, 土日は 0 回である.

[1] http://www.bwl.uni-kiel.de/Prod/psplib/index.html

表 2: PSPLIB に対する計算結果.

問題例のタイプ	問題例の数	最良値を求めた数[1]
j60*_*.sm	480	387 (2)
j90*_*.sm	480	411 (34)
j120*_*.sm	600	433 (213)
j30*_*.mm	550	540 (116)

[1] () は最良値を更新した数で内数.

この問題に対して 1 回 300 秒の探索を 30 回行ったところ, 30 回全てにおいて, 段取り替え作業数に関する全ての制約を満足する解が得られ, うち 11 回の探索においては, 処理日数を変更した (すなわち, 処理モードが m_j^- または m_j^+ である) 作業が 4 つである解が求まった. このように, 段取り替え作業のある現実問題に対して, 高い確率で実用的な解を得ることができた.

なお本問題では, 段取り替え作業の処理日数 q を 1 日と固定しているが, 先行作業と後続作業に依存して変化する問題を扱うことも可能である.

6 今後の課題

本論文における近似解法では, 原理的に最適スケジュールを得ることができない問題例が存在する. そのような問題例に対しても有効な近似解法を開発することが今後の課題である.

参考文献

[1] Cho, J-H. and Y-D. Kim, "A simulated annealing algorithm for resource constrained project scheduling problems", *Journal of the Operational Research Society* 48 (1997) 736-744.

[2] Fisher, H. and G.L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules", in: J.F. Muth and G.L. Thompson (eds.), *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ.

[3] Glover, F., "Tabu search - Part I", *ORSA Journal on Computing* 1 (1989) 190-206.

[4] Kolisch, R., "Serial and parallel resource-constrained project scheduling methods revisited: Theory and computation", *European Journal of Operational Research* 90 (1996) 320-333.

[5] Kolisch, R. and A. Sprecher, "PSPLIB – A project scheduling library", *European Journal of Operational Research* 96 (1997) 205-216.

[6] Mori, M. and C.C Tseng, "A genetic algorithm for multi-mode resource constrained project scheduling problem", *European Journal of Operational Research* 100 (1997) 134-141.

[7] Nonobe, K. and T. Ibaraki, "A tabu search approach to the CSP (Constraint Satisfaction Problem) as a general problem solver," *European Journal of Operational Research* 106 (1998) 599-623.