2001年度日本オペレーションズ・リサーチ学会
秋季研究発表会

# 勤務スケジューリング問題に対する局所探索法

京都大学　*笠野 学　KASANO Manabu
01405524　京都大学　野々部 宏司　NONOBE Koji
01001374　京都大学　茨木 俊秀　IBARAKI Toshihide

## 1　はじめに

実社会においては, 工場, 病院, 製造業, サービス業務など, 様々な場面で従業員の勤務スケジュールを作成することが求められている. この種の問題の多くは組合せ最適化問題として定式化することができ, これまでに数多くの手法が提案されてきた. しかし, 現実に現れる勤務スケジューリング問題は多種多様であり, これらを包括的に扱うことを目的とした研究は数少ない. 本研究の目的は, 様々なタイプの勤務スケジューリング問題を扱うことのできる汎用アルゴリズムを開発することである. そのためにまず, 汎用性に留意しつつ問題の定式化を行い, 次に局所探索法に基づく近似解法の適用を行った. 本研究で提案する局所探索法では, 探索するスケジュールを各従業員の勤務パターンに関する制約を満たすものに限定することで, 探索の効率化を図っている. このアルゴリズムを用いて, 文献や現実に現れる勤務スケジューリング問題を解いたところ, 現時点では, 計算速度, 解の質の点において専用手法よりやや劣る結果となっている. しかし, 本アルゴリズムの適用範囲は他の手法と比べて広く, メタ戦略 [1] など, より高度な手法と組合せることで, 実用性を高めることが期待できる.

## 2　問題の定式化

従業員の集合を $I$ , 勤務タイプの集合を $Q$ とし, スケジュール期間を $T$ とする. 各従業員 $i \in I$ の各期間 $t \in [1,T]$ には, いずれかの勤務タイプ $q \in Q$ が割当てられるものとする. ただし, 従業員が勤務タイプ $q \in Q$ を行う際には, 少なくとも $k_q^-$ 期間連続して行わなくてはならず, 高々 $k_q^+$ 期間しか連続して行うことはできない (連続期間数制約). また, 従業員が勤務タイプ $q$ を終えた後に続けて行なうことのできるタイプは限られており, 集合 $A_q \subseteq Q$ で与えられるものとする. つまり, $q' \in A_q$ のとき, かつ, そのときに限り $q$ の次に $q'$を行うことができる (連続勤務タイプ制約). これにより従業員が行うことのできる勤務パターンを制限することができる.

さらに本定式化では, 以下の制約集合 $C$ を記述することができる. 各制約 $c \in C$ は, 期間 $[t_c^-, t_c^+]$ $(1 \le t_c^- \le t_c^+ \le T)$ において, 集合 $I_c \subseteq I$ に含まれる従業員が集合 $Q_c \subseteq Q$ に含まれる勤務タイプを行うのべ人数の上限 $(u_c)$ , 下限 $(l_c)$ で与えられる. すなわち, 0-1 変数 $x_{i,t,q}(i \in I, t \in [1,T], q \in Q)$ を,

$$x_{i,t,q} = \begin{cases} 1, & \text{従業員 } i \text{ の期間 } t \text{ に} \\ & \quad \text{タイプ } q \text{ が割当てられている,} \\ 0, & \text{その他,} \end{cases}$$

と定義すれば,

$$l_c \le \sum_{i \in I_c} \sum_{t \in [t_c^-, t_c^+]} \sum_{q \in Q_c} x_{i,t,q} \le u_c, \qquad c \in C \qquad (1)$$

である. このタイプの制約を用いることで, 与えられた期間における必要人数や, 各従業員の勤務日数などに関する制約を自由に記述することができる.

本研究では連続期間数制約および連続勤務タイプ制約を満たした上で, $C$ に含まれる制約の違反度の総和を最小化することを目的とする. そのために各制約 $c \in C$ に対して, ペナルティ係数 $\alpha_c, \beta_c (\ge 0)$ を導入し, 制約 $c$ に関する違反度 $p_c(\boldsymbol{x})$ を

$$p_c(\boldsymbol{x}) = \alpha_c \times max\Big\{0, \sum_{i \in I_c} \sum_{t \in [t_c^-, t_c^+]} \sum_{q \in Q_c} x_{i,t,q} - u_c\Big\} + \beta_c \times max\Big\{0, l_c - \sum_{i \in I_c} \sum_{t \in [t_c^-, t_c^+]} \sum_{q \in Q_c} x_{i,t,q}\Big\}$$

と定義する. ここで, $\boldsymbol{x}$ は $|I| \cdot T \cdot |Q|$ 次元 0-1 ベクトル $(x_{i,t,q} | \, i \in I, t \in [1,T], q \in Q)$ である. なお 集合 $C$ に含まれる制約は必ず満たされるとは限らないが, 必ず満たすべき制約 $c$ に対しては, $\alpha_c$ , $\beta_c$を大きな値に設定することで対処する.

つぎに, 連続期間数制約, 連続勤務タイプ制約を簡潔に記述するため, $\boldsymbol{x}$ とは異なる解の表現方法を導入する: 従業員 $i$ が期間 $[1,T]$ に行う勤務タイプを, 勤務タイプの列 (勤務パターンと呼ぶ) $\boldsymbol{y}_i = (y_{i,h} \in Q | \, h = 1,2,\ldots,d_i)$ と正整数の列 $\boldsymbol{z}_i = (z_{i,h} \in Z_+ | \, h = 1,2,\ldots d_i)$ で表す. すなわち, 従業員 $i$ は, 勤務タイプ $y_{i,h}$ を $h = 1,2,\ldots,d_i$ の

順に, それぞれ $z_{i,h}$ 期間ずつ行う. ただし, $y_{i,h} \neq y_{i,h+1}$ である $(h = 1, 2, \ldots, d_i - 1)$. 以下, 解 $\boldsymbol{x}$ と, それに対応する $(\boldsymbol{y} = (\boldsymbol{y_i} | \ i \in I), \boldsymbol{z} = (\boldsymbol{z_i} | \ i \in I))$ を同一視し, 都合の良い方を用いる.

以上をまとめると, 本研究で扱う勤務スケジューリング問題は以下のように定式化される:

$$
\begin{aligned}
min \quad & \sum_{c \in C} p_c(\boldsymbol{y}, \boldsymbol{z}) \\
s.t. \quad & \sum_{h=1}^{d_i} z_{i,h} = T, \qquad \forall i \in I, \\
& k^-_{y_{i,h}} \leq z_{i,h} \leq k^+_{y_{i,h}}, \quad \forall i \in I, \forall h \in \{1, \ldots, d_i\}, \\
& y_{i,h+1} \in A_{y_{i,h}}, \qquad \forall i \in I, \forall h \in \{1, \ldots, d_i - 1\}.
\end{aligned}
$$

これは, 高い汎用性を持っており, 多種多様な勤務スケジューリング問題を包括的に扱うことができる.

## 3 局所探索法

局所探索法とは, ある初期解 $\boldsymbol{x}^{(0)}$ から出発し, 現在の解 $\boldsymbol{x}$ に少し変形を加えることによる解の改善を繰り返し行う方法である. このとき, 現在の解 $\boldsymbol{x}$ に少し変形を加えることで得られる解の集合を近傍 $N(\boldsymbol{x})$ と呼ぶ. 本研究では, 局所探索法に少し工夫を加えた, 反復局所探索法を用いる. 反復局所探索法とは, 近傍内に改善解が存在しなくなったとき, 過去の探索で得られた良い解に通常の探索とは異なる変形を加えたものを初期解として, そこから再び局所探索を行う方法である. 以下, 本研究で用いる 3 つの近傍について述べる.

### 3.1 パターン変更近傍

パターン変更近傍は, 従業員 $i$ のパターン $\boldsymbol{y_i}$ の中の一つの勤務タイプ $y_{i,h}$ を他のタイプに変更することで得られる解の集合である. ただし, そのような解は一般的に実行可能であるとは限らないため, 必要に応じて $y_{i,h-1}, y_{i,h+1}$ など, 前後の勤務タイプの変更を併せて行ったり, それらの期間数を調整するなどして実行可能性を保っている. また, このようにして得られる解をすべて探索するのは効率が悪いため, 以下のようにして近傍の削減を行う.

いま, 制約 $c \in C$ に対し, 期間 $t \in [t^-_c, t^+_c]$ において従業員 $i \in I_c$ によるタイプ $q \in Q_c$ が不足 (超過) しているとする. このとき, その範囲内の $q \notin Q_c$ のタイプを $q \in Q_c$ に変更 ($q \in Q_c$ のタイプを $q \notin Q_c$ に変更) することで違反度 $p_c(\boldsymbol{x})$ を減らすことができる. このように, 現在違反している制約それぞれに対し, その違反度を減少させるタイプ変更のみに限定して近傍探索を行う.

### 3.2 期間変更近傍

期間変更近傍は, 従業員 $i$ のパターン $\boldsymbol{y_i}$ を変えずに, 期間数 $\boldsymbol{z_i}$ だけを変更してできる解の集合である. ある勤務タイプ $y_{i,h}$ の期間数 $z_{i,h}$ を変更したとき, それに伴い, 他の勤務タイプの期間数を修正する必要があるが, その修正は, できる限り $y_{i,h}$ の直前, 直後の勤務タイプから順番に実行可能性を満たす範囲で修正していく.

### 3.3 交換近傍

この近傍は, パターン変更近傍で行った操作を 2 人の従業員のパターン $\boldsymbol{y_{i_1}}, \boldsymbol{y_{i_2}}$ に対して同時に行うことで得られる解の集合である. ただし以下の方法により近傍の削減を行う. 式 (1) で与えられる制約 $C$ のうち, その制約に関わる人数が一人だけであるもの $C_1 \subseteq C$ に注目する. すなわち, $|I_c| = 1, c \in C_1$ である. (実際, 多くの場合においてこのような制約は多数存在する. 例えば, 各従業員の全期間におけるある勤務タイプの必要期間数や, 各従業員があるタイプを 2 回行うのにその間に少なくとも何期間おかなければならない, といった制約が考えられる. ) まず, そのような制約の中から異なる従業員に関わるものを 2 つ選び, $c_1, c_2$ とする. $c_1, c_2$ に対し, その共通期間 $[t^-_{c_1}, t^+_{c_1}] \cap [t^-_{c_2}, t^+_{c_2}]$ の中で同じ期間に割当てられている両従業員のタイプを互いに交換することによって得られる解集合を考える. このような交換近傍内の解は前記 2 つの近傍内の解とは重複しない. この特徴により, 局所最適解における解の変形に交換近傍内の解を用いることで, 解を更新した後に元の解にすぐに戻ってしまうサイクリングを防ぐことが期待できる.

## 4 アルゴリズムの概要

提案手法では, 実行可能な初期解をランダムに生成し, そこから局所探索を開始する. 通常の局所探索には, パターン変更近傍のみを用いる. 局所最適解に陥った場合には, 期間変更近傍内の解, あるいは交換近傍内の解に移動する. これらの変形の性能比較は, 計算実験を通して行う.

## 5 計算実験

実験はワークステーション Sun Ultra 2 Model 2300 (300 MHz, 1 GB memory) 上で C 言語で行った. 局所探索法の終了条件は, 評価関数の値が 0 になるか, あらかじめ定めた CPU 時間が経過したときとする. 結果については発表当日に述べる.

## 参考文献

[1] 柳浦 睦憲, 茨木 俊秀, "組合せ最適化 — メタ戦略を中心として —", 朝倉書店, 2001