

列プール戦略を用いた時間空間ネットワークによる 大規模スタッフスケジューリングの高速化手法

2025年11月29日
氏名：[あなたの名前]

1. はじめに

本資料では、多様な勤務制約と個人の勤務趣向を厳密に考慮したスタッフスケジューリング問題に対し、列生成法（Column Generation）に基づく解法を提案する。特に、時間空間ネットワーク上の最短路探索に伴う計算負荷を軽減するため、過去に生成された列（シフトパターン）を保持・再利用する「列プール戦略」を導入し、大規模問題に対する実用的な計算効率を実現するアルゴリズムを構築する。

2. 数理モデル：主問題（RMP）

問題を Dantzig-Wolfe 分解し、グループ単位でのパターン選択を行う主問題（Restricted Master Problem: RMP）を定義する。

2.1. 記号の定義

- T : 時間帯集合 ($t \in T$)
- S : スキル集合 ($s \in S$) ← 追加
- K : 従業員グループ集合 ($k \in K$)
- h_{ks} : グループ k がスキル s を保有していれば 1, そうでなければ 0 ← 追加
- D_{ts} : 時間帯 t において必要なスキル s の人数 ← 変更
- Ω_k : グループ k の全実行可能シフトパターン集合
- $\Omega'_k (\subset \Omega_k)$: 現在 RMP に組み込まれている列の部分集合
- c_{kp} : パターン p のコスト（人件費+趣向ペナルティ）
- a_{tp} : パターン p の時間帯 t における勤務状態 (0/1)
- x_{kp} : パターン p の採用人数（決定変数）

2.2. 従業員グループの定義と分類基準

本モデルでは、計算規模を抑制しつつ個人の特性を反映させるため、従業員を共通の属性を持つ「グループ（タイプ） k 」に分類して扱う。グルーピングは以下の 2 つの基準に基づく。

- 保有スキル**: 各従業員が保有するスキルセット（例：レジ、調理、接客など）が一致する従業員同士を同一グループとする。パラメータ h_{ks} により、グループ k がスキル $s \in S$ を提供可能か定義される。
- 勤務嗜好（タイムウインドウ）**: 従業員のライフスタイル（朝型、夜型、短時間希望など）に基づき、勤務可能な時間帯や選好が類似する従業員をまとめる。これにより、グループごとに異なるネットワーク構造（勤務可能開始時刻や最大勤務時間）を適用することが可能となる。

2.3. 定式化

$$[\text{RMP}] \quad \min \quad Z = \sum_{k \in K} \sum_{p \in \Omega'_k} c_{kp} x_{kp} + \sum_{t \in T} \sum_{s \in S} M_{ts} \delta_{ts} \quad (1)$$

$$\text{s.t.} \quad \sum_{k \in K} h_{ks} \left(\sum_{p \in \Omega'_k} a_{tp} x_{kp} \right) + \delta_{ts} \geq D_{ts} \quad (\forall t \in T, \forall s \in S) \quad [\pi_{ts}] \quad (2)$$

$$\sum_{p \in \Omega'_k} x_{kp} \leq N_k \quad (\forall k \in K) \quad [\mu_k] \quad (3)$$

$$x_{kp} \geq 0, \quad \delta_{ts} \geq 0 \quad (4)$$

ここで、 δ_{ts} は不足人数を表すスラック変数、 $[\pi_{ts}], [\mu_k]$ は双対変数を表す。式 (2) は、各時間帯 t において、要求される各スキル s の需要 D_{ts} を満たすことを保証する制約である。

3. パターン生成とネットワークモデル

主問題を改善する新たな列は、以下の被約費用 \bar{c}_{kp} を最小化することで探索される。

$$\bar{c}_{kp} = c_{kp} - \sum_{t \in T} \left(\sum_{s \in S} \pi_{ts} h_{ks} \right) a_{tp} - \mu_k \quad (5)$$

3.1. グループごとの勤務制約の詳細定式化

各グループ k の実行可能パターン集合 Ω_k は、労働法規およびグループ特有の契約条件を満たす列のみで構成される。具体的には、ある列 p (ベクトル $a_p = \{a_{1p}, \dots, a_{Tp}\}$) が Ω_k に含まれるための条件は以下のように定式化される。

- a. **勤務時間枠制約:** グループ k の勤務可能開始時刻 T_k^{start} および終了時刻 T_k^{end} の範囲外では勤務しない。

$$a_{tp} = 0 \quad (\forall t \notin [T_k^{start}, T_k^{end}]) \quad (6)$$

- b. **最大・最小勤務時間制約:** 1 日あたりの総実労働時間は、所定の範囲 $[W_k^{\min}, W_k^{\max}]$ 内である。

$$W_k^{\min} \leq \sum_{t \in T} a_{tp} \cdot \Delta t \leq W_k^{\max} \quad (7)$$

ここで Δt は 1 タイムスロットの長さ (例: 0.25 時間) である。

- c. **連続勤務制約:** 一度勤務を開始したら、休憩を除き連続して勤務しなければならない (中抜け禁止)。これはネットワーク上のフロー保存則により保証される。
- d. **休憩取得規則:** 連続勤務時間が H_{break} を超える場合、少なくとも R_{min} 時間の休憩を付与しなければならない。

これらの制約は、時間空間ネットワーク G_k のノードおよびアーチの接続構造として埋め込まれるために、ネットワーク上のパス探索を行うだけで自動的に満たされる。

3.2. 時間空間ネットワークの構築

厳密なパターン生成のため、タイプ k ごとにグラフ $G_k = (V_k, A_k)$ を構築する。

- **ノード:** (t, l) (t : 時刻, l : 連続勤務時間)
- **アーチ:** 勤務継続、休憩、退勤などの遷移を表す。
- **アーチコスト:** $\tilde{c}_{uv} = (\text{本来のコスト}) - \pi_t$

このネットワーク上で Source から Sink への最短路を探索することで、最適な新規パターン p^* を生成できる。

4. 列プールによる計算効率化

ネットワーク G_k 上の最短路探索 (Exact Pricing) は、複雑な制約を網羅できる反面、反復ごとに実行するには計算コストが高い。そこで、生成された列を蓄積する「列プール (Column Pool)」を導入し、探索を階層化する。

4.1. 列プール \mathcal{P}_k の定義と運用

プール \mathcal{P}_k は、過去の反復で生成された全ての（あるいは有望な）パターン p を保持する集合である。

- **Heuristic Pricing (プール探索)** : グラフ探索を行わず、プール \mathcal{P}_k 内の既存パターンに対して最新の双対変数 π, μ を用いて被約費用 \bar{c}_{kp} を再計算する。これは単純なベクトル演算であるため極めて高速である。
- **Exact Pricing (グラフ探索)** : プール内に有効な ($\bar{c}_{kp} < 0$ となる) 列が存在しない場合に限り、ネットワーク G_k 上で最短路問題を解き、真に新しい列を生成する。

この戦略により、高コストなグラフ探索の実行回数を最小限に抑えつつ、列生成法の厳密性を維持することが可能となる。

5. フェーズ 2：個人割り当て (Rostering)

列生成完了後、得られた最適人数 x_{kp}^* (整数解) を基に、以下の割当問題を解き、個々の従業員 $e \in E_k$ を特定する。

$$\min \quad \sum_{e \in E_k} \sum_p \text{preference}_{ep} \cdot z_{ep} \quad (8)$$

$$\text{s.t.} \quad \sum_p z_{ep} = 1, \quad \sum_e z_{ep} = x_{kp}^* \quad (9)$$

6. 総合アルゴリズム

前述の全ての要素 (RMP, プール戦略, Aging, Phase 2) を統合した全体の処理フローを Algorithm 1 に示す。

Algorithm 1 Integrated Column Generation with Pooling Strategy

Require: Demand D_t , Employee Info K, E_k , Network G_k

Ensure: Individual Shift Schedule z_{ep}

1: === Phase 1: Pattern Optimization ===

2: Initialize Ω'_k with dummy columns.

3: Initialize Column Pool $\mathcal{P}_k \leftarrow \emptyset$ for all k .

4: **while** Optimality condition is not met **do**

5: **1. Solve RMP (Linear Relaxation)**

6: Solve linear program defined in Eq.(1)–(5).

7: Get dual variables π^*, μ^* .

8: **2. Column Management (Aging)**

9: Move non-basic columns from Ω'_k to \mathcal{P}_k if inactive for τ iterations.

10: **3. Pricing Step**

11: **for** each group $k \in K$ **do**

12: $found \leftarrow \text{FALSE}$

13: // (A) Heuristic Pricing: Check Pool

14: **for** $p \in \mathcal{P}_k$ **do**

15: Calculate $\bar{c}_{kp} = c_{kp} - \pi^* a_p - \mu_k^*$.

16: **if** $\bar{c}_{kp} < -\epsilon$ **then**

17: Add p to Ω'_k .

18: $found \leftarrow \text{TRUE}$

19: **end if**

20: **end for**

21: // (B) Exact Pricing: Solve Shortest Path

22: **if** $found$ is FALSE **then**

23: Update arc costs in G_k using π^* .

24: Solve Shortest Path Problem on G_k to get path p^* .

25: **if** $\bar{c}_{kp^*} < -\epsilon$ **then**

26: Add p^* to Ω'_k and \mathcal{P}_k .

27: **end if**

28: **end if**

29: **end for**

30: **if** No columns added in entire iteration **then**

31: **BREAK** (Global Optimum Reached)

32: **end if**

33: **end while**

34: **4. Integer Solution**

35: Solve RMP with integer constraints ($x_{kp} \in \mathbb{Z}_{\geq 0}$) using current Ω'_k .

36: Let x_{kp}^* be the optimal number of employees for pattern p .

37: === Phase 2: Individual Assignment ===

38: **for** each group $k \in K$ **do**

39: Solve Assignment Problem (Section 5) using x_{kp}^* .

40: Determine individual assignment z_{ep} .

41: **end for**

42: **return** z_{ep}