分離可能凸型コスト関数をもつプロジェクトスケジューリング問題

01405524 京都大学大学院情報学研究科 *野々部 宏司 NONOBE Koji
01001374 京都大学大学院情報学研究科 茨木 俊秀 IBARAKI Toshihide

1 はじめに

プロジェクトスケジューリング問題とは，与えられた先行制約の下，各作業の処理をいつ開始するかを決定する問題である．スケジュールの評価尺度としては，最大完了時刻や納期遅れ和など，いわゆる正規尺度 (regular measure) がよく用いられるが，現実には，重みつき納期ずれ時間和など非正規尺度が求められることも少なくない．そこで本研究では，それらを含む問題として**分離可能凸型コスト関数をもつプロジェクトスケジューリング問題** (Project Scheduling problem with a Separable Convex cost function, PSSC) をとり上げ，最適スケジュールを求める多項式時間アルゴリズムを提案する．

2 問題定義

$I=\{1,2,\ldots,n\}$ を作業集合とする．各作業 $i\in I$ は処理時間 p_i を持ち，処理を開始すると完了するまで中断することはできない．さらに，各作業 i に対して凸型コスト関数 c_i が与えられ，その開始時刻 x_i に応じて $c_i(x_i)$ のコストが発生する．また，P を作業間の先行関係を表す半順序とする．すなわち，$(i,j)\in P$ のとき，作業 i は作業 j に先行すると言い，i の処理が完了するまで j を開始することはできない．

PSSC は，先行関係 P により与えられる先行制約を満たした上で，総コストが最小となるよう，各作業 i の開始時刻 x_i を求める問題であり，以下のように定式化することができる．ただし，各 x_i は非負の整数値を取るものとし，各 p_i も非負整数であるとする：

$$
\begin{array}{lll}
\text{(PSSC)} & \min & \displaystyle\sum_i c_i(x_i) \\
& \text{s.t.} & x_j - x_i \ge p_i, \quad (i,j)\in P, \\
& & x_i : \text{非負の整数}, \quad i\in I.
\end{array}
$$

以下では，各 c_i は最小値 $\min_{x_i\ge 0} c_i(x_i)$ を持つと仮定する．また，各 x_i が整数値を取ることから，各 c_i は区分線形関数であると仮定し，その区分数を L_i で表す．

なお，各コスト関数 c_i が単調増加関数であるとき，または各作業に対して先行作業が高々一つであるとき，PSSC は効率良く解けることが知られている [1].

3 最適性条件

PSSC の最適性条件について述べる．PSSC の実行可能解 $\boldsymbol{x}$ に対して，有向グラフ $G(\boldsymbol{x})=(I,A(\boldsymbol{x}))$ を，

$$I=\{1,2,\ldots,n\},$$

$$A(\boldsymbol{x})=\{(i,j)\in P \mid x_j - x_i = p_i\}$$

と定義し，

$$d_i^-(x_i)=\begin{cases} -\infty, & x_i=0,\\ c_i(x_i)-c_i(x_i-1), & x_i=1,2,\ldots,\end{cases}$$

$$d_i^+(x_i)=c_i(x_i+1)-c_i(x_i), \quad x_i=0,1,\ldots$$

とする．PSSC の実行可能解 $\boldsymbol{x}$ が最適であるための必要十分条件は，

$$d_i^-(x_i)\le \sum_{(k,i)\in P} y_{k,i} - \sum_{(i,k)\in P} y_{i,k} \le d_i^+(x_i), \quad \text{for all } i,$$

$$x_j - x_i > p_i \Rightarrow y_{i,j}=0, \quad \text{for all } (i,j)\in P,$$

を満たす非負ベクトル $\boldsymbol{y}=\{y_{i,j}\ge 0 \mid (i,j)\in P\}$ が存在することであり，この条件を変形することにより以下の定理が得られる．

定理 3.1 PSSC の実行可能解 $\boldsymbol{x}$ が最適解となる必要十分条件は，有向グラフ $G(\boldsymbol{x})=(I,A(\boldsymbol{x}))$ の任意の閉包集合 (closure) $C\subseteq I$ に対して，

$$\sum_{i\in C} d_i^+(x_i)\ge 0, \quad \text{かつ} \tag{1}$$

$$\sum_{i\notin C} d_i^-(x_i)\le 0 \tag{2}$$

が成立することである．ただし，有向グラフ $G=(I,A)$ の閉包集合 C とは，

$$i\in C \Rightarrow j\in C, \quad \text{for all } (i,j)\in A$$

を満たす節点集合 $C\subseteq I$ のことである． □

4 アルゴリズム

本節では, 定理 3.1 に基づいた PSSC アルゴリズムを提案する. これは, 最適性条件 (2) を満たす実行可能解から始め, 実行可能性と条件 (2) を常に満たしつつ, 最終的には条件 (1) も満たすよう解を修正していくアルゴリズムである.

まず, 各作業 i の開始時刻 x_i を, 先行制約を満たす最早開始時刻に設定する. すなわち,

$$x_i := \max\{0, \max_{(k,i)\in P}\{x_k + p_k\}\}.$$

(これは, 作業集合 I の, 先行関係 P に対するトポロジカル整列 $\sigma_1, \sigma_2, \ldots, \sigma_n$, すなわち, $(\sigma_k, \sigma_\ell) \in P \Rightarrow k < \ell$ を満たす順列を求めることで計算できる.) このようにして得られる $\boldsymbol{x}$ は, 条件 (2) を満たす実行可能解である.

次に, $\boldsymbol{x}$ が条件 (1) を満たすかどうか判定する. これは, 最小重み閉包集合問題 (minimum weight closure problem) を解くことで判定できる. 最小重み閉包集合問題とは, 各節点 i に重み w_i が与えられた有向グラフ $G = (I, A)$ に対して, 重みの総和 $\sum_{i\in C} w_i$ が最小となる閉包集合 $C \subseteq I$ を求める問題であり, 最小カット問題に帰着することで解くことができる [2]. すなわち, 条件 (2) を満たす実行可能解 $\boldsymbol{x}$ に対して, 有向グラフ $G(\boldsymbol{x})$ の節点の重みを $w_i = d_i^+(x_i)$ $(i \in I)$ と定めるとき, 最小重み閉包集合 C^* が

$$\sum_{i\in C^*} d_i^+(x_i) \geq 0 \tag{3}$$

を満たせば, $\boldsymbol{x}$ は条件 (1) を満たし最適解となる. 一方, $\sum_{i\in C^*} d_i^+(x_i) < 0$ であれば, C^* に含まれる作業の開始時刻をまとめて遅らせることで, 実行可能性と条件 (2) を破ることなくコストを減少させることができる. したがって, この手順を繰り返すことで最適解を得ることができる.

以上をまとめると, PSSC を解くアルゴリズムは以下のようになる.

<u>**PSSC アルゴリズム**</u>

手順 0 (最早開始時刻)
各作業 i の開始時刻 x_i を最早開始時刻に設定する. $k := 1$ として手順 k-(1) へ.

手順 k-(1) (最小重み閉包集合)
節点の重み $w_i = d_i^+(x_i)$ $(i \in I)$ をもつ有向グラフ $G(\boldsymbol{x})$ に対して, 最小重み閉包集合 C^* を求める. C^* が (3) を満たせば終了. $\boldsymbol{x}$ は最適解. さもなくば手順 k-(2) へ.

手順 k-(2) (開始時刻増加)
各 $i \in C^*$ に対して

$$x_i := x_i + \min\{\delta_1, \delta_2\}$$

とした後, $k := k + 1$ として手順 k-(1) へ. ここで,

$$\delta_1 = \min\{x_j - x_i - p_i \mid (i,j) \in P, i \in C^*, j \notin C^*\},$$

$$\delta_2 = \min_{i\in C^*} \min\{\delta : 正整数 \mid d_i^+(x_i) \neq d_i^+(x_i + \delta)\}.$$

手順 k-(2) における δ_1, δ_2 は, 作業 $i \in C^*$ の開始時刻増加後, それぞれ実行可能性, 条件 (2) を満たすことを保証するためのものである.

なお, 手順 k の繰り返し回数は, 上記のアルゴリズムのままでは入力サイズの多項式では抑えられない. しかし, 手順 k-(1) で最小重み閉包集合が複数存在する場合, その中から常に節点数最小のものを選ぶことにすれば, $L = \sum_i L_i$ (すなわち, コスト関数 c_i の区分数の合計) として, 繰り返し回数は $O(L)$ となることが示せる.

定理 4.1 PSSC は, $O(L \cdot T(n, |P|))$ の計算手間で解くことができる. ここで, $L = \sum_i L_i$ はコスト関数 c_i の区分数の合計, $T(n, m)$ は, 節点数 n, 枝数 m の有向グラフの最小カットを求めるのに必要な計算手間である. □

5 実装

前節のアルゴリズムを実装する際, 以下の修正を行うことで, 最悪計算時間は変わらないが, 実際の計算時間を減らすことができる:

(i) 手順 k-(1) で最小重み閉包集合 C^* を求める際, 有向グラフ $G(\boldsymbol{x})$ を連結成分に分解し, それぞれに対して最小重み閉包集合問題を解く.

(ii) 手順 k-(2) において, 開始時刻 x_i $(i \in C^*)$ をすべて一律に増加させるのではなく, C^* を, C^* によって誘導される $G(\boldsymbol{x}) = (I, A(\boldsymbol{x}))$ の部分グラフ上の連結成分 $C_1^*, C_2^*, \ldots, C_H^*$ に分解し, 各 C_h^* $(h = 1, 2, \ldots, H)$ ごとに開始時刻 x_i $(i \in C_h^*)$ の増加量を定める.

参考文献

[1] V. Maniezzo and A. Mingozzi, "The project scheduling problem with irregular starting time costs", *Operations Research Letters* 25 (1999) 175–182.

[2] J.C. Picard and M. Queyranne, "Selected applications on minimum cuts in networks", *INFOR* 20 (1982) 394–422.