日本オペレーションズ・リサーチ学会
2019年 秋季研究発表会

直並列機械モデル上での線形時間衝突判定アルゴリズム

05000853　法政大学　*大塚帯紀　OTSUKA Taiki
01309090　法政大学　千葉英史　CHIBA Eishi

1. はじめに

製造ラインにおいて，商品の材料が衝突するとき，材料費，製造ラインの復旧の費用等の損失が発生する．そのため，与えられた製造ラインモデルに対して，衝突確率を求める手法は非常に重要である．

最近，直並列機械モデルに対して衝突確率を求める手法が提案された [1]．この手法では，各並列部に対して，ヒープデータ構造を使って衝突判定を行う．本研究では，各並列部に対して，より効率的な衝突判定法を与える．さらに，計算実験を通して，提案手法の計算時間に関する効果を確認する．

2. 直並列機械モデル

以下の変数を導入する．

1) s: ステージ数.
2) m_l: ステージ l 中の同一並列機械の台数.
3) $J_1, J_2, \ldots, J_n$: n 個のジョブ.
4) t_{tact}: タクトタイム，すなわち，ジョブ J_i と J_{i+1} を投入する時間間隔.
5) $T_i^{(l)}$: ステージ l でのジョブ J_i の処理時間.

n 個のジョブは，タクトタイム t_{tact} の間隔で $J_1, J_2, \ldots, J_n$ の順に入口からステージ 1 中の機械に投入され，ただちに処理がはじまる．ステージ l 中の機械で処理が完了したジョブは，ステージ $l+1$ 中の機械に運ばれて，ただちに処理がはじまる．ステージ s で処理されたジョブは，出口に運ばれる．簡単のため，搬送時間を考慮しない．処理時間 $T_i^{(l)}$ は確率変数であり，i と l に関して，互いに独立である．あるステージにて，全ての機械が処理中のときに，ジョブが到着したとしよう．これを，衝突現象と呼ぶ．

3. 衝突判定アルゴリズム

3.1. ステージ数が 1 のとき

ステージ数 $s = 1$，ステージ 1 中の機械数 m_1，ジョブ数 n，タクトタイム t_{tact}，ステージ 1 でのジョブ J_i の処理時間 $t_i^{(1)}$ が与えられるとき，衝突判定アルゴリズムを提案する．ジョブを投入するときに処理中の機械数に着目するのがキーである．以下にアルゴリズムを示す.

1. $p_i = 0 (i = 1, \ldots, n), i = 1$.
2. $p_i \geq m_1$ なら衝突が発生したとし処理を終える．それ以外なら 3 へ.
3. $i = n$ なら未衝突とし処理を終える．それ以外なら 4 へ.
4. $f = i + \lceil \frac{t_i^{(1)}}{t_{\mathrm{tact}}} \rceil$.
5. $f \leq n$ なら $p_f = p_f - 1$ とし 6 へ，それ以外なら 6 へ.
6. $p_{i+1} = p_{i+1} + p_i + 1, i = i + 1$ とし，$i \leq n$ なら 2 へ.

上記の衝突判定アルゴリズム 1 の時間計算量は $O(n)$ となる．そのため，提案アルゴリズムは既存アルゴリズム [1] より効率的である（既存アルゴリズムの時間計算量は $O(n \log m_1)$）.

3.2. ステージ数が 2 以上のとき

各ステージに対して，ジョブの到着時間あるいは完了時間に関してソート済みと仮定するとき，衝突判定アルゴリズムを提案する．

入力は，ステージ l 中の機械数 m_l，ジョブ数 n，タクトタイム t_{tact}，ステージ l でのジョブ J_i の処理時間 $t_i^{(l)}$ である．変数 st_i と ft_j は，それぞれジョブ J_i のステージへの到着時間，ステージでの完了時間である．また，ジョブの到着時間あるいは完了時間の順番付けは，順列，すなわち集合 $\{1, \ldots, n\}$ から自身への全単射 σ, π で表現し，$st_{\sigma(i)}, ft_{\pi(j)}$ は $st_{\sigma(i)} \leq st_{\sigma(i+1)}, ft_{\pi(j)} \leq ft_{\pi(j+1)}$ を満たすと仮定する．

1. $p_i = 0 (i = 1, \ldots, n), j = 1, i = 1$.
2. $p_i \geq m_l$ なら衝突が発生したとし処理を終える．それ以外なら 3 へ.
3. $i = n$ ならステージ l では未衝突とする，それ以外なら 4 へ.

4. $eft_{\pi(j)} \leq st_{\sigma(i)}$ かつ $j \leq n$ なら，$j = j + 1, p_{j+1} = p_{j+1} - 1$ をし 4 を繰り返す．それ以外は 5 へ．
5. $p_{i+1} = p_{i+1} + p_i + 1$.
6. $i = i + 1$ とし $i \leq n$ なら 2 へ．

上記の衝突判定アルゴリズム 2 の時間計算量は $O(n)$ となる．そのため，提案アルゴリズムは既存アルゴリズム [1] より効率的である（既存アルゴリズムの時間計算量は $O(n \log m_l)$）．

4. 数値結果

提案アルゴリズムの効果を確認するため，衝突確率を求めるアルゴリズムを実装した．正数 c を衝突確率の精度を表す変数として，衝突確率を求めるアルゴリズムの概要を以下に示す．

入力は，ステージ数 s，ステージ l 中の機械数 m_l，ジョブ数 n，タクトタイム t_{tact} である．

1. $loop = 1$.
2. 分布パラメータから処理時間 $t_i^{(l)}$ $(i = 1, 2, \ldots, n, l = 1, 2, \ldots, s)$ を生成する．
3. $s > 1$ なら，各ステージに対して，ジョブの処理終了時刻に関してソーティングを行い Step 4 へ．
4. 前章で与えたアルゴリズムを使って，各ステージに対して衝突の有無を判定する．$loop = loop + 1$．$loop \leq c$ ならば，Step 2 へ，そうでないならば，Step 5 へ．
5. 衝突確率の出力 (Step 4 で衝突が生じた回数 $/c$).

計算機環境は，以下の通り．CPU: Intel Core i7 1.90 GHz, RAM: 32.00 GB, OS: Windows 10 Professional．処理時間は期待値 100，分散 0.01 のアーラン分布に従うとして，$n = 3,000$，$c = 30,000$ とする．

$s = 1$，$m_1 = 10$ のときの計算結果を図 1 に示す．$s = 3$，$m_1 = m_2 = m_3 = 10$ のときの計算結果を図 2 に示す．タクトタイムは，衝突確率が，0 となる最大となる値から，衝突確率が，1 となる最小となる値まで入力した．また，衝突確率を求めまでのアルゴリズムの CPU 時間 10 回分の平均を平均 CPU 時間とした．どちらの結果も，提案アルゴリズムの方が高速であることを確認できる．

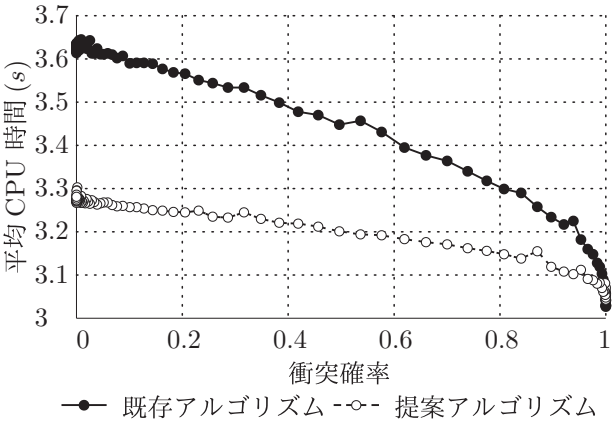

図 1: $s = 1$ の衝突確率に対する平均 CPU Time

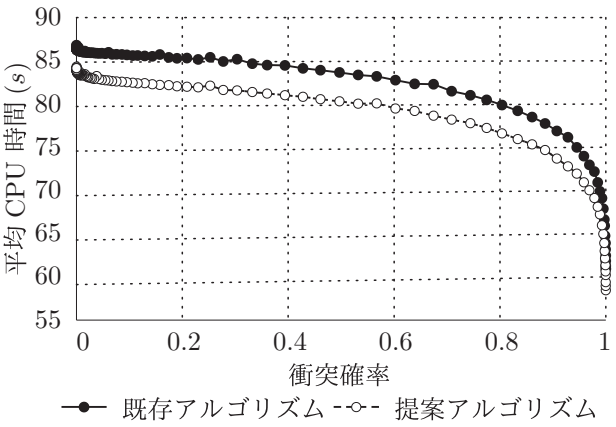

図 2: $s = 3$ の衝突確率に対する平均 CPU Time

5. おわりに

本研究では，各並列部に対して，線形時間で衝突を判定するアルゴリズムを提案した．また，計算実験では，提案アルゴリズムが既存アルゴリズムより高速となることを確認した．

さらに現実に即したシミュレーションをするために，バッファやジョブの移動時間を考慮したモデルに拡張することが，今後の課題になる．

参考文献

[1] 大塚帯紀, 千葉英史，直並列ライン上で衝突確率を求めるアルゴリズム，日本オペレーションズ・リサーチ学会 2017 年秋季研究発表会アブストラクト集，144-145，2017.