

# インプリサイスコンピュテーション手法に対する スケジューリングアルゴリズム

## Scheduling Algorithms for Imprecise Computations

小笠原 秀和

若林 真一

Hidekazu OGASAWARA

Shin'ichi WAKABAYASHI

広島大学 工学部

Faculty of Engineering, Hiroshima University

### 1. まえがき

ハードリアルタイムシステムにおけるタイミングフォールトを回避するアプローチとして、インプリサイスコンピュテーション手法が提案されている<sup>[1,4,5]</sup>。この手法においては、タスクは論理的に必須サブタスクと随意サブタスクの2つの部分に分解される。必須サブタスクは許容可能な質の結果を生み出すために必ずデッドラインまでに完了されなければならない部分であり、随意サブタスクは必須サブタスクの完了後に実行を開始し、必須サブタスクによって生み出された結果を改良する部分である。

スケジュール中の各随意サブタスクが、デッドラインまでに完了されているか、もしくは全く実行されていないかのどちらかであるとき、そのスケジュールは0/1制約を満たすという。Shihらは0/1制約をもつスケジューリング問題について考察し、0/1制約を満たし実行されなかった随意サブタスクの処理時間の和が最小となる最適スケジュールを求める問題がNP-困難であることを示した<sup>[5]</sup>。さらに、すべての随意サブタスクが同一の処理時間をもつという特別な場合に対し、単一プロセッサ上での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している。しかし、タスクが異なる重みをもつ場合については全く考察されていなかった。タスクの重みとはそのタスクの相対的重要度を示す。そこで、本稿ではタスクの重みが異なる場合における、単一プロセッサ上での0/1制約をもつスケジューリング問題について考察を行う。

### 2. スケジューリング問題

#### 2.1 問題の定式化

タスクの重みが異なる場合における単一プロセッサ上での0/1制約をもつスケジューリング問題を以下のように定式化する。なお、各タスク $T_i$ は必須サブタスク $M_i$ と随意サブタスク $O_i$ から構成され、 $O_i$ は $M_i$ の完了後に実行可能となるものとする。また、 $M_i$ 及び

$O_i$  の実行可能時刻, デッドラインは  $T_i$  のそれらと等しいとし,  $T_i$  の処理時間  $\tau_i$  は  $M_i$  の処理時間  $m_i$  と  $O_i$  の処理時間  $o_i$  の和であるとする. フィージブルなスケジュールとは, 以下に示す制約条件をすべて満たしたスケジュールのことをいう. 図 1 にスケジューリング問題 S I の入出力例を示す.

### [スケジューリング問題 S I]

入力: (1)先取り可能なタスク集合  $T = \{T_1, T_2, \dots, T_n\}$ .

各タスク  $T_i \in T$  は, 有理数である 4 つのパラメータ, 実行可能時刻  $r_i'$ , デッドライン  $d_i'$ , 処理時間  $\tau_i$ , 重み  $w_i$  をもつ.

(2)タスク間の先行関係  $\prec$ .

出力: 制約条件を満たし目的関数の値を最大にする  $T$  の単一プロセッサ上でのスケジュール  $S$ .

制約条件: (i)各タスクは実行可能時刻以降にスケジュールされる, (ii)タスク間の先行関係を満たす, (iii)すべての必須サブタスクは必ずデッドラインまでに完了される, (iv)スケジュールは 0/1 制約を満たす.

目的関数:  $P(S) = \sum_{i=1}^n U_i w_i$

$U_i$  は  $O_i$  がスケジュールされていれば 1, そうでなければ 0 であるとする.  $\square$

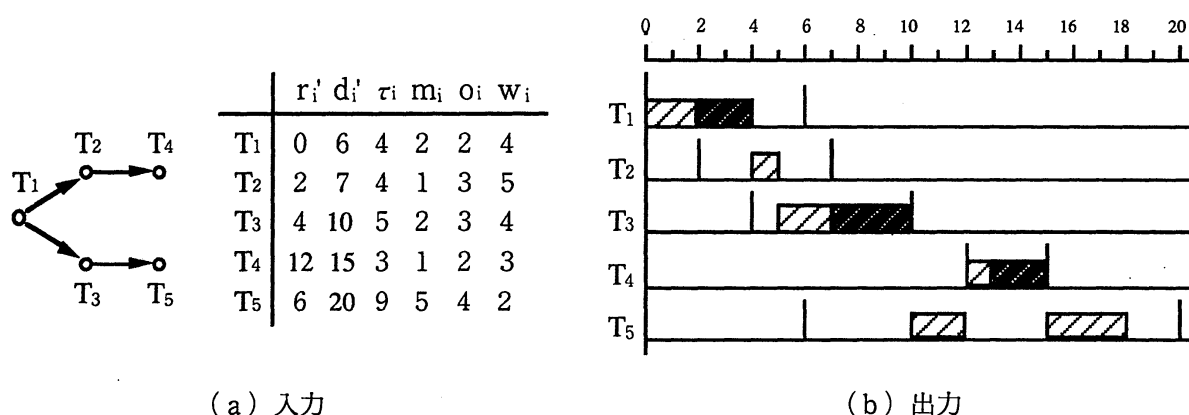


図 1 スケジューリング問題 S I の入出力例

## 2.2 実行可能時刻, デッドラインの変更

各タスク  $T_i$  の実行可能時刻  $r_i'$ , デッドライン  $d_i'$  を次のように変更する.  $r_i$ ,  $d_i$  はそれぞれ変更後の実行可能時刻, デッドラインを表す. また,  $A_i$ ,  $B_i$  はそれぞれ  $T_i$  より先に実行されなければならないタスクの集合, 後に実行されなければならないタスクの集合を表す.

$$r_i = \max \{ r_i', \max_{T_j \in A_i} \{ r_j' \} \} \quad d_i = \min \{ d_i', \min_{T_j \in B_i} \{ d_j' \} \}$$

変更した実行可能時刻，デッドラインをもつTのフィージブルなスケジュールが存在するとき，かつそのときに限り，与えられた実行可能時刻，デッドラインをもつTのフィージブルなスケジュールが存在する<sup>[3]</sup>。また，変更した実行可能時刻，デッドラインを用いることにより，一時的に先行関係を見捨てても後でフィージブルなスケジュールに変換することが可能となる。よって，以降では実行可能時刻，デッドラインは変更したそれらを意味するものとし，先行関係は考慮しない。

### 3. 問題 S I の部分問題の考察

問題 S I は NP-困難であるので，その部分問題について考察を行う。問題 S I は，すべてのタスクの実行可能時刻及びデッドラインが等しいという制約を与えたとしても，まだ NP-困難である。このことは，KNAPSACK問題<sup>[2]</sup>を本問題に多項式時間帰着することにより証明できる。以下では，すべてのタスクの実行可能時刻が等しい場合，実行可能時刻及びデッドラインが等しい場合について考察する。なお，

$$\begin{aligned} o_{\min} &= \min_{1 \leq i \leq n} \{ o_i \} & o_{\max} &= \max_{1 \leq j \leq n} \{ o_j \} \\ w_{\min} &= \min_{1 \leq i \leq n} \{ w_i \} & w_{\max} &= \max_{1 \leq j \leq n} \{ w_j \} \end{aligned}$$

とする。

#### 3.1 すべてのタスクの実行可能時刻が等しい場合

(1)  $o_i < o_j$  ならば  $w_i \geq w_j$  であるとき。

この問題を最適に解くアルゴリズム A S I を以下に示す。A S I では Earliest-Deadline-First アルゴリズム (以下 E D F と略す) を用いる。E D F は先取り可能で優先順位駆動であり，タスクの優先順位はそれらのデッドラインに従って割り当てられる。すなわち，より早いデッドラインをもつタスクほど高い優先順位をもつ。E D F の時間計算量は  $O(n \log n)$  であり，すべてのタスクがデッドラインに間に合うようなスケジュールが存在するならば，必ずそれを出力する<sup>[3]</sup>。なお，スケジュールがプリサイズであるとは，そのスケジュール中のすべてのタスクがデッドラインを満たして完了している場合をいう。

#### <アルゴリズム A S I>

Step1: E D F を用いて必須サブタスクの集合  $M = \{M_1, M_2, \dots, M_n\}$  のスケジュール  $S_m$  を求める。  $S_m$  がプリサイズでなければ終了；フィージブルなスケジュールは存在しない。

Step2: タスクのインデックスを  $o_1 \leq o_2 \leq \dots \leq o_n$  かつ  $w_1 \geq w_2 \geq \dots \geq w_n$  となるように付け替える.

Step3:  $i = 1$  とする.

Step4:  $M \cup \{O_i\}$  を EDF を用いてスケジューリングする. 得られたスケジュールがプリサイズであれば  $M = M \cup \{O_i\}$  とする.

Step5:  $i = n$  であれば  $M$  のスケジュールを EDF を用いて求め, その結果を出力する. そうでなければ  $i = i + 1$  とし, Step4へ.  $\square$

ASI では, EDF を用いて各随意サブタスクをスケジュールするかどうかを決定している. しかし, 実際には, Step1 で必須サブタスクの集合  $M$  を EDF を用いてスケジュールした後は, 時間計算量  $O(n)$  で各随意サブタスクのスケジュール可能性を調べることができる. よって, アルゴリズム ASI の時間計算量は  $O(n^2)$  である. 以下では ASI が最適スケジュールを出力することを証明する.

タスクは,  $o_1 \leq o_2 \leq \dots \leq o_n$  かつ  $w_1 \geq w_2 \geq \dots \geq w_n$  を満たすようにインデックスが付けられているものとする.  $T_i = \{M_1, M_2, \dots, M_{i-1}, T_i, T_{i+1}, \dots, T_n\}$  を最初の  $(i-1)$  個のタスクが随意サブタスクをもたないようなタスクの集合であるとする. このとき, 次の補題が成り立つ.

**【補題1】** タスク集合  $M \cup \{O_i\}$  のフィージブルでプリサイズなスケジュール  $S^i$  が存在するとき, かつそのときに限り,  $O_i$  がスケジュールされるような  $T_i$  の最適スケジュール  $S^i_0$  が存在する.

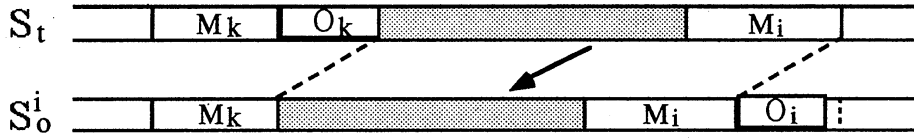
(証明) 明らかに, もし最適スケジュール  $S^i_0$  が存在するならば, タスク集合  $M \cup \{O_i\}$  はフィージブルでプリサイズなスケジュールをもつ.

集合  $M \cup \{O_i\}$  はフィージブルでプリサイズなスケジュール  $S^i$  をもつが,  $O_i$  がスケジュールされるような最適スケジュールは存在しないと仮定する. ここで,  $S_t$  を  $O_i$  がスケジュールされていないような最適スケジュールであるとする. 一般性を失うことなく,  $S_t$  は Earliest-Deadline-First スケジュールであるとする. この仮定に矛盾が生じることを, 以下の3つの場合に分けて考える.

(i)  $S_t$  において  $M_i$  以前にスケジュールされている随意サブタスクが存在する場合.

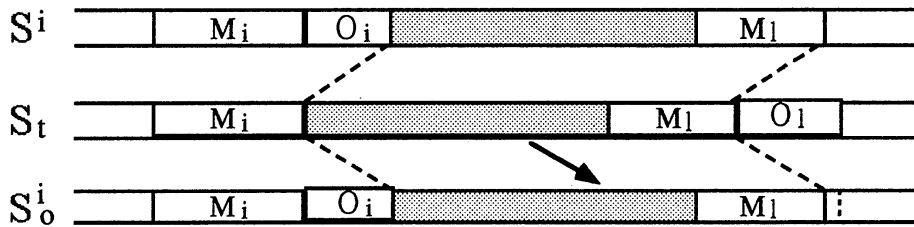
$S_t$  において,  $M_i$  以前にスケジュールされている随意サブタスク中の任意の1つを  $O_k$  とする.  $O_k$  の終了時刻から  $M_i$  の終了時刻までにスケジュールされているタスクを,  $O_k$  を取り除き,  $o_k$  時間だけ早い時刻に割り当てる. すべてのタスクの実行可能時刻が等しいため, この変更は矛盾なく行える.  $o_i \leq o_k$  であるから, これにより  $O_i$  がプロセッサに割り当てられたスケジュールを得ることができる(下図参照). また,  $w_i \geq w_k$  であるた

め、このスケジュールの目的関数の値は少なくとも  $S_t$  のそれに等しい。これは、 $O_i$  がスケジュールされるような最適スケジュール  $S_o^i$  が存在しないという仮定に矛盾する。



(ii)  $S_t$  において、 $M_i$  以前にスケジュールされている随意サブタスクは存在しないが、 $M_i$  以後にスケジュールされている随意サブタスクが存在する場合。

$S_t$  において、 $M_i$  以後で最も早い時刻に割り当てられている随意サブタスクを  $O_1$  とする。  $O_1$  以前にスケジュールされている随意サブタスクは存在せず、かつ  $M \cup \{O_i\}$  のフィージブルでプリサイスなスケジュールが存在することから、 $S_t$  中の  $M_i$  の終了時刻から  $M_1$  の終了時刻までにスケジュールされているタスクは、少なくとも  $o_i$  時間だけ遅い時刻に割当て可能であることがわかる。よって、 $o_i \leq o_1$  であることから、 $O_1$  を取り除きこれらのタスクを  $o_i$  時間だけ遅い時刻に割り当てることにより、 $O_i$  がプロセッサに割り当てられたスケジュールを得ることができる(下図参照)。この場合も、 $w_i \geq w_1$  であるためこのスケジュールの目的関数の値は少なくとも  $S_t$  のそれに等しい。これは、 $O_i$  がスケジュールされるような最適スケジュール  $S_o^i$  が存在しないという仮定に矛盾する。



(iii)  $S_t$  においてスケジュールされている随意サブタスクが存在しない場合。

この場合、 $M \cup \{O_i\}$  のフィージブルでプリサイスなスケジュールが最適となり、 $O_i$  がスケジュールされるような最適スケジュールが存在しないという仮定に矛盾する。

以上より、いずれの場合においても仮定に矛盾が生じる。よって、タスク集合  $M \cup \{O_i\}$  のフィージブルでプリサイスなスケジュール  $S^i$  が存在するとき、 $O_i$  がスケジュールされるような  $T_i$  の最適スケジュール  $S_o^i$  が存在するといえる。  $\square$

【定理 1】 アルゴリズム A S I は最適スケジュールを出力する。

(証明) タスクのインデックスに関する帰納法により証明する。  $M \cup \{O_1\}$  に対してフィージブルでプリサイスなスケジュールが存在すれば、 $O_1$  は補題 1 に従いスケジュール可

能となる。そうでなければ、 $O_1$ がスケジュールされるような $T$ の最適スケジュールは存在しないため、 $O_1$ は無視される。定理がある $j$ に対して真であると仮定する。 $O_{j+1}$ をスケジュールするかどうかの決定を行うときには、 $O_1, O_2, \dots, O_j$ については既に決定されている。このとき、 $ASI$ は $\{M_1', M_2', \dots, M_j', T_{j+1}, M_{j+2}, \dots, M_n\}$ に対するフィージブルでプリサイズなスケジュールが存在するかどうかを調べる。ここで、必須サブタスク $M_i'$  ( $i \leq j$ )は、 $O_i$ が無視されるとき $M_i$ 、スケジュールされるとき $T_i$ であるとする。そして、再び補題1を用いて $O_{j+1}$ がスケジュールされるべきか無視されるべきかを決定する。よって、 $j+1$ に対しても定理は真となる。□

(2)  $o_i = o_{\min}$ ならば $w_i = w_{\min}$ かつ $o_j = o_{\max}$ ならば $w_j = w_{\max}$ であるとき。

この問題をアルゴリズム $ASI$ を用いて解くとき、以下の定理が成り立つ。

【定理2】 $ASI$ を用いてこの問題を解いたときの目的関数の値を $P_a$ 、最適解における目的関数の値を $P_o$ とする。このとき、関係式 $P_a \geq (w_{\min}/w_{\max}) P_o$ が成り立つ。

(証明)  $ASI$ によってスケジュールされた任意サブタスクの数を $n_a$ 、最適スケジュール中の任意サブタスクの数を $n_o$ とする。すべてのタスクが同一の重みをもつ場合に $ASI$ が最適スケジュールを求めることは、(1)の関係が成り立つときに最適スケジュールを求めることから明らかである。すなわち、 $ASI$ はどんなスケジュールよりも多数の任意サブタスクを割り当てるスケジュールを求める。よって、 $n_a \geq n_o$ が成り立つ。また、 $ASI$ によるスケジュールがすべて最小処理時間の任意サブタスクで占められているとしたときの任意サブタスク数は、 $o_i = o_{\min}$ ならば $w_i = w_{\min}$ であるから、 $(P_a/w_{\min})$ となる。 $n_a$ がこれより大きな値をもつことはありえないから、 $(P_a/w_{\min}) \geq n_a$ が成り立つ。同様に、最適スケジュールがすべて最大処理時間の任意サブタスクで占められているとしたときの任意サブタスク数は $(P_o/w_{\max})$ となり、 $n_o \geq (P_o/w_{\max})$ が成り立つ。以上より、 $(P_a/w_{\min}) \geq n_a \geq n_o \geq (P_o/w_{\max})$ 。これより、 $P_a \geq (w_{\min}/w_{\max}) P_o$ を得る。□

### 3.2 すべてのタスクの実行可能時刻、デッドラインが等しい場合

すべてのタスクの実行可能時刻、デッドラインをそれぞれ $r, d$ と表す。3.1で示した(1)の関係が成り立つとき、この問題を時間計算量 $O(n \log n)$ で最適に解くアルゴリズム $ASIRD$ を以下に示す。

#### <アルゴリズムASIRD>

Step1: 全必須サブタスクの処理時間の和が $d-r$ より大きければ終了；フィージブルなスケジュールは存在しない。

Step2: タスクのインデックスを  $o_1 \leq o_2 \leq \dots \leq o_n$  かつ  $w_1 \geq w_2 \geq \dots \geq w_n$  となるように付け替える.

Step3:  $\sum_{i=1}^k o_i \leq d - r - \sum_{j=1}^n m_j$  を満たす最大の  $k$  の値を求める.

Step4:  $T_1, T_2, \dots, T_k, M_{k+1}, \dots, M_n$  をインデックス順にプロセッサに割り当てる.  $\square$

3.1 で示した (2) の関係が成り立つ場合に ASIRD を用いてこの問題を解いたとき, 得られた目的関数の値に関して定理 2 と同様の関係式が成り立つことを示すことができる.

#### 4. あとがき

本稿では, タスクの重みが異なる場合における, 単一プロセッサ上での 0/1 制約をもつスケジューリング問題 (問題 S I) を考察した. そして, その特別な場合に対し, 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し, その正当性の証明を与えた. さらに, 近似的に解くことができる部分問題を示した. 今後の課題としては,

- ・問題 S I の他の部分問題に対する考察,
- ・問題 S I に対する効率のよいヒューリスティックアルゴリズムの開発及び実験的評価等が挙げられる.

#### 文献

- [1] J.Y. Chung, J.W.S. Liw, and K.J. Lin: "Scheduling periodic jobs that allow imprecise results", IEEE Trans. on Comput., Vol.39, No.9, pp.1158-1173 (1990).
- [2] M.R. Garey and D.S. Johnson: "Computers and Intractability: A Guide to the Theory of NP-Completeness", Freeman (1979).
- [3] E.L. Lawler and J.M. Moore: "A functional equation and its application to resource allocation and sequencing problems", Management Sci., Vol.16, No.1, pp.77-84 (1970).
- [4] J.W.S. Liw, K.J. Lin, W.K. Shih, A.C. Yu, J.Y. Chung, and W. Zhao: "Algorithms for scheduling imprecise computations", IEEE Computer, Vol.25, No. 5, pp.58-68 (1991).
- [5] W.K. Shih, J.W.S. Liu, and J.Y. Chung: "Algorithms for scheduling imprecise computations with timing constraints", SIAM J. Comput., Vol.20, No.4, pp.537-552 (1991).