数理解析研究所講究録
64
第790巻 1992年 64-70

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コリズム

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[1,4,5]．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[5]．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τ_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ＳＩの入出力例を示す．

[スケジューリング問題ＳＩ]

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

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τ_i，重みw_iをもつ．

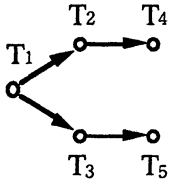
(2)タスク間の先行関係$<$.

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.

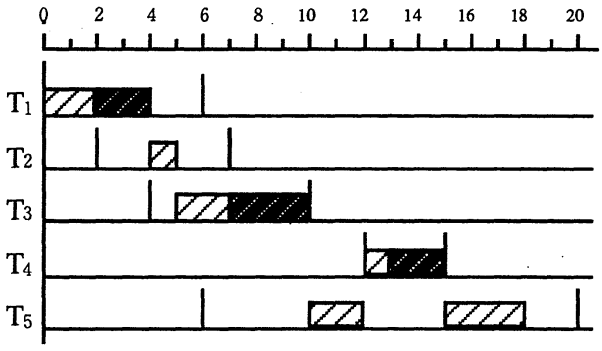
目的関数： $P(S)=\sum_{i=1}^{n} U_i w_i$

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．□



	r_i'	d_i'	τ_i	m_i	o_i	w_i
T_1	0	6	4	2	2	4
T_2	2	7	4	1	3	5
T_3	4	10	5	2	3	4
T_4	12	15	3	1	2	3
T_5	6	20	9	5	4	2

（a）入力



（b）出力

図1　スケジューリング問題ＳＩ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．

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

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[3]．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無視し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ＳＩの部分問題の考察

問題ＳＩ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ＳＩ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[2]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，

$$o_{min} = \min_{1 \le i \le n} \{ o_i \} \qquad o_{max} = \max_{1 \le j \le n} \{ o_j \}$$

$$w_{min} = \min_{1 \le i \le n} \{ w_i \} \qquad w_{max} = \max_{1 \le j \le n} \{ w_j \}$$

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$w_i \geqq w_j$であるとき．

この問題を最適に解くアルゴリズムＡＳＩを以下に示す．ＡＳＩではEarliest-Deadline-Firstアルゴリズム(以下ＥＤＦと略す)を用いる．ＥＤＦ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．ＥＤＦの時間計算量はO(nlogn)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[3]．なお，スケジュールがプリサイスであるとは，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ＡＳＩ＞

Step1：ＥＤＦを用いて必須サブタスクの集合$M = \{M_1, M_2, \cdots, M_n\}$のスケジュール$S_m$を求める．$S_m$がプリサイスでなければ終了；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$o_1 \leqq o_2 \leqq \cdots \leqq o_n$かつ$w_1 \geqq w_2 \geqq \cdots \geqq w_n$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スであれば$M = M \cup \{O_i\}$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．　□

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，$o_1 \leqq o_2 \leqq \cdots \leqq o_n$かつ$w_1 \geqq w_2 \geqq \cdots \geqq w_n$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．$T_i = \{M_1, M_2, \cdots, M_{i-1}, T_i, T_{i+1}, \cdots, T_n\}$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スなスケジュール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_oが存在する．

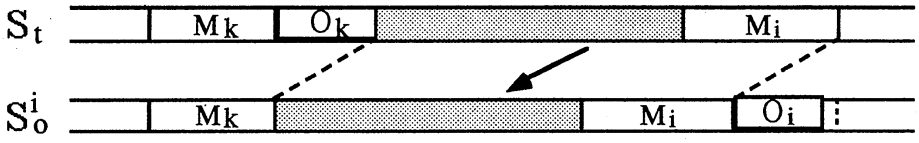
（証明）明らかに，もし最適スケジュールS^i_oが存在するならば，タスク集合$M \cup \{O_i\}$はフィージブルでプリサイスなスケジュールをもつ．

集合$M \cup \{O_i\}$はフィージブルでプリサイスなスケジュール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.

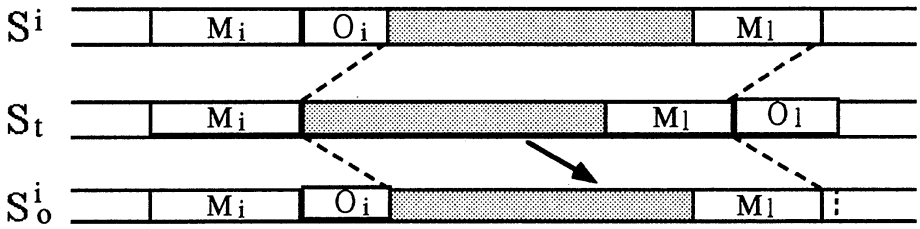
S_tにおいて，M_i以前にスケジュールされている随意サブタスク中の任意の１つをO_kとする．O_kの終了時刻からM_iの終了時刻までにスケジュールされているタスクを，O_kを取り除き，o_k時間だけ早い時刻に割り当てる．すべてのタスクの実行可能時刻が等しいため，この変更は矛盾なく行える．$o_i \leqq o_k$であるから，これによりO_iがプロセッサに割り当てられたスケジュールを得ることができる(下図参照)．また，$w_i \geqq w_k$であるた

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^i_o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$_l$とする．O$_l$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∪{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$_l$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$_i$≦o$_l$であることから，O$_l$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$_i$≧w$_l$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^i_o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∪{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∪{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^i_oが存在するといえる． □

【定理1】 アルゴリズムＡＳＩ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∪{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，…，O_jについては既に決定されている．このとき，ＡＳＩは$\{M_1', M_2', \cdots, M_j', T_{j+1}, M_{j+2}, \cdots, M_n\}$に対するフィージブルでプリサイス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'（$i \leqq j$）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$P_a \geqq (w_{min}/w_{max})\ P_o$が成り立つ．

（証明）ＡＳＩによってスケジュールされた随意サブタスクの数をn_a，最適スケジュール中の随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随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，$n_a \geqq n_o$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随意サブタスクで占められているとしたときの随意サブタスク数は，$o_i = o_{min}$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，$(P_a/w_{min}) \geqq n_a$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随意サブタスクで占められているとしたときの随意サブタスク数は(P_o/w_{max})となり，$n_o \geqq (P_o/w_{max})$が成り立つ．以上より，$(P_a/w_{min}) \geqq n_a \geqq n_o \geqq (P_o/w_{max})$．これより，$P_a \geqq (w_{min}/w_{max})\ P_o$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$o_1 \leqq o_2 \leqq \cdots \leqq o_n$かつ$w_1 \geqq w_2 \geqq \cdots \geqq w_n$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：$T_1, T_2, \cdots, T_k, M_{k+1}, \cdots, M_n$をインデックス順にプロセッサに割り当てる． □

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ＳＩ）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．さらには，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ＳＩの他の部分問題に対する考察，

・問題ＳＩ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).