7Q−1

生産計画問題におけるスケジューリング方式

湯上　伸弘　原　裕貴　吉田　裕之
(株)富士通研究所

1　はじめに

本稿では、半順序を利用したスケジューリング方式について議論する。本方式は、基本的には資源の競合が起こる場合に山崩しを効率的に行なうためのアルゴリズムである。

2　問題の定義

本稿で扱う問題は、取り扱う資源の集合と、その資源の量を変化させるイベントの集合と制約が与えられたときに、制約を満足するようなイベント間の時間的な順序関係を求めることである。ここで、production scheduling問題を例に取ると、資源とは、生産を行なうための機械や道具、人、原料、中間製品、製品などであり、イベントとは、個々のタスクの開始や終了、制約とは、タスクの納期や前後関係、タスクの処理時間などの時間制約と、機械の数や在庫量の範囲などの資源制約である。

時間制約は、イベントの発生時刻に関する制約で、イベントiの発生時刻をTiとしたとき、基本的には次の3種類の不等式の集合になる。

$$Ti \leqq Const.,\ Ti \geqq Const.,\ Ti - Tj \leqq Const.$$

実際には、上の不等式について、等号が含まれる場合とそうでない場合がある。

また、資源制約とは、各資源の量の下限値と上限値の集合であり、以下では、資源rの量の下限値をLr、上限値をUrとする。

3　半順序の制約充足判定

ここでは、ある半順序POが与えられたときPOが制約を満足するかどうかは、以下のようにして判定することができる。簡単のために、以下では、イベントiがイベントjよりも時間的に前であることを、i<jと書く。

3-1　時間制約

時間制約集合に半順序POを表現する不等式の集合

$$[\,Ti - Tj < 0 \mid i<j \in PO\,]$$

を加えた不等式集合を考え、これが解を持つかどうかを調べればよい。ここで現われる不等式は、全て線形不等式であるので、Sup-Inf法を使うことにより、調べることができる。また解がない場合、すなわち、半順序POが制約を満足しない場合POのどの部分が原因(矛盾原因)であるかを求めることできる[1]。

3-2　資源制約

半順序POのもとで、イベントiとの時間的な順序関係が未定であるようなイベントの集合をUKiとすると、イベントiの直前における資源rの量の下限値は、簡単な計算によって、次のように求めることができる。

$$Lir^{before} = Qinit\,r + \sum_{j<i} \Delta Qjr + \sum_{j \in UKi,\ \Delta Qjr<0} \Delta Qjr$$

$$= Qinit\,r + \sum_{Qjr<0, j \neq i} \Delta Qjr + \sum_{j<i,\ Qjr>0} \Delta Qjr - \sum_{i<j,\ Qjr<0} \Delta Qjr \quad (1)$$

この値は、半順序POでイベントiとの順序は未定であるイベントがいつ起ころうとも、イベントiの直前での資源rの量は、この値より小さくはならないという意味で、下限値である。

同様にして、イベントiの直前の資源量の上限値、直後の下限値、上限値は、以下のよう計算できる。

$$Uir^{before} = Qinit\,r + \sum_{Qjr>0, j \neq i} \Delta Qjr + \sum_{j<i,\ Qjr<0} \Delta Qjr - \sum_{i<j,\ Qjr>0} \Delta Qjr \quad (2)$$

$$Lir^{after} = Lir^{before} + \Delta Qir \quad (3)$$

$$Uir^{after} = Uir^{before} + \Delta Qir \quad (4)$$

これらの値を用いることで、半順序POを以下の3通りに分類できる。

(a) $\forall i,r$　$\min(Lir^{before}, Lir^{after}) \geqq Lr$
　　かつ　$\max(Uir^{before}, Uir^{after}) \leqq Ur$

(b) $\exists i,r$　$\max(Lir^{before}, Lir^{after}) > Ur$
　　または　$\min(Uir^{before}, Uir^{after}) < Lr$

(c) その他、すなわち(b)以外の場合で、
　$\exists i,r$　$\min(Lir^{before}, Lir^{after}) < Lr$
　　または　$\max(Uir^{before}, Uir^{after}) > Ur$

A Scheduling method for Production Scheduling Problems
Nobuhiro Yugami, Hirotaka Hara, Hiroyuki Yoshida
FUJITSU LABORATORIES LTD.

(a)の場合は、POによって決まっていな順序関係がどうなろうとも、全てのイベントの前後で全ての資源量が、制約として与えられた下限値と上限値の間にある、すなわち、全ての資源制約が満足される場合である。このとき、もし3-1で述べた時間制約が満足されるならば、POは解である。

(b)の場合は、逆に資源制約を満足しえない場合である。例えば、

$$\min(Lir^{before}, Lir^{after}) = Lir^{before} > Ur$$

である場合を考えると、POによって決まっていな順序関係がどうなろうとも、イベントiの直前で、資源rの量の下限値に関する制約を満足することができない。このとき、（1）式から明らかなように、その原因（矛盾原因）は、

$$(\wedge_{j<i,\ j\in PO,Qjr>0}\ j<i) \wedge (\wedge_{i<j,\ i<j\in PO,Qjr<0}\ i<j)$$

である。正確には、（1）式で和をとる際にLrを越える最小のイベント（j）の組について連言をとればよい。$Lir^{after}>Ur$, $Uir^{before}<Lr$, $Uir^{after}<Lr$の場合も同様である。

(c)の場合は、POで決まっていない順序関係がどうなるかによって、資源制約が満足されない可能性がある場合である。例えば、

$$\max(Lir^{before}, Lir^{after}) = Lir^{before} < Lr$$

であるならば、イベントiの直前で、資源rの量が、制約として与えられている下限値Lrを下回る可能性がある。このような可能性をなくすためには、iとの前後関係が決まっていないイベントとiとの前後関係を決めることにより、Lir^{before}を増やさなければならない。そのためには、（1）式からわかるように、rの変化量が正であるイベントjを選んでPOにj<iを加えるか、rの変化量が負であるイベントjを選んでPOにi<jを加えなければならない。すなわち、POのもとでは、

$$(\vee_{j\in UKi,Qjr>0}\ j<i) \vee (\vee_{j\in UKi,Qjr<0}\ i<j)$$

が真でなければならない。これを不十分条件と呼ぶ。この条件から、逆に矛盾条件を求めることができる。

$$(\wedge_{j<i,\ j<i\in PO,Qjr>0}\ j<i) \wedge (\wedge_{i<j,\ i<j\in PO,Qjr<0}\ i<j) \wedge (\wedge_{j\in UKi,Qjr>0}\ i<j) \wedge (\wedge_{j\in UKi,Qjr<0}\ j<i)$$

この条件が成り立つときのUir^{before}はPOのもとでのLir^{before}に一致する。すなわち、Lrよりも小さくなるから、資源制約を満足しえない。

$Lir^{after}<Lr$, $Uir^{before}>Ur$, $Uir^{after}>Ur$の場合も、同様にして不十分条件と矛盾条件を求めることができる。

4　探索方式

3節で説明したように、ある半順序を与えると、その半順序が制約を満足するかどうか、満足しない場合は、どこを直せばよいかを知ることができる。これを利用して、以下のようにDependency Directed Backtrackingを行なうことにより、全ての制約を満足する半順序を求めることができる。ただし、ある半順序に、イベント間の順序関係を追加したり、削除したりする際には注意が必要である。例えば、[1<2]に2<3を加える際には、2<3だけではなく、1<2と2<3から導かれる前後関係1<3も同時に追加する必要がある。また、[1<2,2<3,1<3]から1<3を削除するさいには、同時に1<2か2<3のどちらかを削除しなければ意味がない。

1　POを適当な半順序に初期設定する
2　POが制約を満足するかどうかを調べる
3　もし、全ての制約を満足するならば終了
4　制約を満足しえないならば、5へ
　資源制約を満足するには不十分なら7へ
5　矛盾条件を保存し、矛盾条件のなかから順序関係を1個（i<j）選択する。
6　POからi<jを削除するために、同時に削除しなければならない順序関係の集合ROを求め、POからi<jとROを削除し、2へ
7　矛盾条件を保存。不十分条件のなかから前後関係を一個（i<j）選択し、POとi<jから導かれる前後関係の集合AOを求める
8　PO∪{i<j}∪AOが既知の矛盾条件を含むなら、7へ、含まないなら9へ
9　PO←PO∪{i<j}∪AOとして2へ

5　考察

本手法では、制約を満足するための必要最低限な前後関係だけを、DDBを用いて求めようとするので、探索の効率化が期待できる。

参考文献

[1] 湯上、原、吉田　「スケジューリング問題へのATMSの適用」　情報処理学会研究会報告90-AI-72-1