■名古屋学芸大学　教養・学際編・研究紀要　第2号　2006年2月

拡張 RCPSP/τ における各種制約のリソース制約を用いた表現方法

Resource Constraints Expression Methods for Multiple Restrictions on Extended RCPSP/τ

堀尾　正典
Masanori HORIO

資源制約付きプロジェクトスケジューリング問題（RCPSP）において、作業の消費量が作業時間に依存して変化する問題は RCPSP/τと呼ばれている。このモデルは、RCPSP の中でも特に困難な問題と言われている。反面、このモデルは多くのスケジューリング問題を包含したフレームワークとして捉えることもできる。我々は、この RCPSP/τモデルをフレームワークとした汎用スケジューラを開発し、各種スケジューリング問題に適用･評価している。

今回の論文では、このような RCPSP/τに対して、いくつかの制約条件（リソース最小必要量の制約、一般化した順序制約、マルチモード制約）を追加した拡張モデルを提案する。これら拡張により、スケジューラの汎用性が向上することが期待できる。その上で、これら多様な制約条件を矩形パターンの包含関係と言う統一的な解法で解決することを試みる。このような統一的な解法は、汎用スケジューラの開発のコスト、効率的な探索アルゴリズム構築の上で重要なものとなる。

Key Word: プロジェクトスケジューリング、RCPSP、RCPSP/τ、汎用スケジューラ

1．はじめに

所与の先行関係が与えられた複数の作業からなるプロジェクトを考える。プロジェクトスケジューリング問題とは、プロジェクトの完了時刻や使用するコストの最小化などを目的として、プロジェクト内の各作業の実施時刻を決定する問題である。各作業の実施に対して、必要となるマンパワーや機械

などの資源（リソース）が制限なく使用できる問題では、PERT/CPM（Program Evaluation and Review Technique / Critical Path Method）と称される手法により最適解を求めることができる [4]。

しかし現実問題では、作業に対して利用できる資源は有限である場合が多い。これらリソースの使用制限が存在する問題は RCPSP（Resource − Constrained Project Scheduling Problem）と呼ばれている [1,15]。

RCPSP は、機械をリソースと捉えれば各種機械スケジューリング（ショップ問題）となる。それだけではなく、作業人数や機器の処理能力などのように単位時間の使用可能量に制限があるような制約や各種離接制約なども、すべてリソースとして表現することができるため、RCPSP は多くのスケジュール問題を含む枠組みとして考えられている [2,5]。実際、各種ショップ問題をはじめとするいくつかのスケジューリング問題が RCPSP の枠組みの中で論じられている [3,5,6,7,13,14]。その反面、RCPSP は大規模な問題に対して最適解を得ることは極めて難しい問題とされ、NP 困難に属することも知られている [1,15]。

RCPSP では、主に次3つのタイプの制約条件が存在する。

①リソースの使用可能量と作業のリソース必要量との間に存在するリソース制約

②各作業の実施順序の間に存在する順序制約

③作業における、使用リソースの排他関係を定義するモード制約

今までの研究において、これら制約条件を拡張・緩和されたさまざまな派生モデルが提案され [7,8]、その中のいくつかのモデルに対して効率的な解探索方法も研究されている [9,10,11]。このような拡張モデルの一つとして RCPSP/ τ が存在する。これは、リソースの使用可能量だけでなく、作業のリソース消費量が作業時間に応じて変化するモデルであり、Hartmann により称された [2]。標準的な RCPSP と比較して、リソース制約を拡張した本モデルでは表現できる制約条件がさらに増えるため、より広範囲なスケジュール問題を包括的に扱うことが可能になる。その反面、解法はより複雑となり、このモデルに対する研究は標準的なモデルと比較してあまりなされていない。

我々は、この RCPSP/τをフレームワークとして汎用性に主眼を置いたプロジェクトスケジューラを開発し、PSPLIB[12] や時間割作成問題、ナース・スケジューリング問題へ適用し、その処理能力や有用性の検証を行っている [13,14]。

この論文では、ソルバーの適用範囲をより広げるため、RCPSP/ τ の制約をさらに一般的に拡張したモデルを考える。しかし、制約条件が複雑になるほど効率的な探索アルゴリズムを持ったソルバーの開発は一層困難となる。各種制約条件に対して、それぞれ個別に対応するアルゴリズムを実装することは、プログラムそのものが複雑かつ長大となり、探索の効率化を考える場合に大きな障害となるからである。そこで、複雑な制約条件をより単純かつ統一的に表現する手法が重要となる。制約条件を単純で統一的なものとして考えることができれば、その制約条件に対する効率的探索アルゴリズムの検討もより容易なものとなる。当然システム構造も単純化し、システム開発のコストが低減する。そこで本論文では、拡張したRCPSP/τにおける複雑な制約条件をよりシンプルな表現方法（矩形パターンを用いた表現方法）で統一的に再構築することを考える。

以下2章では、RCPSPにおけるスケジューリング制約の定式化を行い、3章では矩形パターンを用いたリソース制約の表現方法を示す。この表現方法を用いて4章では順序制約を表現し、5章ではモード制約を示す。6章ではまとめを行う。

2. 本モデルのスケジューリング制約

2.1 RCPSP/τ の定式化

RCPSP/τモデルは、Hartmann の定式化を参考にすると、以下のように定義できる [2]。記号の定義は以下の通りである。

$\boldsymbol{J}$={0, 1, 2, 3,・・・, j,・・・・,J ,J +1}: 全ての作業の集合

$\boldsymbol{T}$={0, 1, 2, 3,・・・, t ,・・・T}: すべてのスケジュール時刻の集合

$\boldsymbol{K}$={0, 1, 2, 3,・・・, k ,・・・K}: リニューアル型リソースの集合

$\boldsymbol{P}_j$: 作業 j の先行作業の集合

p_j : 作業 j の長さ

$R_k(t)$: 時刻 t におけるリニューアル型リソース k の使用可能最大量

$r_{jk}(t)$: 作業 j の開始から t 時間経過後のリソース k の必要量

$x_{j,k}$ { 1：作業 j が時刻 t に終了する時
0：それ以外

これらの記号を用いると、

目的関数

$$\sum_{t=0}^{T} t \cdot x_{J+1,t} \quad \rightarrow \min \tag{2}$$

制約条件

$$\sum_{t=0}^{T} x_{j,t} = 1 \quad j \in \boldsymbol{J} \tag{3}$$

$$\sum_{t=0}^{T} t \cdot x_{h,t} \leq \sum_{t=0}^{T} (t - p_j) \cdot x_{j,t} \quad j \in \boldsymbol{J}, h \in \boldsymbol{P}_j \tag{4}$$

$$\sum_{j=1}^{J} \sum_{b=t}^{t+p_j-1} r_{jk}(t + p_j - b) \cdot x_{j,b} \leq R_k(t) \quad k \in \boldsymbol{K}, t \in \boldsymbol{T} \tag{5}$$

$$x_{j,t} \in \{0,1\} \quad j \in \boldsymbol{J}, t \in \boldsymbol{T} \tag{6}$$

となる。

2.2 リソース制約の拡張

リソース制約とは、その時刻で実施される作業のリソース必要量の合計がリソースの使用可能量を超えてはならないと言う制約であり、(5) 式がこれを表している。本論文では、この制約を「最大使用可能量の制約」と呼ぶ。

しかし現実のプロジェクトスケジューリング問題では、最大に使用できるリソース制約の外にも、最低限これだけのリソースが必要になるという制約が存在する場合が多い。看護師の勤務スケジュールを立案する NSP（Nurse Scheduling Problem）のような人員充当問題などに頻出する [14]。これは、ある日の勤務者が最大で L 人以下、最低 1 人以上必要となるような制約であり、本論分ではこれを「最小必要量制約」と呼ぶことにする。そこで、時刻 t におけるリニューアル型リソース k の必要最低量を $R'_k(t)$ として (5) 式を以下のように変更する。

$$R'_k(t) \le \sum_{j=1}^{J} \sum_{b=t}^{t+p_j-1} r_{jk}(t+p_j-b)\cdot x_{j,b} \le R_k(t) \quad k \in \boldsymbol{K}, t \in \boldsymbol{T} \tag{7}$$

$R'_k(t)$ が常に0である問題は、通常の RCPSP/τのリソース制約に一致する。

2.3 順序制約の拡張

順序制約は、各作業の実施順序の間に存在するもので、各作業には先行すべき作業が決められている。(4) 式がそれに該当する。しかし、(4) 式は先行作業が完了後、後続作業が開始可能という、極めて標準的な順序制約のみを表現している。実際のプロジェクトでは、コンクリートの乾燥などのように、先行作業の完了後一定の養生期間が必要となる場合がある。そこで本論分では、任意の二つの作業、j と h に対して事前に与えられた整数（負でも可）である時間制約数 $\lambda_{h,j}$ を導入し、(4) を以下のように拡張する。

$$\sum_{t=0}^{T} t\cdot x_{h,t} \le \sum_{t=0}^{T} (t-p_j)\cdot x_{j,t} + \lambda_{h,j} \quad j,h \in \boldsymbol{J} \tag{8}$$

このような数 λ を導入することにより、先行作業の終了後開始可能という順序制約だけでなく、先行作業の開始後開始可能、終了後終了、開始後終了などの各種順序制約が表現できることになる。これは次のように説明できる。

作業 h の開始時刻を s_h、後続作業 j の開始時刻を s_j とすると、

$$s_h + p_h = \sum_{t=0}^{T} t\cdot x_{h,t} \tag{9}$$

$$s_j = \sum_{t=0}^{T} (t-p_j)\cdot x_{j,t} \tag{10}$$

① h 終了後に j 開始可能な場合

$s_h + p_h = s_j$ であるから、 $\lambda_{h,j} = 0$ とすればよい。

② h 開始後に j 開始可能な場合

$s_h \leqq s_j$ であるから、(9)(10) より、

$$\sum_{t=0}^{T} t\cdot x_{h,t} - p_h \le \sum_{t=0}^{T} (t-p_j)\cdot x_{j,t} \tag{11}$$

$$\sum_{t=0}^{T} t \cdot x_{h,t} \leq \sum_{t=0}^{T} (t - p_j) \cdot x_{j,t} + p_h \tag{12}$$

よって、$\lambda_{h,j} = p_h$ とすればよいことになる。

③ h 終了後に j 終了可能な場合

$s_h + p_h \leqq s_j + p_j$ であるから同様に、$\lambda_{h,j} = p_j$ とすればよい。

④ h 開始後に j 終了可能な場合

$s_h \leqq s_j + p_j$ であるから同様に、$\lambda_{h,j} = p_j + p_h$ とすればよい。

また、j を左変に h を右辺にした式

$$\sum_{t=0}^{T} t \cdot x_{j,t} \leq \sum_{t=0}^{T} (t - p_h) \cdot x_{h,t} + \lambda_{j,h} \quad j,h \in \boldsymbol{J} \tag{13}$$

も可能となるため、時間枠のある問題も表現できる。

なぜならば、(8) より

$$s_h + p_h \leqq s_j + \lambda_{h,j} \tag{14}$$

(13) 式より、

$$s_j + p_j \leqq s_h + \lambda_{j,h} \tag{15}$$

((15) $-p_j + \lambda_{h,j}$) より、

$$s_j + \lambda_{h,j} \leqq s_h + \lambda_{j,h} - p_j + \lambda_{h,j} \tag{16}$$

従って (14)(16) より

$$s_h + p_h \leqq s_j + \lambda_{h,j} \leqq s_h + \lambda_{j,h} - p_j + \lambda_{h,j} \tag{17}$$

となるので、j の開始は、h の終了後、$-\lambda_{h,j}$ から $\lambda_{j,h} - p_j$ の時間枠の中で実行されなければならないことを意味することになるからである。

2.4 マルチモード制約

作業の実行において、必要となるリソースの間に排他的使用条件、すなわちリソース R_1 かリソース R_2 のどちらかが必要になるような問題が、マルチモード制約の付随した問題である。使用するリソースの組み合わせに対して、モード 1、モード 2、・・・のような場合分けがなされる。マルチモード制約を組み込んだ本モデルの定式化は以下の記号を用いて、次のようになる。

$\boldsymbol{J}$= {0, 1, 2, 3, ・ ・ ・, j , ・ ・ ・, J, J+1}: 全ての作業の集合
$\boldsymbol{T}$= {0, 1, 2, 3 , ・ ・ ・, t , ・ ・ ・T}: すべてのスケジュール時刻の集合
$\boldsymbol{K}$= {0, 1, 2, 3, ・ ・ ・, k , ・ ・ ・K}: リニューアル型リソースの集合
$\boldsymbol{N}$= {0, 1, 2, 3, ・ ・ ・, n , ・ ・ ・N}: ノンリニューアル型リソースの集合
$\boldsymbol{M}$= {0, 1, 2, 3, ・ ・ ・, m , ・ ・ ・M}: すべてのモードの集合
$p_{j,m}$: 作業 j のモード m における長さ
$R_k(t)$: 時刻 t におけるリニューアル型リソース k の使用可能最大量
$r_{j,m,k}(t)$:作業 j のモード m における開始から t 時間経過後のリソース k の必要量
A_n : ノンリニューアル型リソース n の総使用可能量
$a_{j,m,n}$: 作業 j のモード m おけるリソース n の総必要量
$x_{j,m,t}$ { 1: 作業 j のモード m で時刻 t に終了する時 / 0: それ以外

目的関数

$$\sum_{t=0}^{T} t \cdot x_{J+1,1,t} \rightarrow \min \tag{18}$$

制約条件

$$\sum_{m=1}^{M} \sum_{t=0}^{T} x_{j,m,t} = 1 \quad j \in \boldsymbol{J} \tag{19}$$

$$\sum_{m=1}^{M} \sum_{t=0}^{T} t \cdot x_{h,m,t} \le \sum_{m=1}^{M} \sum_{t=0}^{T} (t - p_{j,m}) \cdot x_{j,m,t} + \lambda_{h,j} \qquad j,h \in \boldsymbol{J} \tag{20}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M} \sum_{b=t}^{t+p_j-1} r_{j,k,m}(t + p_{j,m} - b) \cdot x_{j,m,b} \le R_k(t) \quad k \in \boldsymbol{K}, t \in \boldsymbol{T} \tag{21}$$

$$\sum_{j=1}^{J} \sum_{m=1}^{M} a_{j,m,n} \sum_{t=0}^{T} x_{j,m,t} \le A_n \quad j \in \boldsymbol{J}, m \in \boldsymbol{M}, n \in \boldsymbol{N}, t \in \boldsymbol{T} \tag{22}$$

$$x_{j,m,t} \in \{0,1\} \quad j \in \boldsymbol{J}, t \in \boldsymbol{T}, m \in \boldsymbol{M} \tag{23}$$

(22)式は総量のみ上限が設けられたリソース制約で、これを「ノンリニューアル型リソース制約（非再生型リソース制約）」と呼ぶ。これに対して、(21)式の様に単位時間の使用量に制限があるようなリソース制約を「リニューアル型リソース制約（再生型リソース制約）」と呼ぶ。ノンリニューアル型の制約はモード制約が規定された問題に付帯して発生することがある。

3. リソース制約の表現方法

3.1 リソース最大使用可能制約

リニューアル型のリソース量の最大使用可能制約は、次のような矩形のパターンの包含関係で表現ができる。

リソースの使用可能量については、横軸に、単位時間を一目盛りとするスケジューリング時刻を、リソースの使用可能量を縦軸にとった矩形パターンとして表現する。同様に作業のリソース必要量は、その作業の開始時刻を0とした相対時刻を横軸に、縦軸にその時刻における必要量をとった矩形パターンとして表現できる（図1）。

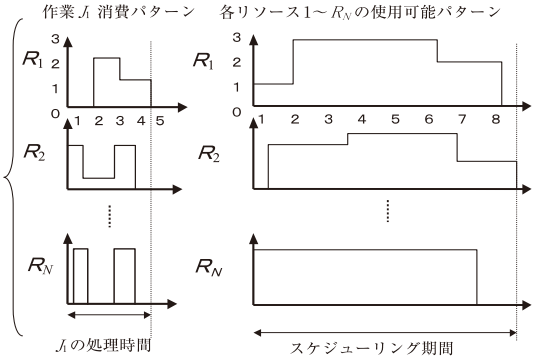

図1　消費量と使用可能量のパターン表現

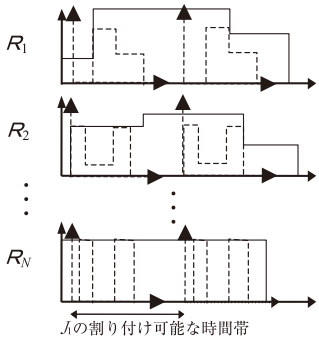

図2　作業の割り付け可能時間帯の算

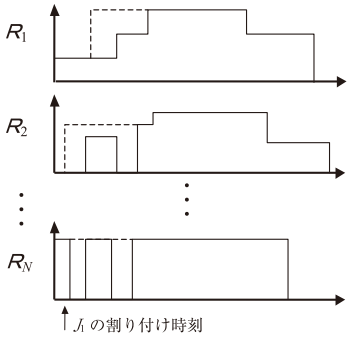

図3　新しいリソース状態の計算

リニューアルリソース制約を充足する時間帯は、使用可能パターンの中にすべての消費パターンが包含される（パッキング）時間帯である（図2）。

この時間帯の中の一つに作業を割付けたとする。リソースが消費されるため、割付け時刻を起点として、使用可能パターンから作業の消費パターンを抜き取る（パターンの差し引き）。これにより新しい使用可能パターンを算出する（図3）。これら作業を、すべての作業を割付できるまで繰り返すことにより、全作業のスケジューリングが可能となる。

3.2　リソース最小必要制約

リニューアル型のリソース量の最小必要制約を矩形パターンの包含関係で実現するため、次のような手法を考える。これは、最大使用可能量をチェックする仮想のパターンを用いて、スケジューリング状態が最小必要量を超えているかどうかを確認する手法である。

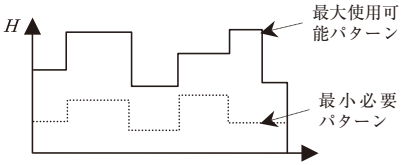

図4　最大使用・最小必要パターン

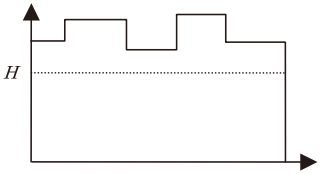

図5　高さチェック用仮想パターン

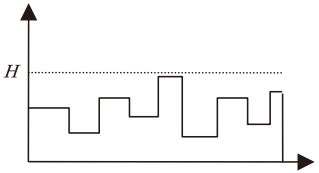

図6　スケジューリング結果パターン

まず、スケジュール期間 L、リソース R のスケジュール問題において、各時刻において R の最小必要量が事前に規定されているものとする。R の最大使用可能量（R の使用可能パターンの最高値）を H とする（図4）。この時、

①高さ H（H は十分な高さを確保するための値）、長さスケジュール期間 L の長方形パターンの上部に最小必要量の矩形パターンを結合した新パターンを作成する（図5）。

②パターン①に対して、一つの作業をスケジューリングした時点で、その消費量パターンを差し引いていく。

③スケジューリング終了時、パターン①の最大使用可能量が H 以下であるならば、すべての時刻で最小必要量を越えていることになるので、スケジューリングは完了（図6）。

ただし、高さが H 以下であることをチェックするためには、矩形パター

ンのパッキング手法だけでは困難である。作業をどこに割付ければよいかという目的評価関数が別に必要になる。

4. 順序制約のリソース制約表現

我々は、ナース・スケジューリング問題における特殊な並び制約を矩形パターンの包含関係を利用した手法で解決している [14]。これは、ある作業が n 回以上連続して出現した場合に限り、次の作業との間隔を m 時間以上あけると言うものであった。

ここでは , 先行作業と後続作業の順序関係を一般化した順序制約に対する矩形パターンの包含関係による表現を試みる。

スケジュール期間 L の問題において、次のような長さ$3L$ 高さ1の仮想使用可能パターンを、順序関係のチェック用として追加する。このパターンは順序関係のある作業 h と j において仮想の長方形消費パターン（それぞれ高さ1長さ L）により消費される（図7）。

作業 h は実際の消費パターン①と網掛けした仮想使用可能パターン②の二つのパターンに対し、それぞれの消費パターン③と⑤を包含チェックすれば、

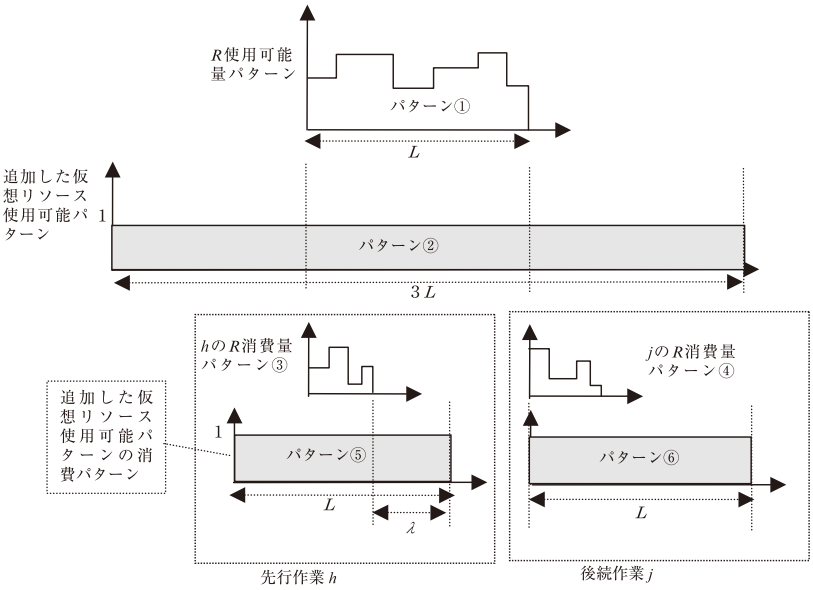

図7　一般化した順序関係をチェックする仮想パターン

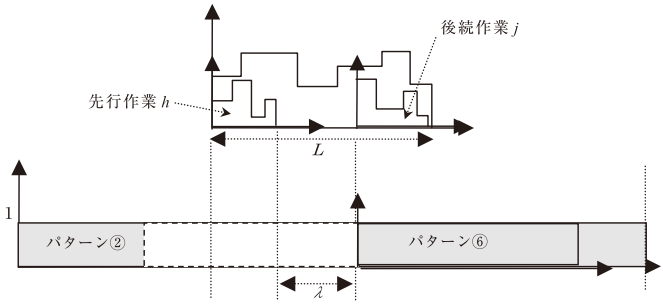

図8　後続作業のスケジューリング

h を割付けることが可能な時間帯が算出できる。追加した矩形パターンだけのパッキングでは、h は0から L の期間内に割付けることが可能であるため、追加したパターン制約が他の制約に影響を与えていないことは明らかである。

作業 h を割付けた時刻にパターン⑤がパターン②から抜き取られる。作業 j は仮想パターンがバッテイングするため（⑥が入らないため）、作業 h の終了後 λ 時間以降でないと割付けることができなくなる（図8）。

λ の値を2.3で示した値に変更すれば、一般化した順序関係が矩形パターンの包含関係で表現できる。

5. マルチモード制約のリソース制約表現

通常の RCPSP はシングルモードである。シングルモードでは、一つの作業の実行において、必要となる複数のリソースが同時に使用される。これに対してマルチモード問題は、一つの作業の実行において複数のリソースが排他的に利用される問題である。この排他関係のおのおのの状態がモードと呼ばれる。しかし各モードを、排他関係にあるリソース消費をもつ別々の作業と考えれば、マルチモード問題は、あるグループ内の作業に対する排他的割り当て条件が付随した問題と同値となる。

このようなマルチモード問題を矩形パターンの包含関係で表現するため次の様な状況を考える。今、作業 j がリソース R を消費する時、ふたつのモード M_1 と M_2 を持っていたとする。この時、M_1 と M_2 をそれぞれ別の作業と考え、どちらか一つを割り当てればよいという排他関係を設定した問題に置き換える。そこで、スケジュール期間 L の問題において、次のような長さ 2_L 高さ1の仮想使用可能パターンを、排他関係のチェック用として追加する（図9）。このパターンは排他関係のある作業 M_1 と M_2 において仮想の長方形消

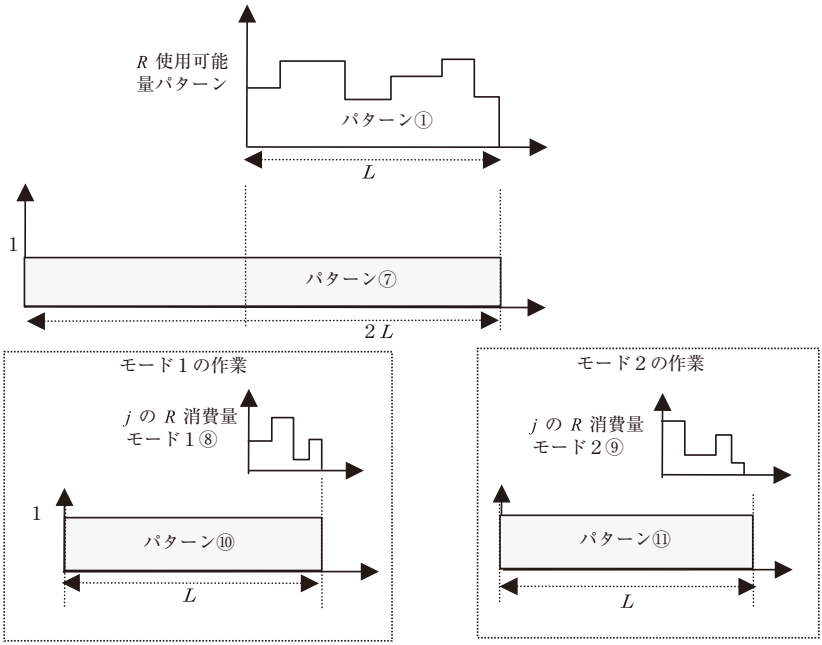

図9　排他関係をチェックする仮想パターン

費パターン（それぞれ高さ1長さ L）により消費される（図9）。M_1（もしくは M_2）が割り当てられれば、パターン⑦からパターン⑩（もしくは⑪）が抜き取られるため、M_2（もしくは M_1）は⑪（もしくは⑩）が⑦に対して割り当てできないため、スケジューリングできなくなる。

6. まとめ

本論分では RCPSP/τに対し、最小必要量制約、一般化された順序制約、マルチモード制約を追加した拡張モデルを提案した。このような拡張モデルでは表現できる制約の枠組みが広がるため、より広範囲な問題をとり扱えるようになることが期待できる。反面、本モデルをベースとした汎用ソルバーの開発は、これら多様な制約を満たすプログラムが複雑・長大とならざるを得ないため一層困難なものになる。このため、効率的な組み合わせ探索のアルゴリズムを構築することも難しくなるであろう。

この論文では、一つの作業に対して様々な仮想パターンを追加していくことにより、これら追加制約がリソース制約と同じものとして表現できること

を示した。本論分で提案したような手法であれば、制約計算は矩形パターンのパッキングであり、効率的な探索アルゴリズムを構築することとは、より目的評価関数を満たすようにして、いかに矩形パターンに矩形パターンをはめ込んでいくのかという問題に集約できることになる。そのため、システム開発もより単純化され、効率的な探索アルゴリズムの構築も期待できる。

一方幾つかの課題も明らかとなった。リソース最小必要制約は、矩形パターンのパッキングのみでは充足計算ができない。高さチェック用仮想パターンの残り状態を見て、高さが H を超える時刻に対して優先的に作業を割付けていく必要がある。割付け時刻の決定はスケジューリングの評価と関係してくる。通常、総所要時間最小化を目的とするスケジューリングでは、割付け時刻は最早可能時刻が選択されることが多い。このため、

①最早可能時刻

②高さチェック用仮想パターンの、高さ H 以下の時刻のように、複数のスケジューリング目的が生ずることになる。このような多目的なスケジューリング問題に対する効率的なアルゴリズを構築することが必要となる。

また、マルチモードに対しても、この算法では排他関係のある作業に対して、スケジューリング（パッキングによる割付け）を一度行い排他関係を確認している。しかし、作業の排他関係はスケジューリングの実行以前に判明していることであるため、このような作業は効率的とは言えない。この非効率性を上手く解消する算法も必要となる。

今後、これら課題に対処し本算法をシステム化し数値実験を実施していく。

引用文献

[1] Brucker,P. , Drexl,A. , Mohring,R. , Neumann,K. and Pesch,E.:“Resource-constrained project scheduling : Notation,classification, models, and methods”, *Eur. J.Oper.Res.*, Vol.112, pp.3-41（1999）

[2] Hartmann,S.: *Project Scheduling under Limited Resources*, Springer-Verlag Berlin Heidelberg（1999）

[3] Hartmann,S.:“Scheduling medical research experiments-An application of project scheduling methods”, *Manuskripte ausden Instituten fur Betriebswirtscaftslehre*, No.452, Universitat kiel, Germany（1997）

[4] 森雅夫他 :「オペレーションズリサーチⅡ」、朝倉書店、pp109-110（1989）

[5] 野々部宏司、茨木俊秀 :“汎用スケジューラー RCPSP によるアプローチー”, オペレーションズ・リサーチ , 第45巻第3号 , pp.10-16（2000）

[6] 野々部宏司、茨木俊秀 :”大きな構造物の生産スケジューリング問題に対する RCSP アプローチ”, スケジューリングシンポジウム99, 1B3（1999）

[7] Neumann,K. , Schwindt,C. and Zimmermann,J.: *Project Scheduling with Time Windows and Scarce Reosurces*, Springer-Verlag（2001）
[8] Weglarz,J.: *Project scheduling:recent models, algorithms, and applications*, Kluwer Academic Publishers（1999）
[9] Hartmann,S. and Kolisch,R.:“Experimental evaluation of state-of-the-art heuristics for the resource-constrained project scheduling problem”, *Eur.J.Oper.Res.*, Vol.127, pp394-407（2000）
[10] Nonobe,K. and Ibaraki,T.:“Formulation and tabu search algorithm for the resource constrained project scheduling problem（RCPSP）”, Technical Report#99010, *Department Applied Mathematics and Physics*, Kyoto University（1999）
[11] Reyck,B.D. and Herrolelen,W.:“A branch-and-bound procedure for the resource − constrained project scheduling problem with generalized precedence relations”, *Eur.J.Oper.Res.*, Vol.111, pp.152-174（1998）
[12] Kolisch,R. and Sprecher,A.:“PSPLIB-A project scheduling library”, *Eur.J.Oper. Res.*, Vol.96, pp.205－216（1997）
（http://www.bwl.uni − kiel.de/Prod/psplib/index.html）
[13] 堀尾正典、鈴木敦夫:“時間制約のある RCPSP/ τ を用いた汎用スケジューラの開発”、日本経営工学会誌 , Vol.54, No.3,（2003）
[14] 堀尾正典、鈴木敦夫:“RCPSP/τモデルを用いたナース・スケジューリング問題の解法”、南山経営研究 , Vol.17 No.3, pp.155-173（2003）
[15] Klein,R.: *Scheduling of Resource−Constrained Projects*, Kluwer Academic Publishers（2000）