(2)

ジョブショップ・スケジューリングの
モデル化と解法に関する研究

1993年2月

玉置　久

目 次

図目次

表目次

第 1 章

序 論

1.1 研究の背景と目的

近年における各種産業技術の進歩は，多様な工業製品の生産加工を可能にし，これが製品の多品種少量化という大きな流れを生じさせている．さらに，社会が豊かになるにつれて，他人と異なるものを持ちたいという欲望の高まりから生じる需要の多様化，そして技術進歩の加速による製品のライフ・サイクルの短縮が，この多品種少量化の傾向に拍車をかけている．

しかし，多品種少量生産は企業に対して大きな課題を与える．すなわち，企業間競争に打ち勝つためには，生産性の向上を図ることが 1 つの条件であるが，限られた機械，人員などの資源によって，それをいかに実現するか，つまり，いかに合理的で無駄のない生産スケジュールを作成し，生産が滞らかに進行するように管理するかという課題である．そのため，詳細な時間的日程を決定するスケジューリング，および円滑な物 (仕掛り品など) の流れに際して，緩衝の役割を果す在庫の管理のための実際的な手法の開発が強く望まれている．

従来から，生産システムにおけるこのようなスケジューリング問題に対して，モデル化および解法の研究が盛んであるが[1, 2, 3]，それらは，

(1) 理想問題 (狭義のスケジューリング問題)：
実際に生じるスケジューリングの問題を単純化し，理論的取り扱いを容易にした問題で，完了時刻などの評価基準 (目的関数) を最小あるいは最大にする最適スケジュールを論理的に求めることが目的とされる，あるいは，

(2) 現実問題 (広義のスケジューリング問題)：
実際に生じる問題の構成要素を (可能な限り) 考慮し，より現実に即した問題で，ある程度良いスケジュールを高速に求めることが目的となる，

のいずれかのクラスを対象としたものであり，理想問題のクラスと現実問題のクラスとの間には，図 1.1 に示すように大きな隔たりがあると考えられる．

さらに，スケジューリング問題の解法について考える場合，図 1.2 に示すように，大きく分けて下記の 3 種のアプローチがある．

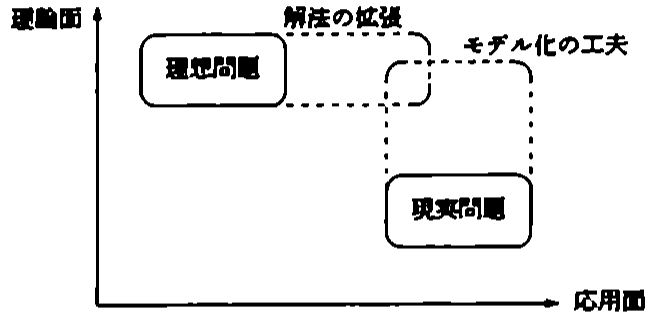


図 1.1 スケジューリング問題の捉え方

横軸(縦軸)は，右(上)にいくほど応用面(理論面)が強いことを表す．

(1) 列挙的手法:
すべてのスケジュールを調べ上げて(あるいは，これと等価なアルゴリズムによって)厳密な最適スケジュールを見出す．

(2) 発見的手法:
最適スケジュール，あるいは，ある程度良いスケジュールを生成するアルゴリズムを利用して，ただ1つのスケジュールを求める．

(3) 探索的手法:
列挙的手法と発見的手法の中間に位置する方法で，すべてのスケジュールからなる集合の部分集合内を探索することによって，その中から最も良いスケジュールを探し出す．

理想問題に対しては，数理的または発見的手法による最適解法の構成あるいは効率的な列挙的手法の構成が中心課題である．一方，現実問題に対しては，より良いスケジュールを求めるための発見的手法の経験的な探索を目的とすることが多い．

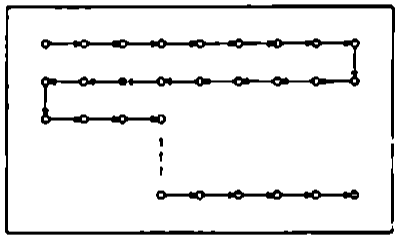
今後，スケジューリング問題の2クラス間の隔たりを小さくするため，

(a) 理想問題に対しては，準最適化に目的を緩和することによって適用範囲を広げ，多少複雑な問題をも取り扱えるようにして応用面の強化を図る，

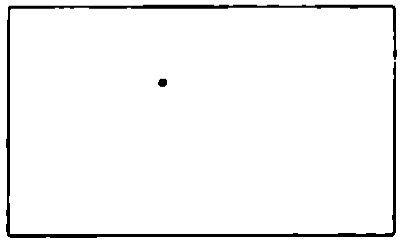
(b) 現実問題に対しては，モデル化の工夫やマンマシン・インタフェースの強化によって理想問題における成果を利用可能にし，理論面の強化を図る，

ことが肝要である(図1.1の破線の部分)．

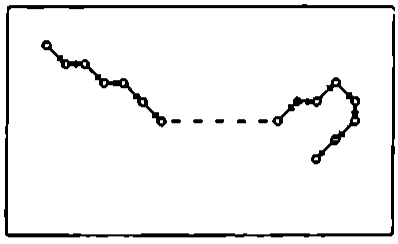
このような背景のもとに，本研究では，現実のスケジューリングにかなり広範な適用可能性を有する，ジョブショップ型の生産工程におけるスケジューリング問題(ジョブショップ・スケジューリング問題)を対象とし，まず，上述の(a)の立場から，仕掛り在庫設備(バッ



(a) 列挙的手法



(b) 発見的手法



(c) 探索的手法

図 1.2 解法の基本的考え方

枠がスケジュールの全体集合，丸が1つのスケジュールをそれぞれ表す．また，黒丸は最終的に解として選ばれるスケジュールを表す．

ファ) を考慮した問題を定義し，種々のモデル化手法を提案して，既存解法の適用を可能にすることを図る．この際，計算例を通して，モデルの特徴を比較するとともに，バッファがスケジュールに与える効果を調べる．次に，(b) の要求に対して，ジョブショップ・スケジューリング問題に立ち返って，実用的な解法として，分解による近似解法および部分的探索に基づく近似解法を提案し，計算例を通してこれらの解法の有効性を明らかにする．

1.2 論文の構成

前述のように，本論文ではジョブショップ・スケジューリング問題を対象とし，

(1) 問題のバッファを考慮した場合のモデル化手法，

(2) 分解による近似解法，

(3) 解空間の部分的探索に基づく近似解法

を提案し，計算例を通して，バッファのスケジュールに与える効果および近似解法の有効性を検討する．以下，本論文の構成について述べる．

第 2 章では，従来のスケジューリング理論で対象とされる問題を体系的に整理する．まず，機械の種類や特性，仕事に課せられる条件，およびスケジュールの評価基準に基づく問題の分類について述べる．これにより，本研究で対象とするジョブショップ・スケジューリング問題の位置付けを明確にしておく．次に，特定の問題について得られている最適解法 (アルゴリズム) を紹介した後，効率的なアルゴリズムが得られていないような問題を取り扱う場合に有効な組合せ最適化手法について述べる．

第 3 章では，本研究で取り上げるジョブショップ・スケジューリング問題の定義を記述するとともに，この問題に対する種々の解法 (最適解法および近似解法) を紹介する．

第 4 章では，バッファを考慮したジョブショップ・スケジューリング問題を定義し，混合整数計画問題への定式化を通して，選択グラフ，タイム・ペトリネットおよびガントチャートに基づく 3 種のモデル化手法を提案する．それぞれのモデルに対して，分枝限定法あるいはリスト・スケジューリング法に基づく解法を構成し，バッファがスケジュールに与える効果を調べるとともに，計算時間および得られるスケジュールの良さという観点から 3 種のモデルを比較する．

第 5 章では，ジョブショップ・スケジューリング問題に対する分解による近似解法を考案する．計算時間の短縮と計算精度の向上という相反する要求に答えるべく，問題の難易度を数量化し，部分問題の求解困難さをほぼ均一なものとするための分解アルゴリズムを提案する．

第 6 章では，ジョブショップ・スケジューリング問題に対して部分的探索に基づく近似解法を構成する．部分的探索法としては，生物の進化過程を模した遺伝アルゴリズム (Genetic Algorithm) に注目し，このアルゴリズムをジョブショップ・スケジューリング問題へ応用す

るための一手法を提案する．さらに，計算精度の向上および計算時間の短縮をともに実現することが可能な，遺伝アルゴリズムの近傍モデルを提案する．

最後に，第7章は本論文のまとめであり，本研究で得られた結論を整理するとともに，今後の課題について述べる．

第 2 章

スケジューリング問題

2.1 スケジューリング問題の定義

スケジューリング問題は，基本的に，1 つあるいは複数の「機械」(人，設備，機械などの総称) を用いて，いくつかの「仕事」(機械を用いて処理される対象のことで，複数台の機械による処理を必要とする場合，処理する機械ごとに「作業」に分割される) を処理するとき，各機械上での仕事 (あるいはその作業) の処理順序を決定すること，言い換えれば，各仕事 (作業) が処理される時間区間，すなわちスケジュールを決定することであると定義される[4, 5, 6]．この問題では，

- 各仕事は，2 つ以上の機械で同時に処理されることはない，
- 各機械は，2 つ以上の仕事を同時に処理することはない，

ということが基本条件である．

最適スケジュールとは，仕事間の先行関係や分割処理可能か否かなどの制約のもとで，すべての仕事を完了するまでに必要な時間 (最大完了時間) などの評価基準を最小 (あるいは最大) にするスケジュールのことである．また，得られたスケジュール，すなわち処理の時間的流れは，例 2.1 に示すように，縦軸に機械，横軸に時間をとってガント・チャートで表されることが多い (図 2.1)．

例 2.1 スケジューリング問題

表 2.1 で与えられる 2 機械フローショップ問題[1]において，各仕事は機械 1, 2 の順に処理されるものとする．このとき，全仕事の完了時間を最小にするスケジュールは図 2.1 のようになる． □

2.2 スケジューリング問題の分類

本節では，Lawler ら[1, 2]によるスケジューリング問題の分類を紹介する．

[1] フローショップ問題の定義については，2.2.2 参照．

表 2.1 2 機械フローショップ問題

	J_1	J_2	J_3	J_4	J_5
p_{j1}	10	9	4	12	7
p_{j2}	9	3	8	6	11

J は仕事を表し，p_{jk} は仕事 J_j の k 番目の作業の処理時間を表す．

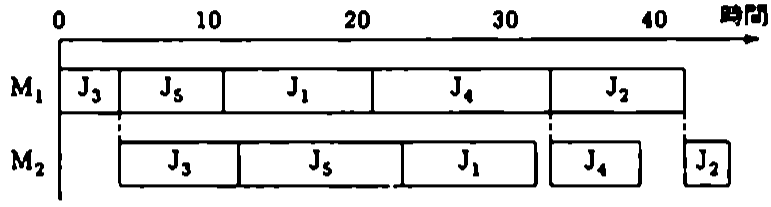


図 2.1 ガントチャート

J は仕事，M は機械を表す．

2.2.1 記号の定義

m 台の機械 M_i $(i = 1, \cdots, m)$ で n 個の仕事 J_j $(j = 1, \cdots, n)$ を処理するものとする．また，各仕事 J_j について，記号

n_j ：作業の数，
p_{jk} ：k 番目の作業の処理時間，
r_j ：準備完了時刻，
d_j ：納期，
w_j ：重み

を定義する．これらの記号を用いて，スケジューリング問題を 3 つの属性

α ：機械の種類と台数，
β ：仕事に関する制約条件，
γ ：スケジュールの良さを表す評価基準

によって分類し，$\alpha \mid \beta \mid \gamma$ と記す．

2.2.2 機械による分類 (α)

最初の項 α は機械の種類と台数を表す．まず，機械が 1 台のみの場合 ($\alpha = 1$) と，複数台ある場合の 2 通りに大別できる．複数台ある場合には，さらに並列機械型とショップ型とに分けられる．

並列機械型では，機械の機能がすべて同じであって仕事はどの機械で処理されてもよい (各仕事が単一の作業からなる)．この問題は，機械の性能 (処理速度) によって，さらに次の 3 つに分けられる．

(1) 等価機械型 ($\alpha = \mathrm{P}$) :
同じ性能の機械が複数台ある．

(2) 一様機械型 ($\alpha = \mathrm{Q}$) :
機械ごとに性能は異なるが，その性能の比 (仕事の処理時間の比) は仕事に関係なく一定である．

(3) 非一様機械型 ($\alpha = \mathrm{R}$) :
機械ごとに性能は異なり，その性能の比も仕事ごとに異なる．

一方，ショップ型では，各機械の機能がすべて異なる．仕事を処理するのに 1 台あるいは複数台の機械が必要となり (1 つの仕事が複数の作業に分割される)，機械の順序によって，さらに，

(1) フローショップ型 ($\alpha = \mathrm{F}$) :
すべての仕事で機械の順序が同一である，

表 2.2 機械の種類や台数による分類

α	分類	
1	1 機械型	
P	並列機械型	等価機械型
Q		一様機械型
R		非一様機械型
F	ショップ型	フローショップ型
J		ジョブショップ型
O		オープンショップ型

(2) ジョブショップ型 (α = J) :
機械の順序が仕事ごとに決まっている.

(3) オープンショップ型 (α = O) :
機械の順序がどの仕事においても自由である.

の 3 つに分けられる. 表 2.2 に α による分類をまとめておく.

さらに, 機械が複数台ある問題でその台数 m を定数 (特定の問題) として扱いたい場合, J3 のように, 種類を表す記号の後ろに自然数を付記することとする. 記号だけの場合は, m を変数 (一般的な問題) として扱うことになる.

2.2.3 仕事による分類 (β)

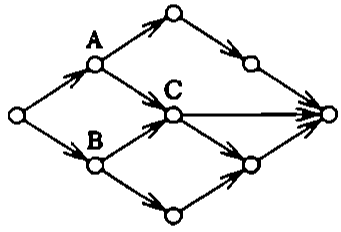
第 2 項 β は仕事に課せられる制約条件を示す.

(1) 先行関係制約 (prec, in-tree, out-tree) :
技術的な理由などにより, 仕事間に先行関係が存在する場合の制約を表す. この先行関係は一般に半順序をなし, 図 2.2 のような有向グラフで表される.

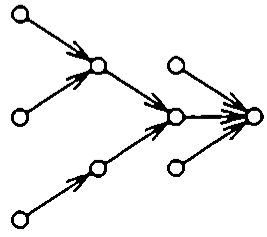
(2) 処理開始可能最早時刻制約 (r_j) :
スケジュール開始時点で, 処理を開始できない仕事が存在する場合の制約を表す.

(3) 作業数制約 ($n_j \leq \bar{n}$) :
ショップ型の場合, 各仕事の作業数に上限がある場合の制約を表す.

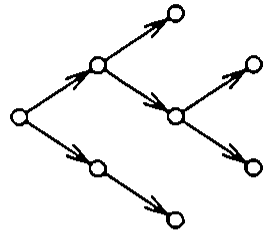
(4) 分割処理可能制約 (pmtn) :
原則として, 仕事 (以下, ショップ型の場合は作業) を分割して行うことはできないが, 仕事の処理を中断しても後で中断したところから再開できるとき, 分割処理可能であるという.



(a) 一般型 (prec)



(b) 入木型 (in-tree)



(c) 出木型 (out-tree)

図 2.2 仕事間の先行関係制約

各節点が仕事を表す．例えば (a) の A, B, C の部分は，仕事 A, B の両方が完了して初めて仕事 C に取りかかれることを示す．

(5) 単一処理時間制約 ($p_{jk} = 1$) :
すべての仕事に要する時間が等しいことを表す.

(6) 資源使用可能量制約 (res) :
仕事を行う際, 特に機械以外の資源が必要で, この資源の使用可能量に制限があることを表す.

2.2.4 スケジュール評価基準による分類 (γ)

評価基準 (目的関数) による分類 γ を次に示す. まず, スケジュールが求まると, 仕事 J_j ($j = 1, \cdots, n$) について,

完了時間 : C_j.
納期ずれ : $L_j = C_j - d_j$.
納期遅れ : $T_j = \max\{0, L_j\}$.
ペナルティ : $U_j = 0$ ($C_j \leq d_j$ の場合), あるいは 1 ($C_j > d_j$ の場合).

が計算できる. これらより, 最大値をとる評価尺度として,

(1) 最大完了時間 $C_{\max} = \max_j C_j$.

(2) 最大納期ずれ $L_{\max} = \max_j L_j$

が定義され, また, 重み付き和をとるものとして,

(3) 重みつき完了時間和 $\sum w_j C_j = \sum_{j=1}^{n} w_j C_j$.

(4) 重みつき遅れ和 $\sum w_j T_j = \sum_{j=1}^{n} w_j T_j$.

(5) 重みつき遅れ仕事数 $\sum w_j U_j = \sum_{j=1}^{n} w_j U_j$

が定義される. ただし, 全仕事の重み係数の値が等しい場合は w_j を省略し, $\sum C_j$ のように記す. ここで, 次のことが重要である.

- $\sum w_j C_j$ と $\sum w_j L_j$ ($= \sum_{j=1}^{n} w_j L_j$) との差は定数 $\sum w_j d_j$ であり, $\sum w_j C_j$ の最小化と $\sum w_j L_j$ の最小化は等価である.
- $L_{\max}$ を最小にするスケジュールは $T_{\max}$ ($= \max_j T_j$) および $U_{\max}$ ($= \max_j U_j$) をも最小にするので, 評価基準として $T_{\max}$ および $U_{\max}$ を考慮する必要がない.

例 2.2 分類例

(1) $1 \mid \mathrm{prec} \mid L_{\max}$:
一般型の先行関係を有する 1 機械型の問題において, 最大納期ずれを最小にする.

(2) R | pmtn, tree | $\sum C_j$:
木型 (入木型あるいは出木型) の先行関係を有する非一様機械型の問題で，分割処理を許しながら完了時間和を最小にする (仕事に複数の制約 β が課せられた問題の例)．

(3) J3 || $C_{\max}$:
3 機械ジョブショップ型の問題において，最大完了時間を最小にする (仕事に関する制約 β がない問題の例)． □

2.3 スケジューリング問題の解法

スケジューリングの目的は，制約条件 (α および β) を満足しながら，目的関数 (γ) の 1 つあるいは複数個を最小にするようなスケジュール (最適スケジュール) を求めることであるが，α, β および γ の組合せによって膨大な種類の問題ができるので，そのすべてを統一的に扱うことは不可能である．さらに，スケジューリング問題は本質的に組合せ最適化問題であり，その規模 (機械数 m や仕事数 n) が大きくなると，途端に求解困難になる．2.2 の分類に従って，スケジューリング問題におけるこの求解困難さ (複雑さ) の相互関係を図 2.3 に示す[1,2]．図中，A から B への矢印は問題 A が問題 B の特別な場合であることを示しており，問題 B は問題 A 以上に難しいといえる．

以下では，これらの問題のうち，発見的手法によって効率的に解かれているものについて，その代表的な解法 (アルゴリズム) を紹介する[1,2]．しかし，このような「効率的なアルゴリズム」がない問題では，組合せ最適化問題に対する一般的な解法を適用しなければならなくなる[1]．組合せ最適化手法については，2.4 に記述する．

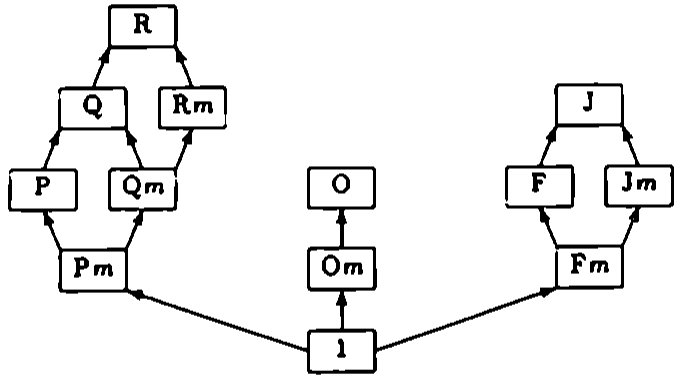
2.3.1 1 機械型の問題

機械が 1 台だけの場合には最大完了時間 $C_{\max}$ は一定なので，$\gamma = C_{\max}$ の問題は考察の対象とならない．$\gamma \neq C_{\max}$ の場合，各仕事 J_j ($j = 1, \cdots, n$) が納期 d_j をもつ場合に $L_{\max}$ を最小にする問題や，$\sum w_j C_j$ を最小にする問題などについては，以下に示すように，効率的なアルゴリズムが知られている．

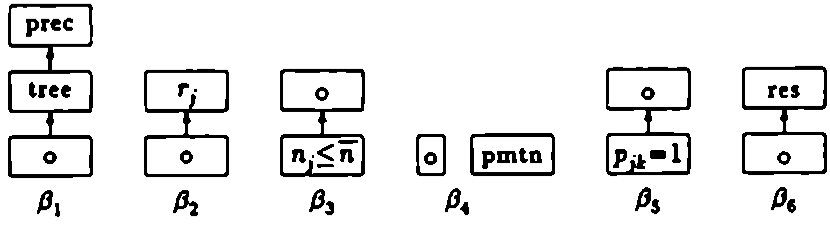
Jackson のアルゴリズム— 1 || $L_{\max}$

納期ずれを最小化する問題に対しては，n 個の仕事 J_j を納期 d_j の非減少順に並べると，最適スケジュールが得られる．しかし，この問題に $\beta = r_j$ の制約が付加されると，効率的なアルゴリズムは存在しなくなる．

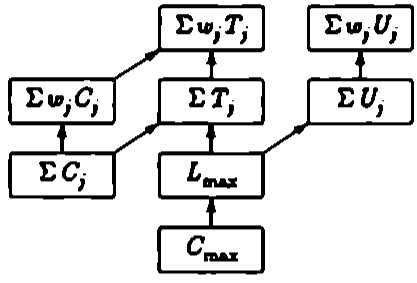
[1] 以下では，ある問題が NP 困難 (NP-hard)[7] であることがわかっている場合に，「効率的なアルゴリズムがない」と記すことにする．



(a) α (m : 自然数)



(b) β



(c) γ

図 2.3 スケジューリング問題の複雑さの関係

A から B への矢印は問題 A が問題 B の特別な場合である (問題 B は問題 A 以上に難しい) ことを示す．なお，o は対応する制約が課せられていないことを表す．

Smith のアルゴリズム— $1 \mid\mid \sum w_j C_j$

重みつき完了時間和を最小化する問題では，仕事 J_j を，比 w_j/p_{j1} の非増加順に並べると最適スケジュールが得られる．すべての仕事の重み w_j が等しい場合，このアルゴリズムは SPT (Shortest Processing Time first: 処理時間最小作業優先) 規則になる．この SPT 規則は，より複雑な問題に対してもよく用いられている規則で，経験的にかなり良いスケジュールが得られることがわかっている．

なお，$1 \mid\mid \sum w_j T_j$ の問題に対しては効率的なアルゴリズムが得られていないなど，1 機械型の問題といえども列挙的手法を必要とする問題も多い．

2.3.2 並列機械型の問題

本節では，並列機械型問題の中でも特に盛んな研究が行われている等価機械型の問題，すなわち n 個の仕事 J_j $(j = 1, \cdots, n)$ が同一の機能と性能をもつ m 台の機械 M_i $(i = 1, \cdots, m)$ のいずれかで処理されるという問題を取り上げる．この問題では，各仕事が単一の作業からなり，機械の性能がすべて同じであるので，仕事 J_j の処理時間を p_j と略記できる．

McNaughton のアルゴリズム— $P \mid \mathrm{pmtn} \mid C_{\max}$

まず，分割処理可能な等価機械型の問題を対象とする．このとき，最大完了時間の最適値 $C^*_{\max}$ は，

$$C^*_{\max} = \max \left\{ \max_j p_j, \frac{1}{m} \sum_{j=1}^{n} p_j \right\} \tag{2.1}$$

で与えられる．第 1 項は処理時間最大の仕事が最大完了時間を決定するために，機械に遊休時間が存在する場合に対応し，第 2 項はすべての機械を常に利用する場合に対応する．

この $C^*_{\max}$ を実現するスケジュールは次のようにして得られる．すなわち，任意の順序で仕事を機械に割り当てていき，時刻 $C^*_{\max}$ まで割り当てられたら仕事を分割して次の機械に移るという操作を，すべての仕事が割り当てられるまで繰り返す．得られたスケジュールにおいて，分割回数は多くとも $(m-1)$ であることがわかる．この分割数を最小にする問題を定義することもできるが，これに対する効率的なアルゴリズムは知られていない．

さらに，先行関係制約が加わった場合，$P \mid \mathrm{pmtn}, \mathrm{tree} \mid C_{\max}$ や $P2 \mid \mathrm{pmtn}, \mathrm{prec} \mid C_{\max}$ に対しては，効率的なアルゴリズムが得られている (アルゴリズムの記述は省略する) が，$P \mid \mathrm{pmtn}, \mathrm{prec}, p_j = 1 \mid C_{\max}$ になると，効率的に解くことができない．

Hu のアルゴリズム— $P \mid \mathrm{tree}, p_j = 1 \mid C_{\max}$

分割処理不可能な場合には，$P2 \mid\mid C_{\max}$ あるいは $P2 \mid\mid \sum w_j C_j$ に対してさえ，効率的なアルゴリズムは存在しない．そこで，効率的に解くことができる $P \mid \mathrm{tree}, p_j = 1 \mid C_{\max}$ の

問題を取り上げ，そのアルゴリズムを紹介する．

まず，木型の先行関係において，木の根に位置する仕事からある仕事 J_j に至るパス (唯一に定まる) に含まれる仕事数を，仕事 J_j のレベル l_j と定義し，すべての仕事のレベル l_j $(j = 1, \cdots, n)$ を求める．次に，処理開始可能な (先行仕事の処理時間区間が決定している) 仕事の中で，in-tree 型の場合はレベルの大きいものから順に，また out-tree 型の場合にはレベルの小さいものから順に空いている機械に割り当てていく．この操作をすべての仕事が割り当てられるまで繰り返すと，最適スケジュールが得られる．

この Hu のアルゴリズムにより，in-tree 型の問題に対しては $L_{\max}$ をも最小にできる．しかし，out-tree 型の問題では，この $L_{\max}$ 最小化に対する効率的なアルゴリズムは知られていない．

近似アルゴリズム— $P \mid\mid C_{\max}$

一般の等価機械型問題 $P \mid\mid C_{\max}$ に対しては，その最適スケジュールを求めるアルゴリズムは得られていないが，近似アルゴリズムは数多く提案されている．よく用いられるものに LPT (Longest Processing Time first: 処理時間最大作業優先) 規則がある．これは，処理時間の大きい作業から優先的に空いている機械に割り当てていく方法であるが，このとき得られるスケジュールの最大完了時間を $C_{\max}^{LPT}$，最適値を $C_{\max}^{*}$ とすると，

$$\frac{C_{\max}^{LPT}}{C_{\max}^{*}} \leq \frac{4}{3} - \frac{1}{3m} \tag{2.2}$$

であることがわかっている．

さらにこの問題は，機械を「箱 (bin)」，仕事を「品物」とみなすと，同じ容量の m 個の箱があるとき，n 個の品物を収容するのに必要な箱の最小容量およびそのときの品物の詰め方を決定することと解釈できる．各箱の容量が与えられているとして m を最小にする問題は箱詰め (bin packing) 問題と呼ばれ，種々の近似解法が提案されている．この箱詰め問題の近似アルゴリズムは $P \mid\mid C_{\max}$ に対してもよく応用されるが，特に，大きな品物 (処理時間の大きい仕事) から順に，それが入り得る最も番号の小さい箱 (機械) に詰めていくという FFD (First Fit Decreasing) 法がよく用いられる．

2.3.3 フローショップ型の問題

n 個の仕事が m 台の機械で順に処理され，処理する機械の順序がすべての仕事で同一であるとき，フローショップ型の問題 (以下，単にフローショップ問題) と呼ばれる．生産ラインの 1 つの工程 (流れ作業工程) などが，この問題の例として挙げられる．

Johnson のアルゴリズム— F2 $\|$ $C_{\max}$

n 個の仕事 J_j $(j = 1, \cdots, n)$ がいずれも 2 つの作業からなり，これらの作業がそれぞれ機械 M_1, M_2 でこの順に処理される場合の，最大完了時間を最小にするスケジュールは以下のようにして求められる．このとき，p_{ji} は仕事 J_j の機械 M_i 上での処理時間に対応する．

まず，n 個の仕事を次の 2 つの集合に分割する:

(1) S_1 : $p_{j1} \leq p_{j2}$ である仕事 J_j の集合．

(2) S_2 : $p_{j1} > p_{j2}$ である仕事 J_j の集合．

ここで，S_1 内の仕事 J_j を p_{j1} の非減少順に並べ，その後に S_2 内の仕事 J_j を p_{j2} の非増加順に作業を並べると，最適スケジュールが得られる．例 2.1 の最適スケジュールは，このアルゴリズムを用いて求められたものである．

非ボトルネックの 3 機械フローショップ問題，すなわち $\max_j p_{j2} \leq \max\{\min_j p_{j1}, \min_j p_{j3}\}$ という関係が満たされている問題に対しては，$(p_{j1} + p_{j2})$ と $(p_{j2} + p_{j3})$ をあらためて p_{j1} および p_{j2} として Johnson のアルゴリズムを適用すると，最適スケジュールが得られる．

しかし，3 機械以上の一般的なフローショップ問題に対しては，このような効率的なアルゴリズムは得られていない．よって，一般のフローショップ問題に対しては，分枝限定法などの列挙を基本とする方法を適用しなければならなくなる．分枝限定法の構成については，3.2.3 でジョブショップ型の問題に対する構成例を記述する．

2.3.4 ジョブショップ型の問題

ジョブショップ型の問題は，スケジューリング問題の中で最も広範な適用可能性を有するものであり，第 3 章で，この問題の定義および解法について詳しく説明する．

2.4 スケジューリング問題と組合せ最適化

2.3 で紹介した効率的な解法は，スケジューリング問題の中で極めて限られた問題に対してのみ実現可能なものである．このような効率的な解法がない問題では，最適な (あるいは最適に近い) スケジュールを求めるために，一般的な組合せ最適化手法に基づく解法を構成しなければならない．以下，組合せ最適化問題の定義およびその代表的な解法を示す[8, 9]．

2.4.1 組合せ最適化問題

組合せ最適化問題とは，組合せ的な制約条件の下である目的関数を最小化 (あるいは最大化) する数理計画問題であり，一般に次のように記述される[6]．

$$\min_x \quad f(x) \tag{2.3}$$

$$\text{subject to} \quad x \in \mathcal{F} \tag{2.4}$$

あるいは

$$\max_x \quad f(x) \tag{2.5}$$

$$\text{subject to} \quad x \in \mathcal{F} \tag{2.6}$$

ただし，$\mathcal{F}$ は，組合せ的条件を含むものである．特に $\mathcal{F}$ が整数条件を含む場合は，整数計画問題と呼ばれる．スケジューリング問題はこの組合せ最適化問題の典型例であり，ここで述べる組合せ最適化の手法は，スケジューリング問題に対する汎用的な解法となり得るものである．

可能領域が組合せ集合となる組合せ最適化では，実数集合を対象とした連続性や微分の概念に基づく古典的な最適化手法を直接利用することはできない．したがって，組合せ最適化問題の解法は，連続変数の最適化手法と本質的に異なるものとなり，一般に，解を数え上げるという列挙法的なアプローチとならざるを得ない．しかし，可能領域 $\mathcal{F}$ の要素 (組合せ) の総数は，有限ではあっても膨大な数にのぼることが多く，これらをすべて列挙するのは現実には不可能である．例えば，n 要素の順列の個数 $n!$ は，$n = 20$ 程度でも $n! \simeq 2.4 \times 10^{18}$ となる．そこで，実際に列挙する範囲をいかに限定するかが重要となる．

分枝限定法は，この目的に合致し，問題固有の構造を最大限利用して，不要な列挙を除外していく計算原理である．

2.4.2 分枝限定法

分枝限定法は，ほとんどすべての組合せ最適化問題に適用できる幅広い計算原理であって，特に効率の良い解法を可能にするような特殊な構造を持たない難しい問題に対して，最適解を求める唯一のアプローチとなっている場合が少なくない．したがってこの分枝限定法は，組合せ最適化問題に対する最も一般的な解法であるといえる．

分枝限定法の基本は，直接解くことが困難な問題 P_0 :

$$\min_x \quad f(x) \tag{2.7}$$

$$\text{subject to} \quad x \in \mathcal{F}_0 \tag{2.8}$$

が与えられたとき，適当な条件を設定して実行可能領域 $\mathcal{F}_0$ をいくつかの部分領域 $\mathcal{F}_k$ に分割し，それぞれに対応する問題 (部分問題) P_k :

$$\min_x \quad f(x) \tag{2.9}$$

$$\text{subject to} \quad x \in \mathcal{F}_k \tag{2.10}$$

のすべてを解くことによって，間接的にもとの問題を解こうとするものである．この一群の部分問題 P_k $(k = 1, 2, \cdots)$ を解いて得られた解の中で $f(x)$ の値が最小のものを見つけれ

ば，これがもとの問題 P_0 の最適解となる．部分問題 P_k を解くことがなお困難なときは，さらに $\mathcal{F}_k$ をいくつかの領域に分けて P_k を分解する．この手順を繰り返し適用すれば，いつかは直接解ける問題に到達し，それらの解を総合すればもとの問題 P_0 の最適解が求められる．これが，分枝限定法の基本的な考え方である．

ところで，計算効率の点からは，不必要な部分問題の生成はできるだけ抑えることが望ましく，

(1) どのように分解するか (分枝操作)，

(2) 部分問題をどの順序で調べるか (探索法)，

(3) 部分問題をどのように終端するか (限定操作)

が重要となる．

分枝操作

問題 P_k を直接解き難い場合，その実行可能領域 $\mathcal{F}_k$ を

$$\mathcal{F}_k = \bigcup_{\ell=1}^{q} \mathcal{F}_{k_\ell} \tag{2.11}$$

となるいくつかの部分集合 $\mathcal{F}_{k_1}, \cdots, \mathcal{F}_{k_q}$ に分割して，問題 P_k をいくつかの部分問題 P_{k_ℓ}：

$$\min_x \quad f(x) \tag{2.12}$$

$$\text{subject to} \quad x \in \mathcal{F}_{k_\ell} \tag{2.13}$$

で置き換えるのが分枝操作である．この過程は図 2.4 の探索図によって表現することができる．

限定操作

前述のように，不必要な部分問題の生成はできるだけ抑えることが望ましい．この目的で，ある部分問題 P_k の最適解が求まる場合 (図 2.4 の二重丸) や，その部分問題から原問題 P_0 の最適解が得られないことが何らかの理由によって結論できる場合 (図の黒丸) には，ただちにそれらを終端して以後の考察から除く．これが限定操作である．問題 P_k を直接解くことが難しくてもその最適値の近似値や下界値 (例 2.3 参照) などが得られれば，それらは分解を制御するための有効な情報となるので，限定操作によく利用されている．

さらに，ある時点で分解も終端もされていない部分問題 (図の白丸) は活性であるといい，いずれ分枝操作か限定操作が加えられる．

例 2.3 下界値

整数計画問題：

$$\min_x \quad z = \sum_{j=1}^{n} c_j x_j \tag{2.14}$$

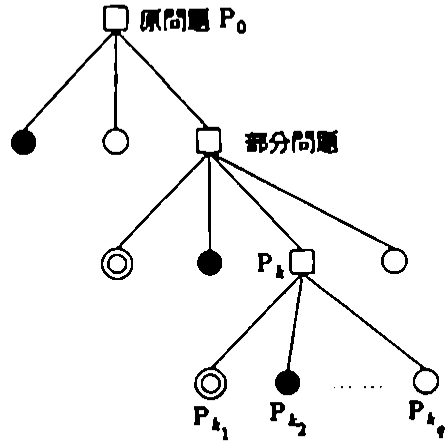


図 2.4 分枝限定法の探索図

●：枝通された部分問題，□：分解された部分問題，
◎：最適解の求まった部分問題，○：活性な部分問題．

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij}x_j \le b_i \qquad (i = 1, \cdots, m) \tag{2.15}$$

$$x_j \in \{0, 1\} \qquad (j = 1, \cdots, n) \tag{2.16}$$

に対する下界値として，整数条件を緩和して得られる線形計画問題：

$$\min_{x} \quad z = \sum_{j=1}^{n} c_j x_j \tag{2.17}$$

$$\text{subject to} \quad \sum_{j=1}^{n} a_{ij}x_j \le b_i \qquad (i = 1, \cdots, m) \tag{2.18}$$

$$0 \le x_j \le 1 \qquad (j = 1, \cdots, n) \tag{2.19}$$

の最適値が一般的に用いられる． □

探索法

活性な部分問題の集合からどの部分問題 P_k を選んでテストするか (探索法) は，求解に必要な計算量に大きな影響を与える．以下に，代表的なものを列挙するが，それぞれ特徴をもっているので，計算目標および使用計算機に応じて適当な探索法を選ぶことになる．

(1) 発見的探索：
各活性部分問題 P_k に対して f^* (f の最適値) の推定値 h を求めておき，h が最小となる部分問題を選ぶ．

(2) 最良下界探索：
各活性部分問題 P_k に対して下界値 g が最小となる部分問題を選ぶ．この方法によると，計算終了までに生成される部分問題の個数は少なくなる (すなわち計算時間が短くなる) が，必要な記憶容量は多くなる．

(3) 深さ優先探索：
活性な部分問題の内で深さ最大のものから，h あるいは g を基準に選ぶ．この方法は最良下界探索と対照的であり，必要な記憶容量は少なくなる．

(4) 幅優先探索：
活性な部分問題の内で深さ最小のものから，h あるいは g を基準に選ぶ．この方法では，直接列挙法的な性質が強くなり，特別な例題を除いてあまり用いられない．

分枝限定法の一般的手順

分枝限定法の全体の手順を以下に示す[9]．ただし，

P_0 ：与えられた問題，

$f^*(P_k)$：部分問題 P_k の最適値，

$g(P_k)$: $f(P_k)$ の下界値,

z : 暫定値 (探索過程において得られている f の最良値),

A : 活性な部分問題の集合

とする.

1° (初期設定) $A = \{P_0\}$, $z = \infty$ とする.

2° (探索) $A = \phi$ ならば 7° へ. $A \neq \phi$ ならば $P_k \in A$ を選び 3° へ.

3° (暫定値改良) $f^*(P_k)$ が得られているならば, $z = \min\{z, f^*(P_k)\}$ として 6° へ.

4° (限定操作) $g(P_k) \geq z$ ならば 6° へ.

5° (分枝操作) P_k から $P_{k_1}, P_{k_2}, \cdots, P_{k_q}$ を生成し, $A = A \cup \{P_{k_1}, P_{k_2}, \cdots, P_{k_q}\} - \{P_k\}$ として 2° へ.

6° (終端) $A = A - \{P_k\}$ として 2° へ.

7° (停止) 計算終了. z は P_0 の最適値 $f^*(P_0)$ を与える. $z = \infty$ ならば, P_0 は可能解をもたない.

このアルゴリズムは最適値 $f(P_0)$ のみを求めるものであるが, 最適解 (の 1 つあるいはすべて) を求める手順を付け加えることも容易にできる.

2.4.3 部分的探索による近似解法

2.4.2 の分枝限定法により組合せ最適化問題の最適解が求められるが, 実行可能解のすべてを調べ尽くすのと等価な探索を行うため, 一般に膨大な計算量を必要とする. そこで, 実行可能解の一部を効果的に探索することにより, ある程度よい解をある程度高速に求めようとする近似解法 (2.2 の分類では探索的手法に属す) のうち, 最近注目されている 2 つの方法を紹介する. なお以下では, 問題 (2.3) および (2.4) において制約条件 (2.4) がない組合せ最適化問題 P :

$$\min_x \quad f(x) \tag{2.20}$$

を取り扱う. 制約がある場合でも, 適当な重みを付けてペナルティとして目的関数に加えることにより, この式 (2.20) の形で記述できる.

シミュレーテッド・アニーリング法

問題 P に対し, 最適化の確率モデル P_p :

$$\min_p \quad p(x_i) f(x_i) \tag{2.21}$$

$$\text{subject to} \quad -\sum_{x_i} p(x_i) \log p(x_i) = H \tag{2.22}$$

$$\sum_{x_i} p(x_i) = 1 \tag{2.23}$$

$$p(x_i) \geq 0 \tag{2.24}$$

を考える[1]．ただし，$p(x_i)$ は 1 つの解 x_i に対する確率 (重み)，H は確率 $p(x)$ のエントロピー (ばらつきの程度) を表す．この問題 P_p は，一定のばらつきのもとで目的関数 $f(x)$ の平均値を最小にするものであり，その解は Lagrange 未定乗数法により，

$$p(x_i) = \frac{1}{Z} \exp\left\{-\frac{f(x_i)}{T}\right\} \tag{2.25}$$

$$Z(T) = \sum_{x_i} \exp\left\{-\frac{f(x_i)}{T}\right\} \tag{2.26}$$

として求められる．ここで，T は式 (2.22) に対応する Lagrange 未定乗数で，$f(x)$ をエネルギー関数とする物理システムとの対応から温度と呼ばれる．T が大きいときは $f(x)$ の最小点を中心にして広範囲に確率が広がり，T が小さくなるほど最小点近傍に確率が集中する．

エントロピーが一定 (温度 T が一定) のもとでの確率化された問題 P_p の解は，式 (2.25)，(2.26) より直接求められるが，非現実的であるので，ここでは式 (2.25)，(2.26) を定常確率分布としてもつような確率過程を，あるサンプルについてシミュレートする．このとき，温度 T を 0 に漸近させることにより，本来の目的関数 (2.20) に一致させることができる．

温度 T を大きい値から小さくしていくことによって，粗い探索から局所的探索へ移るとみなせるが，大切なのは T の変え方にあって，うまく変化させないと局所最小点にとどまりうる．そこで，物質の低温実験において，最終的に秩序立った結晶状態を作り出すための焼き鈍し (annealing) 法の温度制御から，確率モデルにおける大域的最小点に達する T の制御を類推した方法をシミュレーテッド・アニーリング (Simulated Annealing: SA) 法という[10, 11]．SA 法のアルゴリズムを以下に示す．

1° 初期解を定め，温度 T を初期設定する．

2° 冷却状態になっていれば 6° へ．なっていなければ 3° へ．

3° 解の分布が定常状態になっていれば 5° へ．なっていなければ 4° へ．

4° ランダムに解を変化させ，目的関数値の増加分 Δ を計算する．$\Delta \geq 0$ ならば新しい解を受理する．$\Delta < 0$ ならば確率 $p_a(\Delta)$ で受理する (図 2.5 参照)．3° へ戻る．

5° 次の温度 T を選び 3° へ．

6° 終了．

温度 T の変え方に関しては，種々の方法が提案されている[12, 13]．例えば，Kirkpatrick ら[11] では，定数 T_0 および α $(0 < \alpha < 1)$ を用いて，k 回目の繰返しにおける温度 $T(k)$ を，

$$T(k) = \alpha^k T_0 \tag{2.27}$$

[1] $H = 0$ とした場合，式 (2.21) が最小となるような重み関数 $p(x)$ は，式 (2.20) の $f(x)$ の最小値を与える．

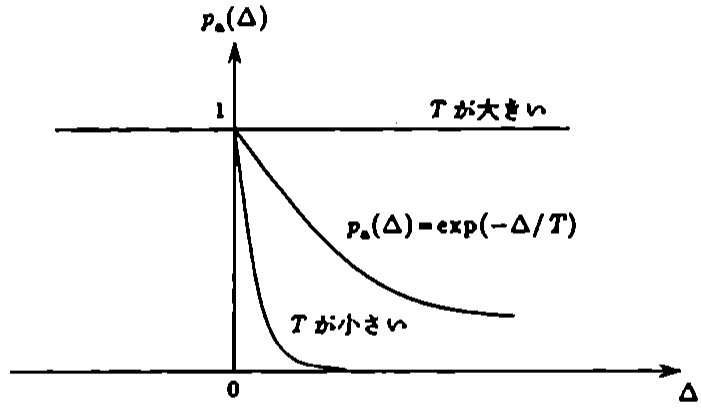


図 2.5 受理確率 $p_a(\Delta)$

としている．また，Geman ら[14] では，c を定数として，

$$T(k) \geq \frac{c}{\log(1+k)} \tag{2.28}$$

とする方法が用いられている．

遺伝アルゴリズム

遺伝アルゴリズム (Genetic Algorithm: GA) は自然界における生物の進化 (集団遺伝) モデル，すなわち世代を形成している個体の集合 (個体群) の中で，環境への適応度の高い個体がより多く生き残り (自然淘汰あるいは選択)，交叉および突然変異を起こしながら次の世代を形成していく過程を模倣した最適化法であり (図 2.6)，最適化問題 P における目的関数 $f(x)$ を適応度に，解の候補 x を個体にそれぞれ対応させる[15, 16]．以下，GA の基本構成を紹介する．

(1) 個体の表現：
自然界での個体を特徴づけているものは，遺伝子が一定の順序で配列している染色体である．GA では，有限固定長の記号列を染色体に，個々の記号を遺伝子に対応させる．すなわち，この記号列で個体を表現し，適応度を計算することになる．

(2) 適応度の計算：
GA では適応度の最大化を図る．このため，必要に応じて，最適化問題の目的関数をこの適応度にマッピングする必要がある．また，適応度は非負でなければならない．

(3) 遺伝演算子の実現：
次の世代を決定する GA の重要な構成部分がこの遺伝演算子である．以下に，代表的な遺伝演算子を列挙する．

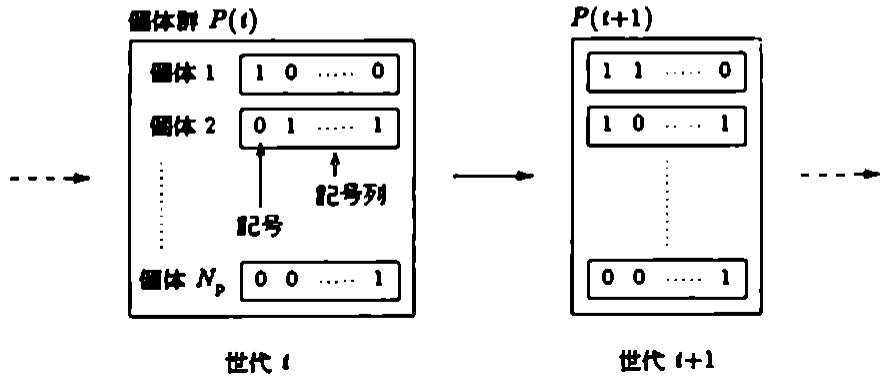


図 2.6 遺伝アルゴリズムの概要

- Selection (選択あるいは淘汰) : 適応度に応じて個体の子を増減させる.
- Crossover (交叉) : 2 つの記号列を分解し再構成する.
- Mutation (突然変異) : 記号列の一部分を確率的に変更する.

(4) パラメータの設定 :

GA を実現する際, 次のパラメータの値をあらかじめ設定する必要がある.

N_g : 世代数,
N_p : 個体群のサイズ (個体数),
p_c : crossover を行う確率,
p_m : mutation を行う確率.

最後に, GA による計算の全体的手順をまとめると以下のようになる.

1° パラメータを設定し, 初期個体群 $\mathcal{P}(1)$ を形成する. 世代 $t = 1$ とする.

2° $t < N_g$ であれば 3° へ. $t = N_g$ であれば 4° へ.

3° 遺伝演算子により次世代 $t+1$ の個体群 $\mathcal{P}(t+1)$ を生成する. $t = t+1$ として 2° へ.

4° 終了.

以上, 本章ではスケジューリング問題の定義, 分類および代表的な解法について述べるとともに, 一般的な組合せ最適化手法を紹介した. 次の第 3 章では, 本研究で対象とするジョブショップ型のスケジューリング問題について詳しく記述する. また, GA については改めて第 6 章で詳述する.

第 3 章

ジョブショップ・スケジューリング問題

3.1 問題の記述

ジョブショップ・スケジューリング問題 (以下，単にジョブショップ問題) では，仕事を処理する機械の順序が各仕事ごとに決まっている．この点を明確にするため，仕事に含まれる作業を以下のように定義する．

仕事 J_j は n_j 個の作業 O_u ($u = N_{j-1}+1, \cdots, N_j$; $N_j = \sum_{k=1}^{j} n_k$; $N_0 = 0$) からなっていて，これら n_j 個の作業 O_u は u の増加順に処理されなければならないとする．また，作業 O_u は機械 M_{μ_u} (μ_u は作業 O_u が処理される機械の番号を表す) 上で処理されるものとし，その所要時間を p_u で表す．この定義では，問題に含まれるすべての作業に一連の番号がつけられることになる．

ジョブショップ問題は，作業を節点に，作業間の先行関係制約を有向弧に対応させた有向グラフで表現できる．このとき各節点には，その節点が表す作業が処理される機械およびその処理時間を付記しておく．問題の一例を，表 3.1 および図 3.1 に示す．

ジョブショップ問題において，最適スケジュールに近くある程度良いスケジュールの集合として，次の 3 種類が知られている [17]．

(1) セミアクティブ・スケジュール：
各機械上での作業の処理順序を変更することなしに，どの作業の処理開始をも早めることができないスケジュール．

(2) アクティブ・スケジュール：
各機械上での作業の処理順序を変更してもよいとして，どの作業も，他の作業の開始時刻を遅らせることなしに，その処理開始を早めることができないスケジュール．

(3) 遅れなしスケジュール：
アクティブ・スケジュールのうちで，いずれかの作業を開始できる時刻に，どの機械も遊休状態にないスケジュール．

これら 3 種のスケジュール集合の包含関係は，

$$\text{遅れなし} \subseteq \text{アクティブ} \subseteq \text{セミアクティブ}$$

表 3.1 ジョブショップ問題の例

仕事	作業 (機械, 処理時間)		先行仕事
J_1	O_1 (M_1, 5)	O_2 (M_2, 4)	—
J_2	O_3 (M_1, 3)	O_4 (M_2, 4)	—
J_3	O_5 (M_3, 2)	—	J_1, J_2

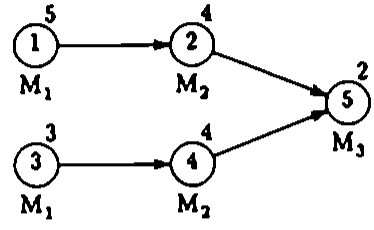


図 3.1 ジョブショップ問題の有向グラフ表現

節点が作業，有向弧が先行関係をそれぞれ表す．節点には，対応する作業の処理時間と処理される機械を付加しておく．

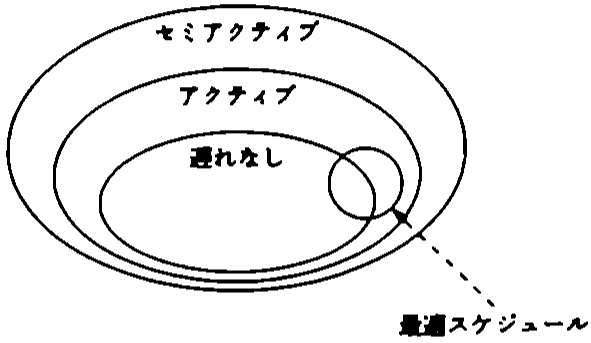


図 3.2 3 種のスケジュール集合

最適スケジュールの集合も付加した.

のようになる (図 3.2). さらに, 最大完了時間 C_{max} など, 完了時刻 C_j のそれぞれについての非減少関数を目的関数として選んだ場合には,

> 最適スケジュールが必ずしも遅れなしスケジュールであるとは限らず, アクティブ・スケジュールであって遅れなしスケジュールでない場合がある; 逆に言うと, アクティブ・スケジュールの中には, 最適スケジュール (の 1 つ) が必ず含まれる,

ということが重要な性質として導かれている [17].

3.2 種々の解法

ジョブショップ問題に対する代表的な解法を列挙する [5]. この問題については, ほとんどの場合に効率的な方法がなく, 最適スケジュールを求めるために列挙による探索が必要となる.

3.2.1 多項式時間解法

機械が 2 台で, 各仕事は高々2 つの作業からなるという制約をもった問題に対しては, 最大完了時間を最小にするための効率的なアルゴリズムがある.

Jackson のアルゴリズム— J2 | $n_j \leq 2$ | C_{max}

まず, n 個の仕事 J_j $(j = 1, \cdots, n)$ を次の 4 つの集合に分割する.

- S_1 : M_1 で処理されるただ 1 つの作業からなる仕事 J_j の集合.

- S_2 : M_2 で処理されるただ1つの作業からなる仕事 J_j の集合.
- S_{12} : 最初の作業 $O_{N_{j-1}}$ が M_1 で処理され, 2番目の作業 O_{N_j} が M_2 で処理される2つの作業からなる仕事 J_j の集合.
- S_{21} : 最初の作業 $O_{N_{j-1}}$ が M_2 で処理され, 2番目の作業 O_{N_j} が M_1 で処理される2つの作業からなる仕事 J_j の集合.

次に, S_{12} 内の仕事 J_j を,

$p_{N_{j-1}} \le p_{N_j}$ である仕事を $p_{N_{j-1}}$ の非減少順, 残りを p_{N_j} の非増加順

に並べ, S_{21} 内の仕事 J_j を,

$p_{N_j} \le p_{N_{j-1}}$ である仕事を p_{N_j} の非減少順, 残りを $p_{N_{j-1}}$ の非増加順

に並べる. また, S_1 および S_2 内の仕事の並べ方については任意とする. ここで,

M_1 では $S_{12} \to S_1 \to S_{21}$,
M_2 では $S_{21} \to S_2 \to S_{12}$

の順に作業を並べると, 最適スケジュールが得られる.

3.2.2 混合整数計画法

フローショップ型の問題と同様に, 3機械以上の一般のジョブショップ問題に対しては, 前述の $J2 \mid n_j \le 2 \mid C_{\max}$ に対する Jackson のアルゴリズムのような効率的な解法はないので, 一般的な組合せ最適化の手法である列挙的手法を適用しなければならない. このような手法の1つに, 混合整数計画問題に定式化し, 標準ライブラリ (プログラム・ライブラリ) を利用するという方法がある. これは最も容易な方法であるといえるが, 先に述べたスケジュールに関する性質をはじめ, 個々の問題の特徴を有効に利用することができず, 求解に要する計算量は一般に多くなる. 以下に, $J \mid\mid C_{\max}$ に対する定式化の手順を示す. 仕事に関する制約 (β) が付加される場合や, スケジュール評価基準 (γ) が $C_{\max}$ 以外の場合にも, $J \mid\mid C_{\max}$ と同様にして定式化できる.

まず, 作業 O_u ($u = 1, \cdots, N_n$) の処理開始時刻を t_u (中間変数) とする. 仕事に含まれる作業の処理順序は定まっているから,

$$t_u - t_{u-1} \ge p_{u-1} \quad (u = N_{j-1} + 2, \cdots, N_j;\ j = 1, \cdots, n) \tag{3.1}$$

という関係が成立する. 次に, $\mu_u = \mu_v = i$ なる2作業 O_u と O_v が, 機械 M_i 上で同時に処理されることがないという制約は,

$$t_u - t_v \ge p_v \quad \text{または} \quad t_v - t_u \ge p_u \quad (u, v = 1, \cdots, N_n) \tag{3.2}$$

表 3.2 3 機械ジョブショップ問題

仕事	作業 (機械, 処理時間)			先行仕事
J_1	O_1 (M_1, 4)	O_2 (M_2, 2)	O_3 (M_3, 2)	--
J_2	O_4 (M_1, 3)	O_5 (M_2, 1)	O_6 (M_3, 2)	—

と表すことができる．この「または」で結ばれた 2 式を「および」で表すために，0-1 整数変数 (決定変数)

$$y_{uv} = \begin{cases} 0\ , \ M_i \text{ 上で } O_v \text{ が } O_u \text{ に先行する場合}, \\ 1\ , \ M_i \text{ 上で } O_u \text{ が } O_v \text{ に先行する場合}, \end{cases} \tag{3.3}$$

を定義し，十分大きな正数 M を用いると，式 (3.2) は，

$$My_{uv} + (t_u - t_v) \geq p_v \qquad (\mu_u = \mu_v;\ u < v;\ u, v = 1, \cdots, N_n) \tag{3.4}$$

および

$$M(1 - y_{uv}) + (t_v - t_u) \geq p_u \quad (\mu_u = \mu_v;\ u < v;\ u, v = 1, \cdots, N_n) \tag{3.5}$$

と表すことができる．さらに，最大完了時間を t とすると，

$$t \geq t_{N_j} + p_{N_j} \quad (j = 1, \cdots, n) \tag{3.6}$$

という制約を付加すればよい．

以上の議論より，問題は，

$$\min_{y} \quad t \tag{3.7}$$

$$\text{subject to} \quad (3.1),\ (3.4),\ (3.5),\ (3.6) \tag{3.8}$$

$$t_u \geq 0 \qquad (u = 1, \cdots, N_n) \tag{3.9}$$

$$y_{uv} \in \{0, 1\} \quad (\mu_u = \mu_v;\ u < v;\ u, v = 1, \cdots, N_n) \tag{3.10}$$

と表される．

例 3.1 混合整数計画問題への定式化

表 3.2の J3 || C_{max} の例題を，混合整数計画問題として表現すると，

$$\min_{y} \quad t \tag{3.11}$$

$$\text{subject to} \quad t_2 - t_1 \geq 4 \tag{3.12}$$

$$t_3 - t_2 \geq 2 \tag{3.13}$$

$$t_5 - t_4 \geq 3 \tag{3.14}$$

$$t_6 - t_5 \geq 1 \tag{3.15}$$

$$My_{14} + (t_1 - t_4) \geq 3 \tag{3.16}$$

$$M(1 - y_{14}) + (t_4 - t_1) \geq 4 \tag{3.17}$$

$$My_{25} + (t_1 - t_4) \geq 1 \tag{3.18}$$

$$M(1 - y_{25}) + (t_1 - t_1) \geq 2 \tag{3.19}$$

$$My_{36} + (t_1 - t_4) \geq 2 \tag{3.20}$$

$$M(1 - y_{36}) + (t_4 - t_1) \geq 2 \tag{3.21}$$

$$t \geq t_3 + 2 \tag{3.22}$$

$$t \geq t_6 + 2 \tag{3.23}$$

$$t_1, t_2, t_3, t_4, t_5, t_6 \geq 0 \tag{3.24}$$

$$y_{14}, y_{25}, y_{36} \in \{0, 1\} \tag{3.25}$$

のようになる. □

3.2.3 分枝限定法

ここでは，ジョブショップ問題 $J \mid\mid C_{max}$ に対する選択グラフ (disjunctive graph) を利用した分枝限定法の構成例を紹介する[18, 19]．例題としては，表 3.2 に示す $J3 \mid\mid C_{max}$ の問題を利用する．

まず，図 3.3 (a) のような各作業を節点で表した有向グラフ $G = (\mathcal{V}, \mathcal{E})$ を考える．ここに，$\mathcal{V}$ は作業に対応する節点 u $(u = 1, \cdots, N_n)$ と，仮の出発節点 0 および仮の終了節点 $*$ の計 $(N_n + 2)$ 個の節点の集合である．$\mathcal{E}$ は有向弧 (u, v) の集合で，以下のときに $(u, v) \in \mathcal{E}$ である．

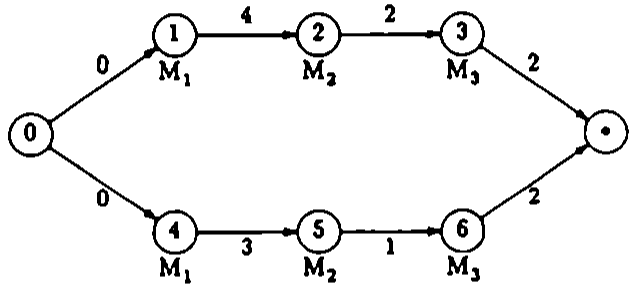
(1) 1 つの仕事内で，作業 O_u が O_v に先行する．

(2) $u = 0$ (出発節点) で，O_v が仕事内の最初の作業である．

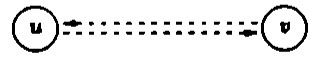
(3) O_u が仕事内の最後の作業で，$v = *$ (終了節点) である．

また，各有向弧 (u, v) には p_u (ただし $p_0 = p_* = 0$ とする) を重みとしてつける．

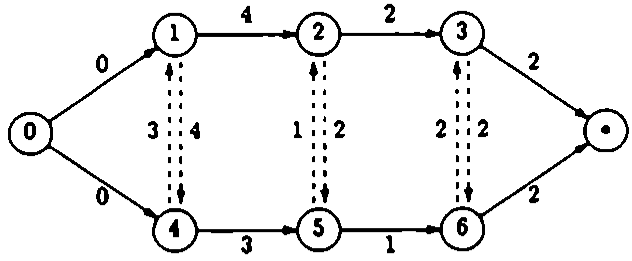
基本条件「各機械は同時に 2 つ以上の作業を処理することはできない」をグラフ G 上で考察するために，作業 O_u と O_v が同一の機械上で行われる場合は，図 3.3 (b) のように節点 u と節点 v とを互いに反対方向の 2 本の有向弧で結ぶ．有向弧 (u, v) には p_u を重みとして付加する．このとき，各有向弧を選択弧といい，対 $\{(u, v), (v, u)\}$ を選択弧対という．さらに，グラフ G にすべての選択弧対を追加して得られるグラフ，すなわち各機械上における作業順序のあらゆる可能性を付加した図 3.3 (c) のようなグラフ $G_0 = (\mathcal{V}, \mathcal{E} \cup \mathcal{D})$ を選



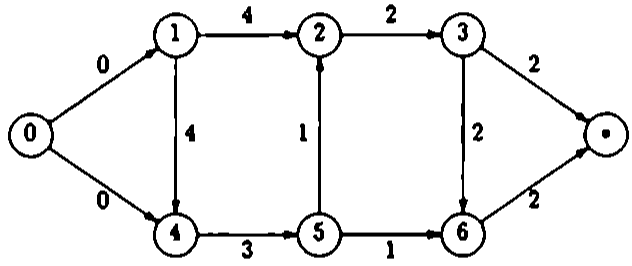
(a) ジョブショップ問題のグラフ表示 G



(b) 選択弧対



(c) 選択グラフ G_0



(d) 選択弧対がすべて解消されたグラフ G.

図 3.3 ジョブショップ問題の選択グラフ表現

択グラフという．ここに，$\mathcal{D}$ は選択弧対の集合である．

各選択弧対が結ぶ2つの作業 O_u, O_v は同時には行えない最小個数の作業の対であり，選択弧対から一方の選択弧対を消去すること，すなわち対応する2作業間の順序を決めることを選択弧対の解消という．これらすべての選択弧対を解消して得られるグラフ G_s が閉路を含まなければ，G_s は1つの実行可能スケジュールに対応する (図3.3 (d))．また，節点0から節点 $*$ に至るパスについて，そのパス上にある節点の重みの和をパスの長さ，これが最大になるようなパスをクリティカル・パスとそれぞれ定義する．このときスケジュールの最大完了時間 C_{max} は，グラフ G_s 上でのクリティカル・パスの長さとして計算され[1]，この最小値を与えるグラフが最適スケジュールに対応する．

次に，選択グラフ上での分枝限定法における分枝操作と下界値計算の概要を記述する[1]．

分枝限定法の探索図において，根にはグラフ G_0，根以外の各頂点には G_0 から1つ以上の選択弧対が解消されたグラフ $G(\mathcal{Y}) = (\mathcal{V}, \mathcal{E} \cup \mathcal{Y} \cup \mathcal{D}')$ を対応させる (図3.4 (a))[11]．$\mathcal{Y}$ は対が解消されて残った有向弧の集合であり，$\mathcal{D}'$ は未だ解消されていない選択弧対の集合である．つまり，分枝操作は1つ以上の選択弧対を解消するという操作である．

一方，下界値としては，グラフ $G(\mathcal{Y})$ から $\mathcal{D}'$ に含まれる弧をすべて開放除去したグラフ $G'(\mathcal{Y}) = (\mathcal{V}, \mathcal{E} \cup \mathcal{Y})$ (図3.4 (b)) 上でのクリティカル・パスの長さを利用できる．

以下，分枝操作の方法として代表的な2つの方法について述べる[19]．

アクティブ・スケジュール生成法

この方法は，アクティブ・スケジュールが得られるように，先行作業から順に開始時刻を決めていくものであり，分枝操作は1つの作業の開始時刻を決定することに対応する．

直前作業の処理時間区間がすべて決定しており，かつ，それ自身の処理時間区間が決定していない作業の集合を C とし，まず，

$$C = \{O_u \mid ((0, u) \in \mathcal{E}) \cap ((v, u) \notin \mathcal{E}, v \neq 0)\} \tag{3.26}$$

とする．また，$\mathcal{Y} = \phi, \mathcal{D}' = \mathcal{D}$ とする．

作業 O_u の最早開始時刻 t_u を，節点0と u を結ぶクリティカル・パスの長さで定義し，グラフ $G'(\mathcal{Y})$ 上で t_u $(O_u \in C)$ を計算する．さらに，

$$t(C) = \min \{t_u + p_u \mid O_u \in C\} \tag{3.27}$$

を与える作業を O_{u^*} とし，O_{u^*} を処理する機械を M_{i^*} $(i^* = \mu_{u^*})$ とする．次に，集合 Q および $\mathcal{R}$ を，それぞれ，

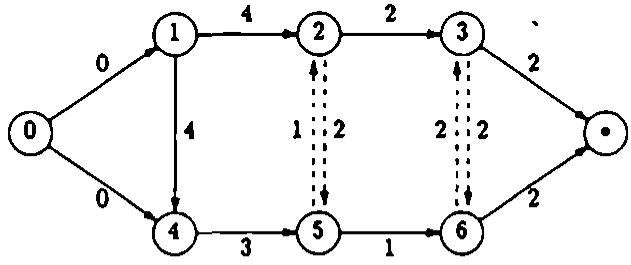
$$Q = \{O_u \mid (O_u \in C) \cap (\mu_u = i^*) \cap (t_u < t(C))\} \tag{3.28}$$

---

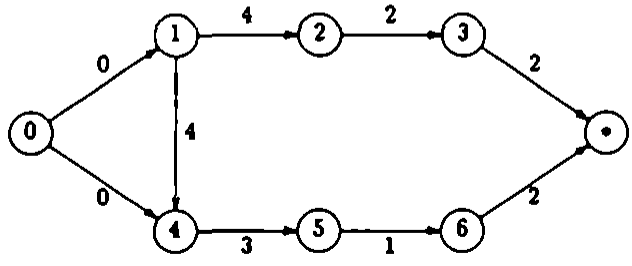
[1] クリティカル・パスの長さは効率的に (多項式時間で) 計算できる．

[2] 分枝限定法の詳細については，2.4.2 を参照されたい．

[11] グラフの節点と探索図の節点とを区別するために，本章では以下，探索図の節点を「頂点」と記す．



(a) G_0 から 1 つ以上の選択弧対が解消されたグラフ $G(\mathcal{Y})$



(b) $G(\mathcal{Y})$ から未解消の選択弧対を除去したグラフ $G'(\mathcal{Y})$

図 3.4 選択グラフ上での分枝限定法の説明図

$$\mathcal{R} = \{O_u \mid (\mu_u = i^*) \cap (O_u \text{ の開始時刻は決定していない})\} \tag{3.29}$$

と定義する．このとき，集合 $\mathcal{Q}$ から1つの作業 O_u を選び，すべての選択弧対 $\{(u, v), (v, u)\}$ (ただし，$O_v \in \mathcal{R} - \{O_u\}$) から，それぞれ (u, v) を残すように選択弧対を解消することによって，1つの頂点から $|\mathcal{Q}|$ 個の頂点に分枝する．このとき，

$$\mathcal{C} = \mathcal{C} \cup \{O_k \mid ((u, k) \in \mathcal{E}) \cap (\text{すべての } O_r\ ((v, k) \in \mathcal{E}) \text{ の時間区間が決定している})\} - \{O_u\} \tag{3.30}$$

$$\mathcal{Y} = \mathcal{Y} \cup \{(u, v) \mid O_v \in \mathcal{R} - \{O_u\}\} \tag{3.31}$$

$$\mathcal{D}' = \mathcal{D}' - \{\{(u, v), (v, u)\} \mid O_r \in \mathcal{R} - \{O_u\}\} \tag{3.32}$$

のように集合 $\mathcal{C}, \mathcal{Y}, \mathcal{D}'$ を更新する．

この分枝操作を，すべての作業の開始時刻が決まる，すなわち $\mathcal{C} = \phi$ となるまで繰り返すと，1つの実行可能スケジュールが得られる．

コンフリクト解消法

1台の機械上で2つの作業が同時に行われるようなスケジュールは実行可能でない．このとき，これら2つの作業がコンフリクトを起こしているという．コンフリクト解消法は，まず何らかの方法でスケジュールを決定し，その後コンフリクトをチェックするという方法であり，ある1つの選択弧対を解消することに分枝操作を対応させるものである．

まず，$\mathcal{Y} = \phi, \mathcal{D}' = \mathcal{D}$ とする．作業 O_u の，先行関係 $\mathcal{E} \cup \mathcal{D}$ を満たしながらの最早開始可能時刻を t_u で表す．全作業の t_u を計算し，スケジュール (必ずしも実行可能ではない) を決定する．

得られたスケジュールにおいて，コンフリクトを起こしている作業がない場合，このスケジュールは実行可能であり，さらに分枝する必要はない．コンフリクトを起こしている作業がある場合，最も早い時刻に起こしている2作業 O_u, O_v 間の選択弧対 $\{(u, v), (v, u)\}$ を解消する．すなわち，

$$\mathcal{Y} = \mathcal{Y} \cup \{(u, v)\} \tag{3.33}$$

$$\mathcal{D}' = \mathcal{D}' - \{(u, v), (v, u)\} \tag{3.34}$$

あるいは，

$$\mathcal{Y} = \mathcal{Y} \cup \{(v, u)\} \tag{3.35}$$

$$\mathcal{D}' = \mathcal{D}' - \{(u, v), (v, u)\} \tag{3.36}$$

とすることにより，2つの頂点に分枝する．

3.2.4 近似解法

ジョブショップ問題に対して，混合整数計画法あるいは分枝限定法を利用して最適スケジュールを求める方法を紹介したが，いずれも列挙による探索を基本とするので，一般に膨大な計算量を必要とする．そのために，現実に発生するジョブショップ問題のほとんどは，最適ではないが実用的なスケジュール (近似最適スケジュール) を求めることすら難しいのが現状である．

このような背景のもと，短時間である程度良いスケジュールを求めるための近似解法が数多く提案されている[3, 20, 21]．

(1) リスト・スケジューリング法：
何らかの基準に基づいて作業に優先順序をつけ，この順に空いている機械に割り当てていく方法である．たとえば，

- SPT (Shortest Processing Time first: 処理時間最小作業優先)，
- LPT (Longest Processing Time first: 処理時間最大作業優先)，
- EDD (Earliest Due Date first: 納期最早作業優先)，
- LPR (Longest Processing time Remaining first: 処理残り時間最大作業優先)

などがある．また，これらの優先規則を動的に切り換えるためのルール・ベースを利用したスケジューリングも提案されている．

(2) 分解による方法：
n 個の仕事からなる集合をいくつかの部分集合に分割し，それぞれの部分集合に含まれる仕事から構成される問題を解き，全体のスケジュールを作成する方法である．部分集合への分割の仕方によって種々の方法が考えられる．

さらに最近，部分的探索法として，

(3) 生体における情報処理を模倣した手法：
ニューロコンピューティング法，シミュレーテッド・アニーリング法，あるいは遺伝アルゴリズムなどを応用して準最適スケジュールを求める方法，

も盛んに研究されている (2.4.3 参照)．

以上，本章ではジョブショップ・スケジューリング問題の定義および代表的な解法について記述した．次の第 4 章では，ジョブショップ型でバッファを考慮した場合のスケジューリング問題を定義し，バッファがスケジュールに与える効果を調べる．また，第 5 章および第 6 章では，ジョブショップ問題に対する近似解法について，3.2.4 の (2) および (3) に基づく解法を考案する．

第 4 章

バッファを考慮したスケジューリング問題

4.1 まえがき

従来から研究されているスケジューリングの理論および手法を見れば，生産工程の様々な形態と仕事の種々の特性に応じて，またスケジュールの良さを表す目的関数の種類に応じて，各種の問題が定義されている．そして近年におけるコンピュータの目覚ましい高速化・大容量化を背景として，それらの問題に対する多くの解法 (アルゴリズム) が提案されている[1, 2]．しかし，いずれにしても従来のスケジューリング問題では，各種の在庫について陽に考慮することはせず，もっぱら生産の効率化 (工期の短縮や生産量の増大) のみに注目してきた．

ところが，現実の生産工程を見れば，原料在庫，半製品 (仕掛り品) 在庫および製品 (完成品) 在庫のいずれもが工程のスケジュールに大きな影響を与えていることは明らかである．すなわち，生産と在庫は密接に関連しており，それらの計画は本来一体として取り扱われるべき性質のものである．この章では，生産工程の各段階で仕掛り品在庫を貯える設備 (バッファと呼ぶ) を陽に考慮したジョブショップ型のスケジューリング問題について考察する．

在庫管理について見れば，過剰な在庫は大きな金利負担などの損失を伴うので，これを極力減らす方向で合理化を図るいくつかの方式が提案されている．例えば，発注点法と定期発注法に代表される伝統的な方式や[22]，カンバン方式[23]，MRP (Material Requirements Planning) のように生産システムの柔軟性を強調するための方式[24] などがある．しかし，生産工程内の各段階における仕掛り品在庫のように，自給自足的な在庫を考慮してスケジューリングを行おうとする場合には，これらの在庫管理方式をそのまま適用することはできない．

この章では，まず仕掛り品バッファを考慮したジョブショップ・スケジューリング問題を定義する[25]．次に，この問題に対して 3 種のモデル化手法を提案し，組合せ最適化のための分枝限定法などの手法を適用できるようにする．最後に，具体的な計算例を通して，計算時間および得られる解の良さの点から 3 種のモデルの比較を行うとともに，バッファの容量がスケジュールに与える効果を定量的に明らかにする．

4.2 問題の記述

4.2.1 スケジューリングと在庫

ジョブショップ型の生産工程では，外部から入った原材料が加工を施されて半製品となり，それらがさらに幾段かにわたって加工されて出荷対象となる最終製品が生産される．すなわち，原材料および幾種かの仕掛り品(半製品)が加工されながら，工程内を流れているという見方ができる．しかし，これらの物の流れは連続的ではなく，原材料が納入されてから加工が開始されるまでの間，各加工段階における処理と処理の間，および最終製品が生産されたのち出荷されるまでの間には，物の流れに停滞が生じる．このような流れの停滞を受け止めて緩衝の役目をするのが各種の在庫である．特に，加工工程の各段階で仕掛り品の在庫を適切に持つことによって，後続する作業を遅滞なく開始し，生産完了時間の増加を防止することができる．

そこで，ジョブショップ型生産工程における在庫を，

(1) 原料在庫：
原材料が納入されてから先頭作業(先行作業のない作業)を開始するまでの間の在庫，

(2) 仕掛り品在庫(半製品在庫)：
先行作業が完了してから後続の作業を開始するまでの間の半製品の在庫，

(3) 製品在庫：
最終作業(後続作業のない作業)が完了してから最終製品を出荷するまでの間の在庫，

の3種に分類し[26]，ここでは特に，(2)の仕掛り品在庫に注目する．通常，スケジューリングの目的は在庫を削減して生産性を向上させることとされている．しかし，上で述べたような在庫の効用を積極的に活用するために，工程内の各段階に適当な容量のバッファ(仕掛り品在庫を置くための設備)を置き，このバッファを有効に利用してスケジューリングを行えば，生産性をより向上させ得ることが期待できる．

4.2.2 バッファを考慮したスケジューリング問題

従来のスケジューリング問題は，与えられたすべての仕事の処理をそれぞれ1回ずつ行えば完了するような生産工程を対象としてきた．しかし，実際の生産工程の多くでは，仕事は複数回繰り返して処理され，その間に仕掛り品在庫が補充され，また消費されている．したがって，従来型(1回処理型)のスケジューリング問題によって仕掛り品在庫の影響(効果)を調べることはできない．

以下では，ジョブショップ型の生産工程において各作業の直後に仕掛り品を一時的に保持するためのバッファを設け，その作業の処理が完了するとただちに仕掛り品をバッファに入れ，後続の作業を開始するときに，このバッファから1つ取り出すものとする．そして，

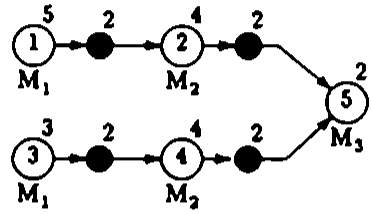


図 4.1 バッファを考慮したスケジューリング問題

節点 ○ および ● は，それぞれ作業およびバッファを表す．節点 ○ には対応する作業が処理される機械と処理時間を，節点 ● には対応するバッファの容量を，それぞれ付加する．

このバッファの容量を有限とし，所要数 (複数) の最終製品をまとめて生産する場合のスケジューリングを考える．さらに，スケジュールの良さを表す目的関数は上記の所要数を生産するのに必要な時間 (最大完了時間) $C_{\max}$ とし，それを最小化する問題をバッファを考慮したジョブショップ・スケジューリング問題 (以下，単にバッファを考慮したスケジューリング問題) と定義する．問題の一例を図 4.1 に示す．

4.3 選択グラフ・モデル

4.3.1 混合整数計画問題への定式化

4.2.2 で定義されたバッファを考慮したスケジューリング問題において，まず各機械上での作業順序を決定変数に選び，混合整数計画問題 (Mixed Integer Programming formula: MIP) として定式化する．MIP を混合整数計画法で解くことは実際の求解上は有効でないが，問題を明確に表現できる唯一の方法であり，4.3.2 では，この MIP をもとに選択グラフ・モデルを構築する．

$$\min_{t} \quad t \tag{4.1}$$

$$\text{subject to} \quad t_{ik} - t_{i,k-1} \geq p_i \qquad (i = 1, \cdots, N_m;\ k = 1, \cdots, D) \tag{4.2}$$

$$t_{jk} - t_{ik} \geq p_i \qquad ((i, j) \in \mathcal{P};\ k = 1, \cdots, D) \tag{4.3}$$

$$t_{jk} - t_{i,k+b_{ij}} \leq 0 \qquad ((i, j) \in \mathcal{P};\ k = 1, \cdots, D - b_{ij}) \tag{4.4}$$

$$(M + p_j)y_{ik_i,jk_j} + (t_{ik_i} - t_{jk_j}) \geq p_j$$
$$(\mu_i = \mu_j;\ i < j;\ k_i = 1, \cdots, D;\ k_j = \alpha_{ij}, \cdots, \beta_{ij}) \tag{4.5}$$

$$(M + p_i)(1 - y_{ik_i,jk_j}) + (t_{jk_j} - t_{ik_i}) \geq p_i$$
$$(\mu_i = \mu_j;\ i < j;\ k_i = 1, \cdots, D;\ k_j = \alpha_{ij}, \cdots, \beta_{ij}) \quad (4.6)$$

$$t \geq t_{iD} + p_i \qquad (i \in \mathcal{F}) \quad (4.7)$$

$$t_{ik} \geq 0 \qquad (i = 1, \cdots, N_m;\ k = 1, \cdots, D) \quad (4.8)$$

ただし，

p_i ：作業 O_i の処理時間，

μ_i ：作業 O_i の処理される機械番号，

b_{ij} ：作業 O_i と作業 O_j の間にあるバッファ B_{ij} の容量，

M ：十分大きな正定数 ($\geq \max |t_i - t_j|$)，

D ：最終製品の所要数 (各作業の必要処理回数)，

$\mathcal{P}$ ：先行関係 (O_i が O_j に先行するとき $(i, j) \in \mathcal{P}$)，

$\mathcal{F}$ ：最終作業 (後続作業のない作業) の集合，

t ：最大完了時間 $C_{\max}$ ，

t_{ik} ：作業 O_i の k 回目の処理開始時刻，

y_{ik_i,jk_j} ：作業対 O_i, O_j $(\mu_i = \mu_j)$ の処理順序を決める決定変数; M_{μ_i} $(= M_{\mu_j})$ 上で，
O_i の k_i 回目の処理が O_j の k_j 回目に先行するとき 1，そうでないとき 0，

とする．式 (4.5) および式 (4.6) の添字については，$(i, j) \in \mathcal{P}$ のときは，

$$\alpha_{ij} = \max\{1,\ k_i - b_{ij} - \lceil p_i/p_j \rceil\}, \quad (4.9)$$

$$\beta_{ij} = \max\{1,\ k_i - 1\} \quad (4.10)$$

とし ($\lceil x \rceil$ は x 以上の最小の整数)，一方 $(i, j) \notin \mathcal{P}$ のときは，

$$\alpha_{ij} = 1, \quad (4.11)$$

$$\beta_{ij} = D \quad (4.12)$$

とする．

ここで，制約条件 (4.2) ～ (4.8) について説明しておく．式 (4.2) は，作業 O_i の $(k-1)$ 回目の処理が k 回目の処理に先行することを表す．式 (4.3) は，作業 O_i が作業 O_j に先行する場合，O_i の k 回目の処理が O_j の k 回目の処理に先行するという条件である．次の式 (4.4) はバッファ容量の制約を表すものである．O_i が O_j に先行する場合，O_i により生産される半製品がバッファ容量分 b_{ij} まで作り置きできることに対応して，O_j の k 回目の処理開始が O_i の $(k + b_{ij})$ 回目の処理開始よりも前であればよいことを表す．ただし，式 (4.4) では O_i の開始時点でバッファ B_{ij} の空きを調べていることになる．O_i の終了時点で，バッファ B_{ij} の空きを調べるのであれば，

$$t_{jk} - t_{i,k+b_{ij}} \leq p_i \qquad ((i, j) \in \mathcal{P};\ k = 1, \cdots, D - b_{ij}) \quad (4.13)$$

とすればよい.

式 (4.5) および式 (4.6) は機械条件を表す. すなわち, 機械は同時に 1 つの作業しか処理できないので, 同じ機械を使用する作業対 O_i, O_j の間にはその処理順序が定められなければならないことを表す. 2 作業 O_i, O_j の間に先行制約が存在するときは, 制約条件 (4.2) および (4.3) を考慮して添字 k_j の上限および下限を, それぞれ式 (4.9), (4.10) および式 (4.11), (4.12) により定める. さらに式 (4.7) は, $C_{\max}$ が作業終了時刻の最大値で与えられることを表している. 式 (4.8) は, 作業の開始時刻が非負となることを保証するためのものである.

4.3.2 モデルの記述

ここでは, 4.3.1 の MIP に基づいて, 問題を選択グラフにより表現する[27]. これにより, スケジューリング問題に対する厳密解法のうちで最も有効とされている選択グラフ上での分枝限定法 (3.2.3 参照) を, バッファを考慮した場合にもそのまま適用することができる.

先に定義したバッファを考慮した問題では作業が複数回処理されるので, まず最初に各回の処理 (以下, 作業単位) に対応して節点を設ける. 節点番号は便宜的に作業番号を表す数字にその処理回数を添字として付加し, 例えば, 1_2 (O_1 の 2 回目の処理) のように表す. さらに, 仮想の出発節点 0 と終了節点 $*$ を追加する.

次に, 制約条件 (4.2), (4.3), (4.4), (4.7) および (4.8) に対応する有向弧を付け加える. まず, 式 (4.2) に対応する節点 i_{k-1} と i_k とを重み p_i をもつ有向弧で結ぶ. 式 (4.3) に対しては, $(i,j) \in \mathcal{P}$ のときに, 節点 i_k と j_k とを重み p_i をもつ有向弧で結ぶ. さらに, 式 (4.4) に対応して, $(i,j) \in \mathcal{P}$ であれば節点 j_k と $i_{k+b_{ij}}$ とを重み 0 をもつ有向弧で結ぶ. 式 (4.7) については, $i \in \mathcal{F}$ であれば, 節点 i_D と節点 $*$ とを重み p_i をもつ有向弧で結ぶ. 式 (4.8) に対しては, 仮想節点 0 と先頭作業 O_i の 1 回目の処理を表す節点 i_1 とを重み 0 をもつ有向弧で結ぶ.

最後に, 制約条件 (4.5) および (4.6) に対応する選択弧対を付け加える. すなわち, $\mu_i = \mu_j$ であるすべての節点対 i_k, j_l について, i_k から j_l あるいは j_l から i_k に向かう有向弧で結ばれたパスがない場合に, これら 2 節点 i_k, j_l 間を選択弧対で結ぶ. 各選択弧には, その起点となる節点が表す作業の処理時間を重みとして付加する.

一例として, 図 4.1 の問題を選択グラフで表現すると図 4.2 となる. ただし, 最終製品所要数 $D = 3$ とした. バッファを考慮したスケジューリング問題の選択グラフ表現において作業単位をあらためて作業と見なすと, 従来型のスケジューリング問題とまったく同形となる. したがって, バッファを考慮したスケジューリング問題に対して, これまでに提案されている選択グラフ上での種々の解法をそのまま適用できる.

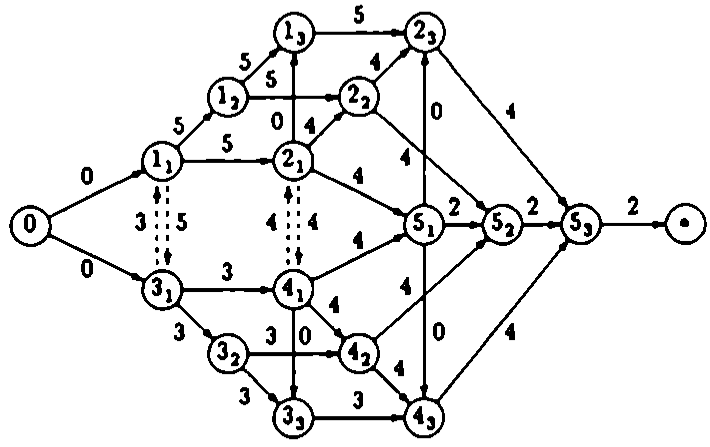


図 4.2 バッファを考慮したスケジューリング問題の選択グラフ表現

図 4.1 に対応する．ただし，図が複雑になるのを避けるため，選択弧対のほとんどを省略してある．

4.4 タイム・ペトリネット・モデル

一般に，バッファを考慮したスケジューリング問題に対して最適スケジュール (最適解) を求めたい場合には，アクティブ・スケジュール集合での探索を基本とする解法を構成しなければならない．4.3 での選択グラフ・モデルは，このような分枝限定法の適用を可能とするものである．しかし，ある程度良いスケジュールでよい場合には，探索範囲を遅れなしスケジュール集合に限定することによって，計算時間を短縮できる．さらに，遅れなしスケジュール集合内での最適なスケジュールは大域的な最適スケジュールであることが多く，最適でない場合でも，最適にかなり近いスケジュールであることがわかっている[17]．そこで，以下ではまず，バッファを考慮したスケジューリング問題を 4.3.1 とは異なる形の混合整数計画問題 (MIP) に定式化する．次にこの定式化を基にして，タイム・ペトリネットで問題を記述し，遅れなしスケジュール内での探索を目的とした分枝限定法を構成する[28]．

4.4.1 混合整数計画問題への定式化

作業の開始時刻を決定変数として，MIP に定式化する．なお，時間を離散化しておき，単位時間を期と呼ぶこととする．

$$\min_{y} \quad \sum_{t'=1}^{T} w_{t'} \tag{4.14}$$

$$\text{subject to} \quad x_{i,*,T+1} = D \qquad (i \in \mathcal{F}) \tag{4.15}$$

$$x_{i,j,t+1} = x_{ijt} - y_{jt} + y_{i,t-p_i+1} \qquad ((i,j) \in \mathcal{P};\ t = 1, \cdots, T) \tag{4.16}$$

$$y_{it} = u_{it1} = \cdots = u_{it,t+p_i-1} \qquad (i = 1, \cdots, N_n;\ t = 1, \cdots, T - p_i + 1) \tag{4.17}$$

$$\sum_{\substack{i=1 \\ \mu_i = c}}^{N_n} \sum_{t=1}^{T} u_{it t'} \le 1 \qquad (c = 1, \cdots, m;\ t' = 1, \cdots, T) \tag{4.18}$$

$$y_{it} \in \{0, 1\} \qquad (i = 1, \cdots, N_n;\ t = 1, \cdots, T) \tag{4.19}$$

$$0 \le x_{ijt} \le b_{ij} \qquad ((i,j) \in \mathcal{P};\ t = 1, \cdots, T+1) \tag{4.20}$$

$$w_{t'} \ge \sum_{t=1}^{T} u_{it t'} \qquad (i = 1, \cdots, N_n;\ t' = 1, \cdots, T) \tag{4.21}$$

$$w_1 \ge w_2 \ge \cdots \ge w_T \tag{4.22}$$

ただし，

p_i ：O_i の処理時間，

μ_i : O_i の処理される機械番号.

b_{ij} : O_i と O_j の間にあるバッファ B_{ij} の容量.

T : C_{max} の上限値 (あらかじめ設定).

D : 最終製品の所要数.

$\mathcal{P}$: 先行関係 (O_i が O_j に先行するとき $(i, j) \in \mathcal{P}$).

$\mathcal{F}$: 最終作業 (後続作業のない作業) の集合.

x_{ijt} : バッファ B_{ij} の第 t 期の半製品在庫数.

y_{it} : O_i の開始時刻 (期) を表す決定変数; O_i の処理が第 t 期に始められるとき 1, そうでないとき 0.

$u_{itt'}$: 第 t 期に始められた作業 O_i が第 t' 期に行われるとき 1, そうでないとき 0.

とする. なお, 最終作業 O_i $(i \in \mathcal{F})$ の直後のバッファを $B_{i\bullet}$ と記すことにし, このとき $(i, \bullet) \in \mathcal{P}$ とする.

ここで, 制約条件 (4.15) ~ (4.18) および (4.20) について説明しておく. 式 (4.15) は, 第 $(T+1)$ 期に最終製品在庫数がその所要数 D に等しくなることを表す. 式 (4.16) は, バッファ B_{ij} の第 $(T+1)$ 期の在庫数を, 作業 O_j が第 $(t+1)$ 期に開始される場合は第 T 期の在庫数から 1 減じ, さらに作業 O_i が第 t 期に完了する場合には 1 加えた値とすることを表す. 式 (4.17) は, 第 T 期に始められた作業 O_i の処理が p_i 期間連続して行われるという条件である. 式 (4.18) は, 各期においてそれぞれの機械上で高々1 つの作業しか行われないことを表す. 式 (4.20) は, バッファ B_{ij} の在庫数が非負でかつその容量を超えないという条件である.

さらに, 目的関数 (式 (4.14)) で最小化される $w_{t'}$ は,

$$w_{t'} = \max_i \left\{ \sum_{t=1}^{T} u_{itt'} \right\} \tag{4.23}$$

$$= \begin{cases} 0, & \sum_{i=1}^{N_m} \sum_{t=1}^{T} u_{itt'} = 0 \text{ のとき} \\ 1, & \sum_{i=1}^{N_m} \sum_{t=1}^{T} u_{itt'} \neq 0 \text{ のとき} \end{cases} \tag{4.24}$$

と表すことができ, $\sum_{t'} w_{t'}$ を最小化することは, 1 台以上の機械が処理を行っている期間をできるだけ少なくすること, 言い換えると, すべての機械が休んでいる期間をできるだけ多くすることに対応する. さらに式 (4.22) により, 遊休期間は後のほうに集中する. この遊休期間を除くと C_{max} が最小となるスケジュールが得られるので, 式 (4.14) ~ (4.22) で表される MIP は, C_{max} 最小化問題と本質的に等価であるといえる.

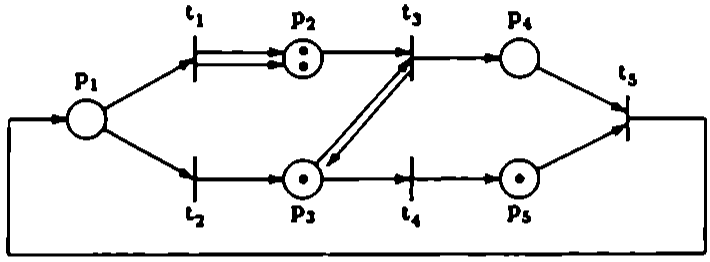


図 4.3 ペトリネット (PN) の一例

$p_1 \sim p_5$ は PL を，$t_1 \sim t_5$ は TR をそれぞれ示す．
また，黒丸 • はトークンを表す．

4.4.2 モデルの記述

4.4.1 のように時間を離散化して MIP に定式化すると，バッファを考慮したスケジューリング問題は，半製品到着，処理，(処理の施された) 半製品の送り出しという一連のいわゆる離散的な事象が繰返し発生する離散事象システムにおける問題と見なすことができる．この離散事象システムを表現する有力な手法として，ペトリネット (Petri Net: PN)[29]，あるいは PN に時間の概念を導入したタイム・ペトリネット (Timed Petri Net: TPN)[30,31] がある．以下では，TPN を用いてバッファを考慮したスケジューリング問題を表現する手法を示す．

ペトリネット (PN)

PN のグラフ表現は，2 種類の節点 (node) および有向枝 (arc) によって構成される．通常，節点と有向枝には次の記号が用いられる．

○ : プレース (place)
| : トランジション (transition)
→ : 有向枝

PN では，有向枝はプレース (以下 PL) とトランジション (以下 TR) とを結合するものであり，始点が PL のときは終点が TR，始点が TR のときは終点が PL である．したがって PL から PL へ，あるいは TR から TR への有向枝は存在しない．図 4.3 に PN の一例を示す．$p_1 \sim p_5$ は PL を，$t_1 \sim t_5$ は TR をそれぞれ示す．図 4.3 では p_1 から t_1 へ向かう有向枝 (t_1 への入力枝) や，t_1 から p_2 へ向かう有向枝 (t_1 からの出力枝) が存在する．このとき p_1 を t_1 に対する入力プレースといい，p_2 を t_1 に対する出力プレースという．

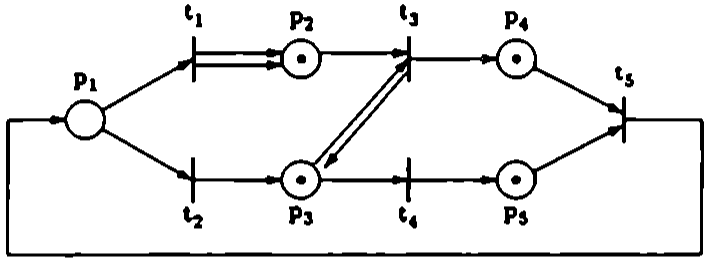


図 4.4 PN でのトランジションの発火に伴うマーキングの変化

図 4.3 の例で，t_3 が発火した後のマーキングを示す．

PN では一般に，システムの状態を PL 内の

• : トークン (token)

の分布状況 (マーキング) によって，また，システムの状態遷移を TR の発火に伴うトークンの移動 (マーキングの変化) によって表現する．ここで，TR の発火規則は以下のように定められている．

(1) TR t_j へのすべての入力 PL p_i において，p_i から t_j への枝数以上のトークンが p_i 内に存在するとき，t_j は発火可能である．

(2) TR t_j の発火に伴い，すべての入力 PL p_i から，p_i から t_j への枝数に等しいトークンを取り去る．

(3) TR t_j の発火に伴い，すべての出力 PL p_k に対して，t_j から p_k への枝数に等しいトークンを生成し，割り当てる．

図 4.3 の例では，発火可能な TR は t_3 と t_4 であり，t_3 が発火すると，トークンの分布は図 4.4 に示すようになる．図 4.4 において発火可能な TR は，t_3, t_4 および t_5 である．このように，ペトリネットでは発火可能な TR を順に発火させることにより，システムの状態遷移を表すことができる．

TR の発火に際して，発火可能 TR が 2 個以上存在することとき，一方を発火させることによって他方が発火できなくなる関係を競合と呼ぶ．図 4.4 の例では，TR t_3 および t_4 がともに発火可能であるが，先に t_4 が発火すると t_3 が発火できなくなる．このような競合関係にある TR について，その発火順序を定めることを競合の解消という．このとき，競合の解消の仕方によっては，以後のシステムの状態が異なるものとなる．

タイム・ペトリネット (TPN)

純粋な PN では時間経過を表現することができないので，限時要素を含むシステムを表現する場合には PN を拡張する必要がある．TPN は PN の TR に発火継続時間を持たせることによって，システムの時間依存性を表現できるようにしたものである．図 4.5 に TPN の一例を示す．なお，TPN のグラフ表現において，発火継続時間を持つ TR を ▮ で表示するものとする．

TPN において，TR の発火は次の規則に従って行われる．

(1) TR t_j へのすべての入力 PL p_i において，p_i から t_j への枝数以上のトークンが p_i 内に存在するとき，t_j は発火可能である (PN の発火規則 (1) に同じ)．

(2) TR t_j の発火に伴い，すべての入力 PL p_i から，p_i から t_j への枝数に等しいトークンを取り去る (PN の発火規則 (2) に同じ)．

(3) TR t_j の発火は，その発火継続時間 τ_j が経過した後完了する．

(4) TR t_j の発火が完了した時点で，すべての出力 PL p_k に対して，t_j から p_k への枝数に等しいトークンを生成し，割り当てる (PN の発火規則 (3) に対応)．

図 4.6 に，TPN での TR の発火に伴うトークンの移動の様子を示す．

TPN の拡張

TPN のトランジション (TR) を作業に，プレース (PL) をバッファにそれぞれ対応させることにより，バッファを考慮したスケジューリング問題を TPN で表現することができる．このとき，TR の発火で作業の処理を，PL (バッファPL) 内のトークン数でバッファ内の半製品の数を表すことになる．バッファごとにさらにもう 1 つの PL (バッファ容量 PL) を設け，そのトークン数でバッファの空き容量を表現する場合には，

(1) ある作業が開始される時点で，直前のバッファから半製品が取り出されて空き容量が 1 つ増えることに対応して，TR が発火を始める瞬間に直前のバッファ容量 PL にトークンを送る，

(2) その作業の処理が完了してはじめて半製品ができることに対応して，TR が発火を終える時点でバッファPL にトークンを送る，

という 2 つの要求を満足しなければならない[1]．すなわち，ある動作が行われるとき，その開始時点においてある条件を成立させ，終了時点でまた別の条件を成立させるということである．

通常の TPN では，このようなシステムの状態を 2 段階に変化させる動作を 1 つの TR で表すことはできない．そこで，TR に 2 種類の出力プレースを持たせて TPN を拡張した拡

[1] バッファ容量 PL を省略し，バッファPL にトークン数の上限を設けることによってこれらの要求を満たすことができるが，本章では，分枝限定法の構成を容易にするために以下に述べる方法で TPN を拡張した．

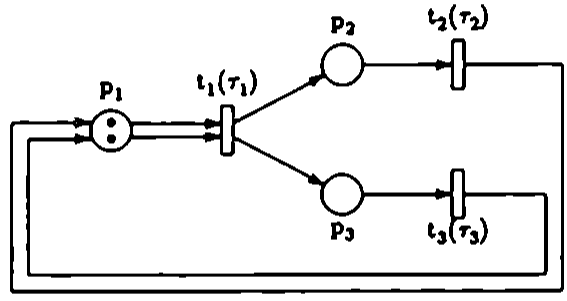
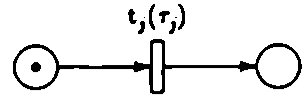


図 4.5 タイム・ペトリネット (TPN) の一例

$t_j(\tau_j)$ は t_j の発火継続時間が τ_j であることを表す.



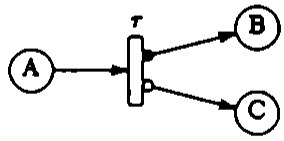
(a) $t = T - 0$

(b) $T < t < T + \tau_j$

(c) $t = T + \tau_j - 0$

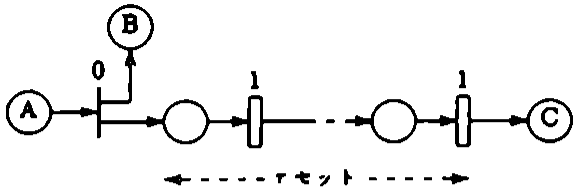
図 4.6 TPN でのトランジションの発火に伴うマーキングの変化

$t_j(\tau_j)$ が時刻 $t = T$ に発火する場合のマーキングの変化を示す.



(a) 基本モデル

円がプレース，四角形がトランジションにそれぞれ対応する．プレース Ⓐ に 1 つ以上のトークンが存在する場合にトランジションは発火でき，発火してから r 時間経過した後発火が完了する．黒丸を介して接続される出力プレース Ⓑ には発火が開始される時点で 1 つのトークンが入れられ，白丸を介して接続される出力プレース Ⓒ には発火が完了した時点で 1 つのトークンが入れられる..



(b) 等価表現

2 種類のトランジション，すなわち発火継続時間が 0 の 0-TR (|) と発火継続時間が 1 の 1-TR (|) とを用いると，基本モデル (a) は，等価的に (b) のように表現できる．

図 4.7 拡張タイム・ペトリネット

張TPNを提案する[28]．図4.7 (a)に拡張TPNの単位モデルを示すが，黒丸を介して接続される出力PL Ⓑ にはTRが発火する時点でトークンが送られ，白丸を介して接続される出力PL Ⓒ には発火が終了する(発火してから τ だけ経過した)時点でトークンが送られる．

2種類のTR，すなわち発火に単位時間要するTR (1-TR)と即時に発火が終了するもの(0-TR)を定義すると，図4.7 (a)は等価的に図4.7 (b)のように表現できる．ただしある時刻において，まず発火可能な0-TRを発火させてから，次に1-TRを発火させて時間を進めるというように，TRの発火に優先関係を設けておく．この等価表現(b)を用いるとTRやPLの数は増えるが，トークンに付随する時間の概念を省略できるので，解析やシミュレーションを行う場合には有効である．一方(a)の表現法は図を簡略化できるという利点があるので，以下ではこれら2種の表現法を適宜使い分ける．

拡張TPNによる表現

4.4.1のMIPにおける制約条件と，拡張TPNのPLおよびTRとを表4.1のように対応づけることにより，問題を拡張TPNで表現する．一例として，図4.1の問題を拡張TPNで表現すると図4.8のようになる．

4.4.3 分枝限定法の構成

原則として，0-TRおよび1-TRを自然発火させる．すなわち，時間を単調に進めながら，ある時刻においてまず発火可能な0-TRを発火させ(処理開始可能作業を始めることに対応)，次に1-TRを発火させる(その時刻に開始される作業および処理中の作業を1単位時間実行することに対応)ということを繰返す(自然発火規則)．この自然発火規則に基づいて生成されるスケジュールは，遅れなしスケジュールに限定されるので，分枝限定法を適用することにより，最適に近いスケジュールが比較的高速に求められる．

分枝限定法の構成について言うと，TPNモデルに固有なのは分枝操作と下界値計算の方法である．まず，0-TRおよび1-TRの発火と分枝操作を以下のように対応させる．ただし，実際に複数の子頂点が生成されるのは，ある時刻に複数の0-TRが競合する場合だけである．

1° 0-TRの発火：
0-TRの中から発火可能なものを選び出す．発火可能なものがあれば，同時に発火可能なものの極大集合(発火可能なTRの組合せからなる集合のうち，他の真部分集合となっていないもの)を求め，各集合ごとに分枝しマーキングを変える．このとき分枝終了．発火可能なものがなければ2°へ．

2° 1-TRの発火：
1-TRの中から発火可能なものを選び出す．発火可能なものがなければ終端する(実行可能スケジュールが得られている)．発火可能なものがあれば(これらは競合しない)，

表 4.1 制約条件とトランジションおよびプレースとの対応

拡張タイム・ペトリネット	混合整数計画問題
バッファPL	制約式 (4.16)
バッファ容量 PL	制約式 (4.20)
機械 PL	制約式 (4.18)
原材料 PL、製品 PL	制約式 (4.15)
自然発火規則	制約式 (4.17), (4.22)
0-TR の発火	変数 $y = 1$
1-TR の発火	変数 $u = 1$

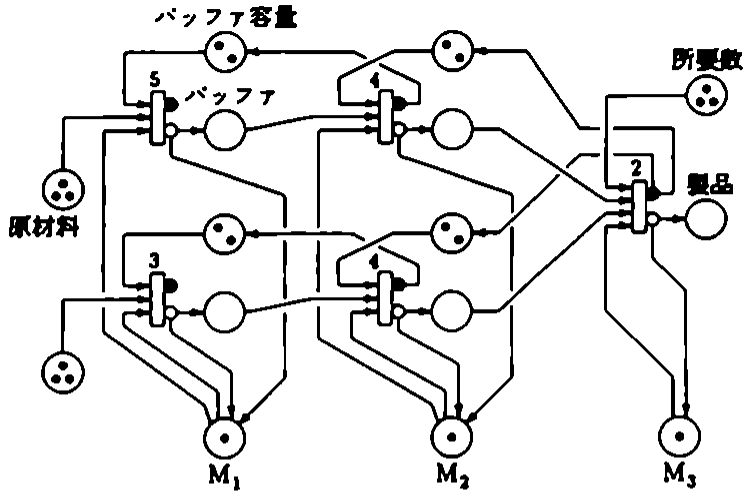


図 4.8 拡張タイム・ペトリネット表現の例

図 4.1 の問題に対応する.

頂点を1つ生成し，マーキングを変化させるとともに時刻を1進める．

次に，限定操作に利用する下界値を計算する方法として，次の2通りを考える．

(1) TPN モデルで，バッファ容量 PL とこれに接続する枝，およびバッファ PL とこれに接続する枝を取り除き，さらに作業ごとに新たに1つの入力 PL を設定して残り必要処理回数分のトークンを入れた後，最終マーキングに至るまでに必要な時間を現時刻に加えて下界値とする．すなわち M_i 上での残り作業の処理時間の和を s_i とすると，(現時刻 + $\max_i s_i$) となる．

(2) 上の (1) とは逆に，機械 PL とこれに接続する有向枝を取り除き，最終マーキングに至るまでに最小限必要な時間を現時刻に加えて下界値とする．

なお，最終マーキングとは，最終作業直後のバッファに対応するバッファ PL (図 4.8 中の製品 PL に対応) に，最終製品所要数に等しいトークンがあるようなマーキングのことである．

方法 (1) は機械に遊休時間がないものとし，また方法 (2) は半製品が滞りなく理想的に流れるとして，下界値を計算するものである．先行関係制約が多く機械の負荷率が低くなるような問題に対しては方法 (2) を，逆の場合には方法 (1) を適用すれば，限定操作が効率的に行われるものと考えられる．なお，4.6 における計算例では，方法 (1) による下界値を用いることにする．

4.4.4 繰返しスケジューリング

最終製品を1つ生産するのに必要な作業群を1セットと呼ぶことにすると，4.4.2 でのモデルは，時刻 0 に生産を開始して所要セット数をまとめて処理する場合に，C_{max} を最小化するスケジューリング (一括スケジューリング) を対象とするものである．最終製品所要数が多くなると，そのすべてを一括して生産するスケジュールを作成することはほとんど不可能であり，いくつかのセットに分割して繰返しスケジュールを求めることが必要となる (図 4.9)．そこで，複数セット周期での繰返しスケジュールを求めることを考える (繰返しスケジューリング)．

繰返しスケジューリングでは，

(1) 繰返しセット数 (1 回の繰返しで生産される最終製品数) s の決定，

(2) 繰返し時間パターン (機械ごとの処理開始可能時刻のずれ) r_i $(i = 1, \cdots, m)$ の決定，

(3) 繰返し周期の最小化[1]

が重要である．まず，(1) と (2) が与えられた場合に (3) の繰返し周期を最小化する問題は，図 4.10 に示す PL と TR を，図 4.8 の拡張 TPN に機械ごとに1組ずつ追加することによってモデル化でき，4.4.3 の分枝限定法を用いて求解できる．次に (1) の繰返しセット数の決

[1] 機械 M_i において，1つのセットの最初の作業を開始してから最後の作業を完了するまでの時間を t_i とするとき，最大値 $\max_i t_i$ を繰返し周期と定義する．

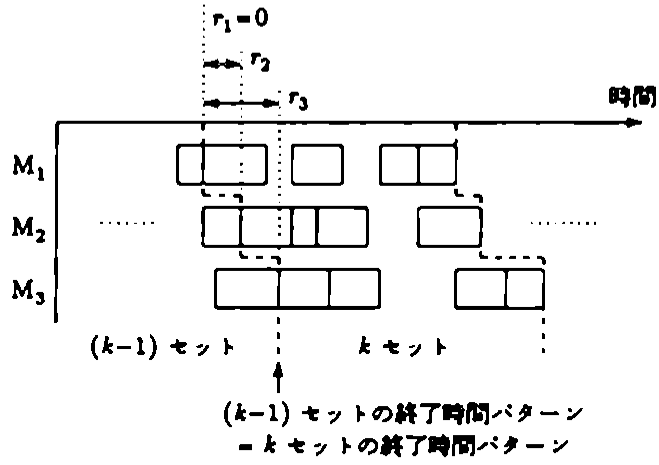


図 4.9 繰返しスケジューリングの例

$m=3$ の場合の例を示す．機械ごとの次セットの仕事の処理を開始できる時刻のずれ (r_i) をパターンと呼ぶ．繰返しスケジューリングでは，第 $(k-1)$ セットの終了時間パターンが第 k セットの開始時間パターンとして用いられる．

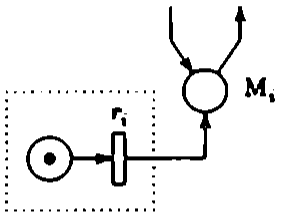


図 4.10 追加される 1 組のトランジションとプレース

M_i に対応するプレースには，r_i 時間の後にトークンが入れられる．

定は，セット数を 1 から順に 1 ずつ増加させて繰返しスケジュールを求めるという手続きを，1 セット当りの繰返し周期がそれ以上短縮されなくなるまで反復することによって行う[1]．なお，(2) の繰返し時間パターンはあらかじめ設定するものとする．

4.5 ガントチャート・モデル

4.3 の選択グラフ・モデルおよび 4.4 のタイム・ペトリネット・モデルは，ある評価基準を最小あるいは最大にするスケジュールを数理計画的手法によって求めるためのものであり，スケジュールの生成される過程を把握しにくい．したがって，これらのモデルはヒューリスティックな解法を構成する場合には適さない．そこで本節では，バッファを考慮した問題に対してガントチャートに基づいたモデル化手法を示すとともに，分枝限定法およびリスト・スケジューリング法を構成する[27,32]．

4.5.1 モデルの記述

ガントチャートとは，横軸に時間，縦軸に機械をとって，各作業が処理される時間区間を図示したものである (図 2.1)．

ガントチャート・モデルでは，ガントチャート上に作業を 1 つずつ割り付けていくことにより，すなわち，作業の処理時間区間を必要処理回数分だけ順に決めていくことによりスケジュールを作成する．まず，既にその処理時間区間が決定している作業を実行したと仮定し，必要処理回数に達していない作業について，

(1) 直前の (それぞれの) バッファに半製品が 1 つ以上入れられる時刻の最大値 t_1，

(2) 直後のバッファに空きができる時刻 t_2，

[1] 実際には，繰返しセット数の上限 s_{max} をあらかじめ設定しておいて，s_{max} まで反復すると強制的に終了させ，それまでで 1 セット当りの繰返し周期が最も短いものを繰返しセット数とする．

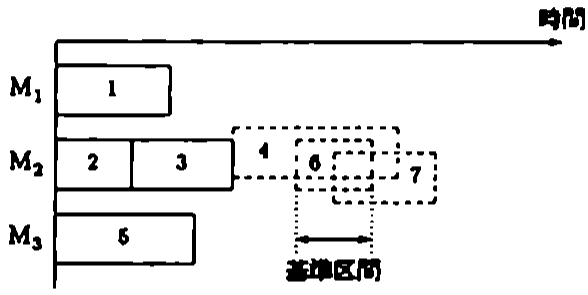


図 4.11 候補作業と基準区間

処理開始可能作業 O_4、O_6 および O_7 のうち，O_6 の最早終了時刻が最小となる．よって，作業 O_6 が候補作業，その可能区間が基準区間として選ばれる．

(3) 使用する機械が空く時刻 t_3

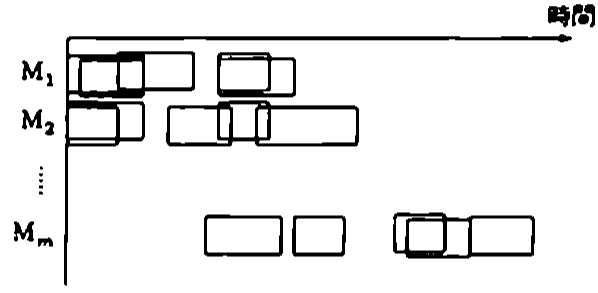
を計算する．ただし，それぞれの時刻が未定の場合は ∞ とする．さらに，$\max(t_1, t_2, t_3) < \infty$ となる作業を処理開始可能作業と定義する．

すべての処理開始可能作業について，まずその最早開始可能時刻 ($\max(t_1, t_2, t_3)$) と最早終了可能時刻，すなわち最早処理可能時間区間 (以下単に可能区間) を求める．そして，処理開始可能作業の中から最早終了可能時刻の最も早いものを選び出し，その作業を候補作業，その可能区間を基準区間と呼ぶ (図 4.11)．

候補作業以外の処理開始可能作業の可能区間が基準区間と重なっていなければ，候補作業の処理をその可能区間 (基準区間) に行う (ことを決定する)．重なっていれば，次節以後で述べる方法で 1 つの作業を選び，それを処理する．このような作業の処理時間区間の決定をすべての作業の必要処理回数が満たされるまで繰り返すと，1 つの実行可能スケジュールが得られる．

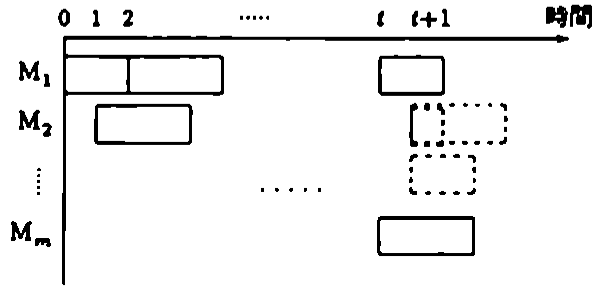
ここで，3 種類のモデル (選択グラフ (4.3)，タイム・ペトリネット (4.4)，ガントチャート (4.5)) におけるスケジュール生成過程を図示すると図 4.12 のようになる．選択グラフ・モデルでは，あらかじめすべての作業が割り付けられているが，同一機械上での作業の処理時間区間に重なりがあり実行可能ではない．これらの重なりを順に解消していくことによって，実行可能スケジュールが得られる．一方，タイム・ペトリネット・モデルおよびガントチャート・モデルでは，作業が 1 つずつ順に割り付けられることによって，実行可能スケジュールが得られる．ただし，タイム・ペトリネット・モデルでは (離散化された) 時間を 1 つずつ進めながら処理開始 (割付け) 可能作業が調べられるのに対して[1]，ガントチャート・

[1] 時間ごとに処理開始可能作業を調べていくという条件が付加されることにより，TPN モデルにおいて得られるスケジュールは遅れなしスケジュールに制限される．



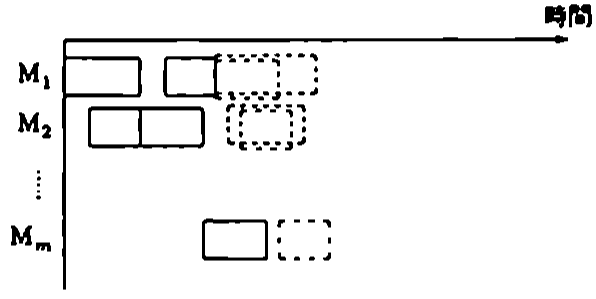
(a) 選択グラフ・モデル

あらかじめすべての作業を割り付けた後，同一機械上での作業の処理時間区間に重なりを順に解消していく．



(b) タイム・ペトリネット・モデル

(離散化された) 時間を 1 つずつ進めながら処理開始可能となる作業を 1 つずつ順に割り付けていく．



(c) ガントチャート・モデル

ガントチャート上で作業を 1 つずつ順に割り付けていく．

図 4.12 スケジュール生成過程

モデルでは，開始可能時刻に関係なく処理開始可能作業が抽出される．

4.5.2 分枝限定法の構成

候補作業以外の可能区間が基準区間と重なっている場合には，候補作業も含めて，重なっている作業のうち 1 つの作業をその可能区間に行うことに対応して分枝操作を行う．すなわち，重なっている作業の数だけ子頂点が生成されることになる．一方，下界値 g としては，

$$g = \max_i \{d_i + s_i\} \tag{4.25}$$

を用いる．ただし，

d_i : 機械 M_i 上で処理時間区間が決定している作業の終了時刻の最大値，
s_i : 機械 M_i 上の作業の残り処理時間の和

である．

4.5.3 リスト・スケジューリング法の構成

候補作業以外の可能区間が基準区間と重なっている場合には，候補作業も含めて，重なっている作業の中から，次式で計算される優先度 Θ の最も高い作業を優先して処理することとする．

$$\Theta = w_1 p_u + w_2 e_u + w_3 b_u + w_4 c_u + w_5 a_u \tag{4.26}$$

ただし $w_1, w_2, \cdots, w_5$ は重み係数であり，各作業 O_u について，

p_u : 作業処理時間，
e_u : 最大後続時間，
b_u : バッファ占有率，
c_u : 最大後続作業数，
a_u : 目標 (必要処理回数) 達成率

とする．最大後続時間 (数) とは，その作業を始めてから最終作業を終えるまでに最小限必要な時間 (作業数) であり，バッファ占有率とは，その作業の直後にあるバッファ内の半製品数をその容量で除した値 (複数の場合はその最大値) である．

4.6 計算例および考察

4.6.1 一括スケジューリング

分枝限定法

まず，3機械5作業の問題例(図4.13)に対して，バッファ容量を(最終作業直後のバッファ(容量 ∞)を除き)均一に2として最終製品の所要数を変化させ，3種類のモデル上での分枝限定法を用いて最適スケジュールを求めた[1]．計算時間の比較を図4.14に示す．図より，以下のことがわかる．

(a) タイム・ペトリネット(TPN)モデルでは，探索範囲を遅れなしスケジュールに限定することにより，他の2つのモデルよりも計算時間が短縮される．また，最終製品所要数を大きくした場合の計算時間の増加も緩やかである．

(b) 選択グラフ・モデルとガントチャート・モデルは，ともにアクティブ・スケジュールを対象とするものであるが，スケジュール生成過程の違いによって，選択グラフ・モデルのほうが計算時間が短い．

次に，4機械10作業の例題(図4.15)に対して最終製品の所要数を1から9に変化させ，TPNモデル上での分枝限定法により求解を行った．このときのバッファ容量(全バッファで均一)と最大完了時間 C_{max} との関係を図4.16に示す．図より，

(a) バッファ容量を大きくすることによって C_{max} を短縮できること

がわかる．これは，仕掛り品を一時保持するためのバッファを設けることにより，それぞれの作業を処理できる時間帯が広くなって機械の遊休時間が減少するためである．さらにこの結果より，最終製品を複数個まとめて生産する場合に，バッファを考慮してスケジュールを作成することが有効であると言える．また，

(b) バッファ容量を増すことの C_{max} 短縮に及ぼす効果は，容量増大につれて逓減すること，

(c) C_{max} を最小にするために必要な最小限のバッファ容量，すなわちコストやスペースの増加を伴う不必要なバッファを設けなくてもよいという意味での，最適バッファ容量が存在すること(図中に ○ 印で示す)，

(c) 最終製品所要数が異なると，この最適バッファ容量も異なること，

が確認できる．

[1] タイム・ペトリネット・モデルの場合，最適スケジュールが得られる保証はないが，ここではすべての場合について最適スケジュールが得られた．

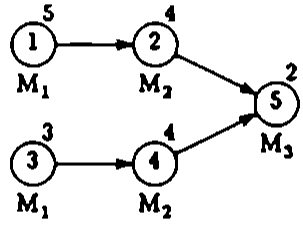


図 4.13 3 機械 5 作業の問題例

作業直後のバッファ ● を省略してある.

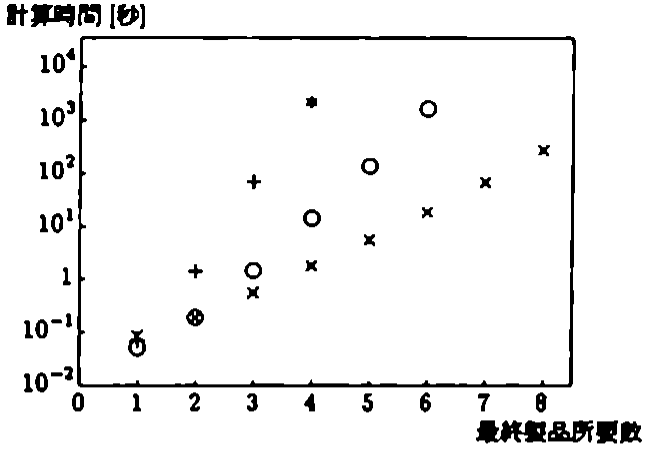


図 4.14 分枝限定法に要する計算時間の比較

記号 ×, ○ および + はそれぞれ, タイム・ペトリネット・モデル, 選択グラフ・モデルおよびガントチャート・モデルによる結果を示す. 分枝限定法については, すべてのモデルで深さ優先探索を利用した. また, 記号 • は, ガントチャート・モデル上での分枝限定法で 200,000 頂点を探索するまでに要した時間を表す.

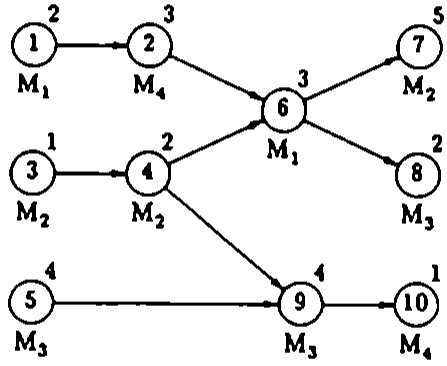


図 4.15 4 機械 10 作業の問題例

作業直後のバッファ ● を省略してある.

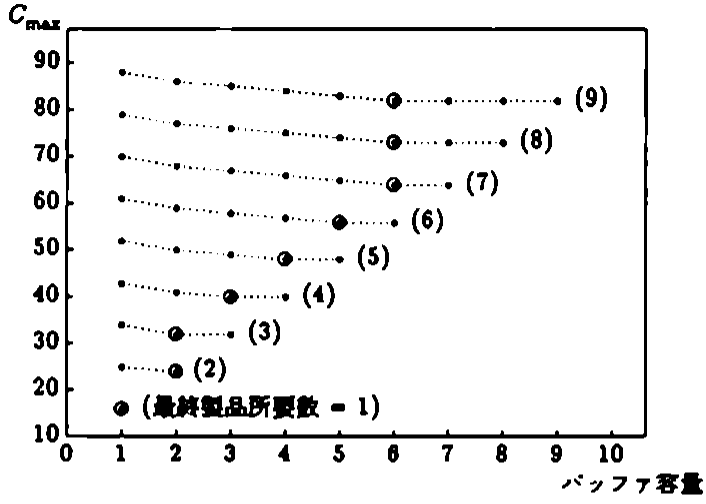


図 4.16 分枝限定法による結果

タイム・ペトリネット・モデル上での分枝限定法により得られる最大完了時間 C_{max} を表す. 記号 ○ は各所要数について, C_{max} を最小にするための必要最小限のバッファ容量を示す.

リスト・スケジューリング法

リスト・スケジューリング法においては，式 (4.26) の重み $w_1, \cdots, w_5$ を適当に設定することによって種々の優先規則を構成することができるが，ここでは，$w_1, \cdots, w_5$ のうち 1 つだけが 0 でなく，他の 4 つはすべて 0 であるような，

- R_1： 処理時間最小作業優先規則 ($w_1 = -1$)，
- R_2： 最大後続時間最大作業優先規則 ($w_2 = 1$)，
- R_3： バッファ占有率最小作業優先規則 ($w_3 = -1$)，
- R_4： 最大後続作業数最大作業優先規則 ($w_4 = 1$)，
- R_5： 目標達成率最小作業優先規則 ($w_5 = -1$)

の 5 種類に限定する．

仕事間に図 2.2 に示すような 4 種類の先行関係制約 (一般型，入木型，出木型，制約なし) を有する 13 仕事の問題で，

仕事に含まれる作業数 ：2 ～ 4，
作業が処理される機械 ：M_1 ～ M_6，
作業の処理時間 ：1 ～ 12，
仕事間の先行関係制約数：12 あるいは 0

を一様乱数によって決定した問題例をそれぞれ 100 個用意し，最終製品所要数を 10 および 50，バッファ容量を 1 ～ 10 として計算を行った．それぞれの場合について，100 個の問題例に対する C_{max} の平均値を求めた結果を図 4.17 に示す．図より，

(a) バッファ容量を大きくすることにより C_{max} が必ずしも減少するとは限らない，

ことが確認できる．優先規則によっては，直後のバッファが空いている限り特定の作業を繰り返して処理することがあり得るので，このような現象が生じることになる．最適スケジュールと比較した場合，この点が，リスト・スケジューリング法によって得られるスケジュールの顕著な性質であると言える．さらに，

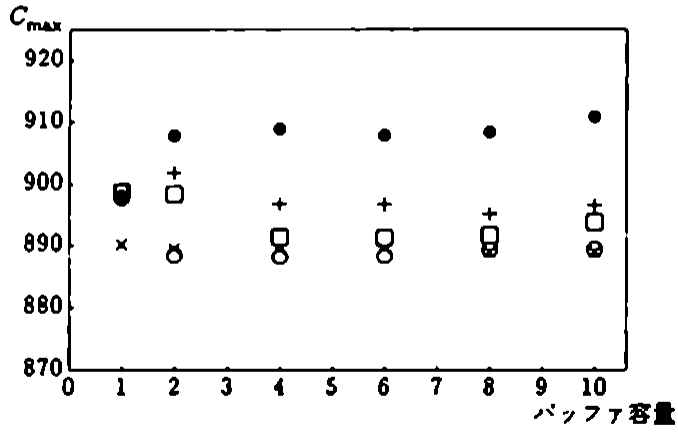
(b) C_{max} を小さくするには，R_3 および R_5 が比較的有効である，

(c) R_3 により得られるスケジュールの C_{max} は，バッファの大小にほとんど無関係である，

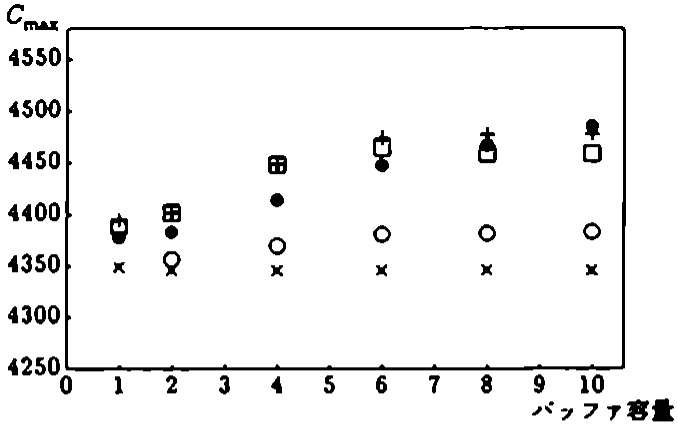
ことがわかる．R_3 および R_5 で得られるのは繰返しに近いスケジュールであり，C_{max} を小さくするには繰返しスケジュールが有効であると推測される．なお，図 4.17 の結果をもとにして，C_{max} の最小化に有効な優先規則を整理した結果を表 4.2 に示しておく．

次に，得られるスケジュールにおけるバッファの平均占有率，段取替え回数，および作業開始時刻の分布を調べたところ，以下のことがわかった．

(d) C_{max} の最小化に有効な R_3, R_5 を用いると段取替え回数が増える．



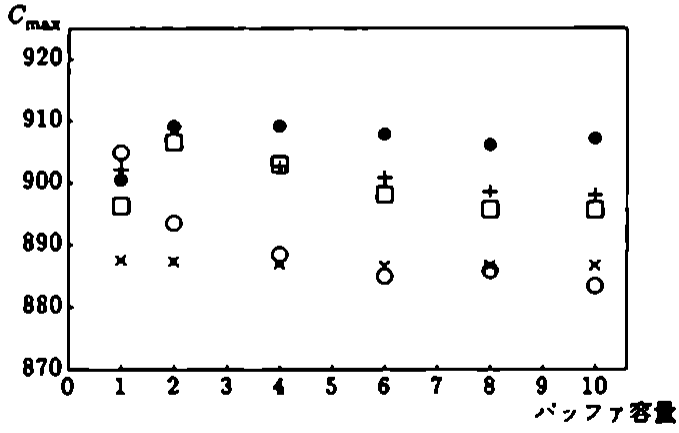
(a) 一般型先行関係，最終製品所要数 = 10 の場合



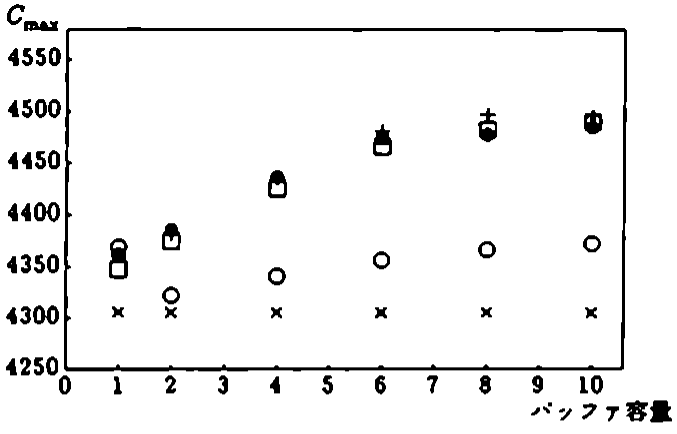
(b) 一般型先行関係，最終製品所要数 = 50 の場合

図 4.17 リスト・スケジューリング法による結果

記号 ●, □, ○, + および × はそれぞれ，R_1, R_2, R_3, R_4 および R_5 を用いた場合に得られる最大完了時間 C_{max} を表す．



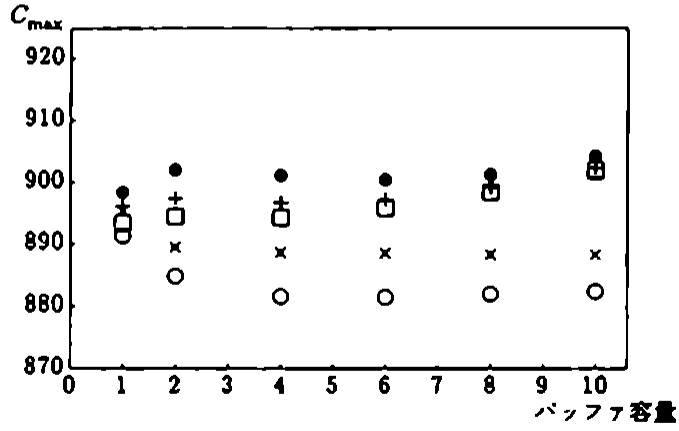
(c) 入木型先行関係，最終製品所要数 = 10 の場合



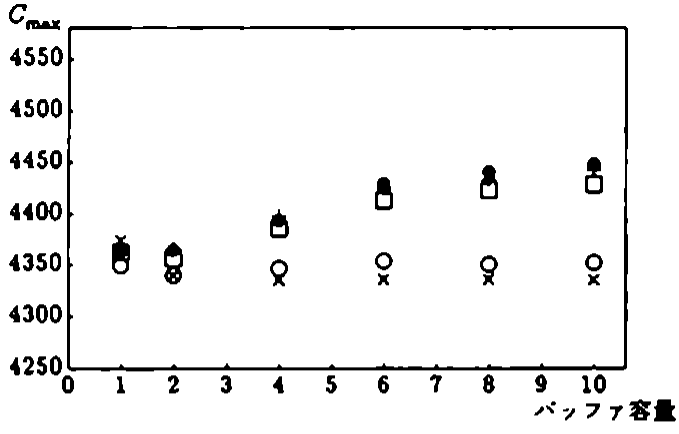
(d) 入木型先行関係，最終製品所要数 = 50 の場合

図 4.17 (続き)

記号 ●, □, ○, + および × はそれぞれ，R_1, R_2, R_3, R_4 および R_5 を用いた場合に得られる最大完了時間 C_{max} を表す．



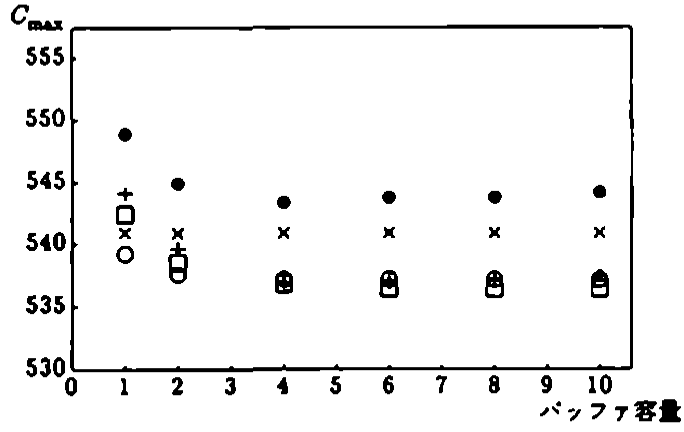
(e) 出木型先行関係，最終製品所要数 = 10 の場合



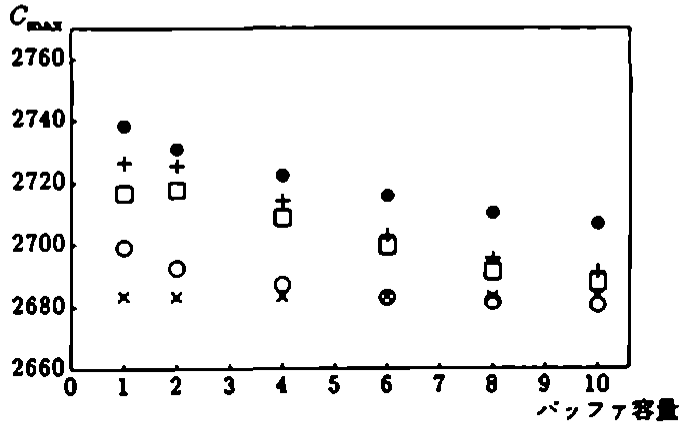
(f) 出木型先行関係，最終製品所要数 = 50 の場合

図 4.17 (続き)

記号 ●, □, ○, + および × はそれぞれ，R_1, R_2, R_3, R_4 および R_5 を用いた場合に得られる最大完了時間 C_{max} を表す．



(g) 先行関係なし，最終製品所要数 = 10 の場合



(h) 先行関係なし，最終製品所要数 = 50 の場合

図 4.17 (続き)

記号 ●, □, ○, + および × はそれぞれ，R_1, R_2, R_3, R_4 および R_5 を用いた場合に得られる最大完了時間 C_{max} を表す．

表 4.2 C_{max} の最小化に有効な優先規則

先行関係制約の型	最終製品所要数/バッファ容量	
	大	小
一般型	R_3, R_5	R_3, R_5
入木型	R_5	R_3, R_5
出木型	R_3, R_5	R_3
なし	R_3, R_5	R_2, R_3

(e) R_2, R_4 を用いるとバッファ占有率が高くなる．すなわち，生産途中で機械の故障などが起こった場合でも，バッファ内の半製品を使って後続する作業を滞りなく行うことが可能となる．

(f) C_{max} および段取替え回数の比較では同じような傾向を示した R_3 と R_5 は，ともにすべての作業を平均的に処理していくものであるが，この傾向は R_5 のほうが顕著である．

以上，バッファを考慮したスケジューリング問題に対する 5 種類の優先規則を取り上げ，それらの特徴を計算例を通して調べた．現実の問題に適用する場合には，ここで調べた特徴を考慮した上で，式 (4.26) の重み係数の複数個を非零とする複合優先規則や時刻や状況によって優先度を変化させるルールなど，より複雑な優先規則を構築することが必要になると考えられる．

4.6.2 繰返しスケジューリング

6 機械 32 作業の問題例 (図 4.18) に対して，バッファ容量を均一に 2，繰返しセット数の上限を 5 と設定し，TPN モデル上での分枝限定法を適用して繰返しスケジュールを求めた結果を表 4.3 に示す．なお繰返し時間パターンとしては，最終製品所要数 D を 1 ～ 20 として求めた一括スケジュールの終了時間パターンから 10 個を選んで用いることにした．さらに機械ごとに，

(1) 負荷時間 t_L :
その機械上での作業処理時間の和，

(2) 最小滞留時間時間 t_F :
その機械上での最初の作業が開始されてから，最後の作業が完了するまでに必要な時間

を定義し (ともに 1 セット分の作業を対象とする)，機械に関する最大値 t_L^* および t_F^* を求める．このとき，t_L^* は 1 セット当りの繰返し周期の下限値 (繰返しセット数や繰返し時間パ

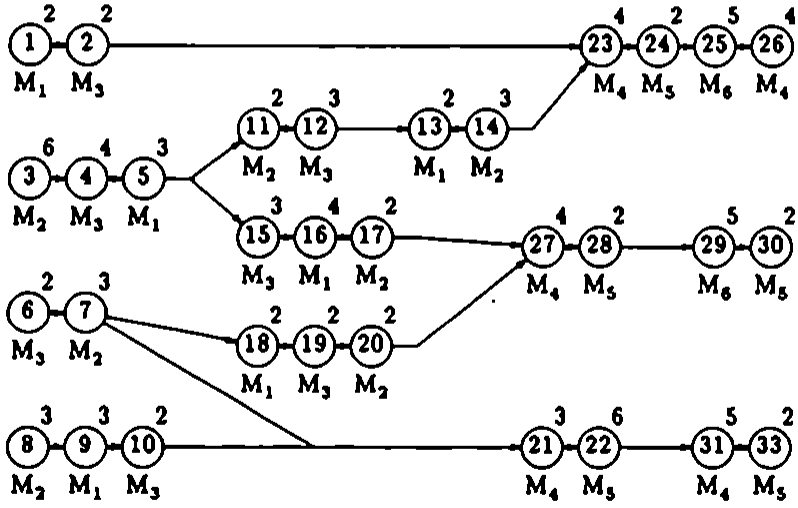


図 4.18 6 機械 32 作業の問題例

作業直後のバッファ ● を省略してある.

表 4.3 繰返しスケジューリングによる結果

繰返しパターン	1 繰返し周期に含まれるセット数 (S)	繰返し周期 (T)	1 セット当りの繰返し周期 (T/S)
1, 2, 3	2	40	20
4, 5, 6	3	60	20
7, 8, 9	4	80	20
10	5	100	20

繰返しパターンは, 繰返し周期 (S) の非減少順に番号付けしてある.

ターンに無関係)，t_P^* は 1 セット周期の繰返しスケジュールにおける繰返し周期の下限値をそれぞれ与えることになる．図 4.18 の問題例については，$t_L^* = 20$，$t_P^* = 25$ である．表より，以下のことがわかる．

(a) すべての繰返し時間パターンに対し，1 セット当りの繰返し周期の限界を実現する繰返しスケジュールが得られている．すなわち，複数セット周期の繰返しスケジューリングによって，最も負荷の高い機械の遊休時間を 0 にすることができる．

(b) この問題に対しては，2 セット周期の繰返しスケジュールにより，1 セット周期での繰返しスケジュールでは不可能な繰返し周期を実現している．

これらの結果は，1 セットでなく複数セットの繰返しスケジューリングの有効性を示しているとともに，所要数が多くなった場合には，膨大な時間をかけて一括スケジュールを求めなくても，いくつかのセットに分割して繰返しスケジュールを求めればよいことを表していると言える．ただし，繰返しスケジューリングの有効性が問題例ごとに異なることは直観的にも明らかであり，その適用に当たっては，問題の構造をあらかじめ調べておく必要がある．

4.7 むすび

本章では，ジョブショップ型の生産工程における仕掛り品在庫バッファを考慮したスケジューリング問題を定義するとともに，選択グラフ，タイム・ペトリネット (TPN)，およびガント・チャートによるモデル化手法を提案した．選択グラフ・モデルによれば，バッファを考慮した問題に対して従来のスケジューリング理論における既存解法の適用が可能となる．TPN モデルでは，一括スケジューリングおよび繰返しスケジューリングのための分枝限定法を構成した．ガントチャート・モデルは，スケジュール生成過程がわかり易いという特徴を有するものであり，このモデルを用いて，分枝限定法およびリスト・スケジューリング法を構成した．

計算例を通して，まず分枝限定法に要する計算時間については，探索範囲を遅れなしスケジュールに限定することにより，TPN モデルでは他の 2 つのモデルよりも計算時間が短縮されることがわかった．次に，バッファのスケジュールに及ぼす効果を調べたところ，一括スケジューリングでは，バッファを考慮することによって最大完了時間 C_{max} の短縮が可能であること，および C_{max} を最小にするために必要な最小限のバッファ容量 (最適バッファ容量) が存在することが確認された．繰返しスケジューリングに関しては，複数セット周期の繰返しスケジュールを求めることにより，問題例によっては最も負荷の高い機械の遊休時間を 0 にできること，および 1 セット当りの繰返し周期を，1 セット周期の繰返しスケジューリングでは実現不可能な値まで短縮できることを確認した．最後に，リスト・スケジューリング法における典型的な 5 つの優先規則の比較を行ったところ，最大完了時間最小化には，バッファ占有率最小作業優先規則 (R_3) および目標達成率最小作業優先規則 (R_5) などの，す

べての作業を平均的に処理していく優先規則が比較的有効であることがわかった.

今後の課題として, 4.6.1 に述べた意味でのバッファ容量の最適化手法の開発, バッファの C_{max} 短縮に対する効果のみでなく機械故障に対するスケジュールのロバスト性増大に及ぼす効果の検討, およびリスト・スケジューリング法における複合優先規則の比較・検討などが挙げられる.

第 5 章

分解による近似解法

5.1 まえがき

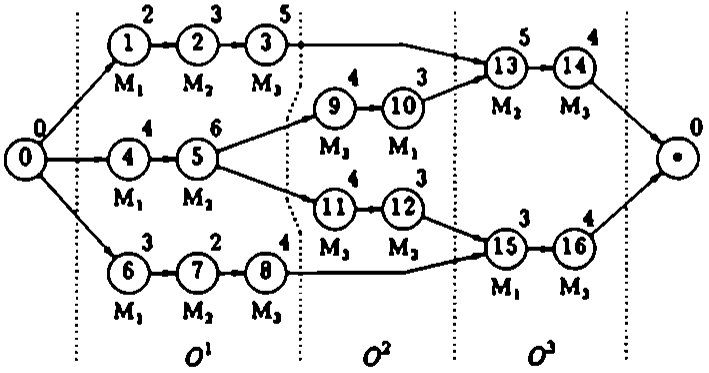
スケジューリング問題は一般に離散的な変数を扱う組合せ最適化問題であり，その求解には多大な量の計算を必要とする．現実に発生する規模のスケジューリング問題のほとんどは，最適ではないが実用的な解すなわち近似解を求めることすら難しいのが現状である．このような場合に有効な方法として，分解による近似解法があるが，スケジューリング問題に関してはほとんど報告されていない[33, 34]．そこで，本章ではジョブショップ・スケジューリング問題で最大完了時間 C_{max} を最小化する場合を取り上げて，まず，問題の求解困難さを表す指標として「難易度」を定義・数量化し，その上で，部分問題の難易度がほぼ等しくなるような分解アルゴリズムを提案する[35]．さらに，計算例によって分枝限定法やリスト・スケジューリング法と比較することにより，難易度に基づく分解法の有効性を示す．

5.2 分解による近似解法

ジョブショップ問題は一般に NP 完全な問題であるので，問題の規模が大きくなると求解に要する時間は指数関数的に増加し，たちまち求解が困難となる．そこで，分解による近似解法 (分解法)，すなわちジョブショップ問題 (原問題) を構成する作業集合をいくつかの部分集合に分割し，それぞれの部分集合に含まれる作業で構成される問題 (部分問題) を解くことにより，全体のスケジュールを作成する方法が有効となってくる．以下では図 5.1 のように，与えられた先行関係制約に基づいて時系列的に問題を分解する方法について検討する．

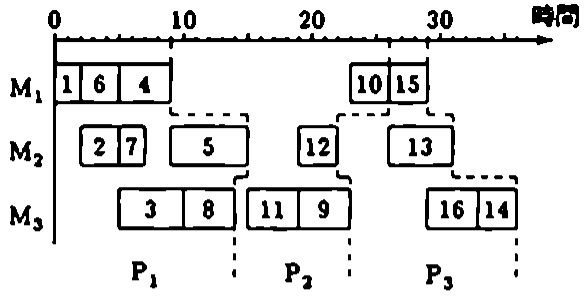
分解法では，規模の小さい問題を個々に解くことになるから，原問題を一括して解くのに比べて計算時間が短縮される．しかし，各部分問題について単独に最適化を行うため，全体としての最適解が得られるとは限らない．また，極端に小規模な部分問題に分解すると解の質を低化させる可能性が増す．したがって，分解法においては，各部分問題が同程度に難しく，各部分問題の求解に要する計算量が現実の許容最大量 (許容時間，計算機の能力などによる) であれば，最も効率よく分解されたと言える．

以上の考察より，ジョブショップ問題に対して特定の分枝限定法 (コンフリクト解消法) を



(a) 部分集合への分割

作業全体が3つの部分集合に分割されていることを示す.



(b) 部分問題の求解

それぞれの部分集合に含まれる作業から構成される部分問題 P_k は, k の増加順に解かれる.

図 5.1 部分問題への分解の例

適用する場合に限定して，まず問題の難易度を数量化し，次にこの難易度に基づいた分解アルゴリズムを提案する．

5.3　難易度の数量化

1 台の機械で 2 つ (以上) の作業が同時に行われるようなスケジュールは実行可能ではない．このとき，これらの作業がコンフリクトを起こしているという．コンフリクト発生の可能性が高い問題は，それだけ作業の順序に自由度があり，解きにくいと考えられる．そこで，コンフリクト発生の度合いを問題の求解困難さの指標と考え，問題の難易度と呼ぶ．以下にその数量化の一方法を示す．

1° 作業 O_u の最大先行時間 h_u，最大後続時間 e_u から，各作業の最早開始時刻 t_u^E および最遅終了時刻 t_u^L を求める[1]．

$$t_u^E = h_u \tag{5.1}$$

$$t_u^L = h_* - e_u + p_u \tag{5.2}$$

さらに，あらかじめ (h_u/e_u) の非減少順に作業番号を付け替えておく．

2° 機械 M_i 上でのコンフリクト発生の可能性

$$c_i = \sum_{\substack{\mu_u=1 \\ v>u}} \sum_{\mu_v=1} \left(\frac{d_{uv}^I + d_{uv}^S}{p_u + p_v} \cdot \frac{2N_n}{u+v} \right) \tag{5.3}$$

を計算する．ここに，

$$d_{uv}^I = \max\{0,\ p_u + p_v - \max(\Delta_{uv},\ \Delta_{vu})\} \tag{5.4}$$

$$d_{uv}^S = \min(\Delta_{uv},\ \Delta_{vu}) \tag{5.5}$$

である．ただし，$\Delta_{uv} = t_u^L - t_v^E$ とする．

3° 機械 M_i について，処理 (使用) 頻度 f_i を計算する：

$$f_i = \frac{\sum_{\mu_u=1} p_u}{\sum_u p_u}. \tag{5.6}$$

4° 次式により難易度 d を求める：

$$d = \sum_i (f_i c_i). \tag{5.7}$$

---

[1] 最大先行時間 h_u とは，スケジュール開始時点から O_u が開始されるまでに最小限必要な時間のことで，最大後続時間 e_u とは，O_u を開始してからすべての作業を完了するまでに最小限必要な時間のことである．問題を有向グラフで表現した場合，節点 0 から節点 u および節点 u から節点 $*$ に至るクリティカル・パスの長さが，それぞれ h_u，e_u に等しくなる．

ここで，難易度を計算するために定義した変数について説明する．まず，1° は準備である．最早開始時刻 t_u^E で表されるスケジュール (最早開始スケジュール) は，機械条件すなわち機械は同時に複数の作業を処理できないという条件を無視したものである．このときの最大完了時間 C_{max} $(= t_*^E = t_*^L)$ を変化させない範囲内で，各作業をなるべく遅く始めるようなスケジュール (最遅終了スケジュール) での作業の終了時刻が最遅終了時刻 t_u^L を与える．さらに，作業 O_u を t_u^E と t_u^L の間に処理するようなスケジュール (実行可能であるとは限らない) の全体を初期スケジュールと呼ぶことにする．

2° では各機械上でのコンフリクトの可能性を計算する．コンフリクト解消法は，最早開始スケジュールから初めて，最も早い時刻にコンフリクトを起こしている作業対から順に順序付けを行っていく (コンフリクトを解消していく) 方法である．初期スケジュールにおいて，各作業 O_u の処理予定時間区間 Γ_u を $[t_u^E, t_u^L]$ と定義すると，Γ_u と Γ_v の重なり d_{uv} が大きいほどコンフリクトが起こりやすく，d_{uv} と問題の求解困難さとの間には正の相関があると予想される．そこで，d_{uv} の最大値 d_{uv}^S と最小値 d_{uv}^L を用いてコンフリクトを起こす可能性 c_i $(i = \mu_u = \mu_v)$ を定量化する．なお，係数 $2N_n/(u+v)$ は，2 作業 O_u および O_v の番号 (の平均値) が小さいほど，すなわちこれら 2 作業間でのコンフリクトが早い時刻に起こる可能性が高いほど，c_i への影響を大きくするためのものである．

3° で求める機械 M_i の使用頻度 f_i は $[0, 1]$ の範囲にあり，この f_i が大きいほど，c_i の難易度 d に対する影響が大きいと考えられる．

5.4 分解アルゴリズム

5.4.1 一括分解アルゴリズム D_1

一括分解アルゴリズムでは，原問題をいくつかの部分問題に分割し，その後これらの部分問題を順に解く．作業集合 O を q 個の部分集合 O^k $(k = 1, \cdots, q)$ に分解し，O^k に含まれる作業から構成される部分問題 P_k を k の増加順に解くことになる．

0° 各部分問題の相対難易度の上限値 r を設定する．

1° (h_u/e_u) の非減少順に作業番号を付け替え，$k = 1$, $O^1 = \{O_1\}$, $j = 1$ とする．

2° $j = N_n$ であれば $q = k$ として終了．そうでなければ $j = j + 1$ とする．

3° O_j を O^k に加えることにより (P_k の難易度 d_k) / (原問題の難易度 d) が r を越えなければ，$O^k = O^k \cup \{O_j\}$ として 2° へ．越えれば $O^{k+1} = \{O_j\}$ とする．

4° $k = k + 1$ として 2° へ．

5.4.2　逐次分解アルゴリズム D_2

部分問題への分解をその求解に先立って行うのではなく，1 つの部分問題の生成と求解を繰り返す方法を逐次分解アルゴリズムと呼ぶ．分解の指標として用いる難易度はコンフリクトの解消 (作業順序の決定) とともに変化する量であるから，以前の部分問題を解いた結果得られる作業順序を難易度の計算に反映させることによって，より正確な難易度を求めることができる．

0° 各部分問題の相対難易度の上限値 r を設定する．

1° (h_u/e_u) の非減少順に作業番号を付け替え，$k=1$, $\mathcal{O}^1=\{O_1\}$, $j=1$ とする．

2° $j=N_n$ であれば $q=k$ として終了．そうでなければ $j=j+1$ とする．

3° O_j を $\mathcal{O}^k$ に加えることにより (d_k/d) が r を越えなければ，$\mathcal{O}^k=\mathcal{O}^k\cup\{O_j\}$ として 2° へ．越えれば 4° へ．

4° 部分問題 P_k を解く．この結果を用いて O_u $(u=j+1,\cdots,N_n)$ の h_u と e_u を再計算し，(h_u/e_u) の非減少順に作業番号を付け替える．さらに，$k=k+1$, $\mathcal{O}^k=\{O_j\}$ として 2° へ．

5.4.3　反復分解アルゴリズム D_3

1 つの実行可能なスケジュールは，図 5.2 (a) の有向グラフに，そのスケジュールで定められる同一機械上での作業順序を追加することにより，図 5.2 (b) のような有向グラフで表現できる．このとき，最大完了時間 C_{max} は，有向グラフ (図 5.2 (b)) 上で，節点 0 から節点 $*$ に至るクリティカル・パスの長さとして与えられる．したがって，逐次分解アルゴリズムで問題を解いた後に，このクリティカル・パス上の作業を可能な限り前の部分集合へ移動し，更新された部分問題を解き直すと，クリティカル・パスが短縮されて C_{max} を減少させ得ると期待できる．そこで，反復分解アルゴリズムでは，作業の移動と部分問題の求解を C_{max} が減少しなくなるまで繰り返すことにする．

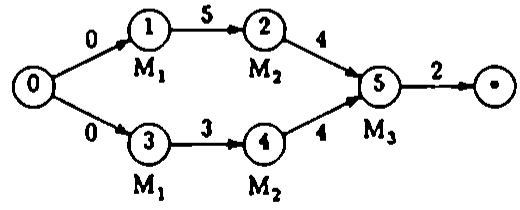
0° 各部分問題の相対難易度の上限値 r を設定する．

1° (h_u/e_u) の非減少順に作業番号を付け替え，$k=1$, $\mathcal{O}^1=\{O_1\}$, $j=1$ とする．

2° $j=N_n$ であれば $C^*_{max}=C_{max}$, $q=k$ として 5° へ．そうでなければ $j=j+1$ とする．

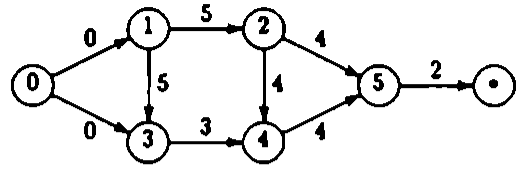
3° O_j を $\mathcal{O}^k$ に加えることにより (d_k/d) が r を越えなければ，$\mathcal{O}^k=\mathcal{O}^k\cup\{O_j\}$ として 2° へ．越えれば 4° へ．

4° 部分問題 P_k を解く．この結果を用いて O_u $(u=j+1,\cdots,N_n)$ の h_u と e_u を再計算し，(h_u/e_u) の非減少順に作業番号を付け替える．さらに，$k=k+1$, $\mathcal{O}^k=\{O_j\}$ として 2° へ．



(a) 問題の有向グラフ表現

仮の出発節点 0 と仮の終了節点 • を付加してある.



(b) 実行可能スケジュールの一例

機械番号は省略してある.

図 5.2 実行可能スケジュールの有向グラフ表現

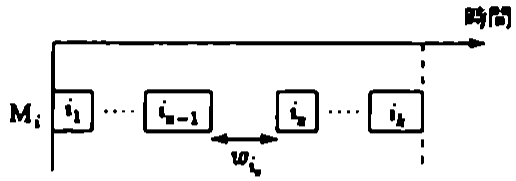


図 5.3　割り込み法

部分問題 $P_1 \cdots P_k$ の求解が終わった状態を示す．w_{i_u} は作業 O_{i_u} を開始する直前の機械 M_i の遊休時間を表す．

5° クリティカル・パス上の作業を 1 つ前の部分集合に移動して，部分問題 $P_1, \cdots, P_q$ を再定義し，これらの部分問題を順に解く．6° へ．

6° $C_{max} < C^*_{max}$ であれば $C^*_{max} = C_{max}$ として 5° へ．そうでなければ終了．

5.5　部分的改良法

5.5.1　割り込み法

5.4 で述べた分解アルゴリズムは，いずれも部分問題を前から順に解いていくものである．この際，部分問題 P_k を解いた結果のうち，部分問題 P_{k+1} の求解にあたって利用されるのは機械の空き時刻のみであるので，全体として求められるスケジュールはセミアクティブ・スケジュールであるが，アクティブ・スケジュールである保証はない[1]．以下で述べる割り込み法は，機械の使用状況を併せて利用することによって，アクティブ・スケジュールが得られるように工夫し，最大完了時間の短縮を図るものである．

ある機械 M_i に注目し，部分問題 P_k の求解が終わった時点で，M_i 上での作業処理順序が $O_{i_1}, O_{i_2}, \cdots, O_{i_k} \in \bigcup_{l=1}^{k} \mathcal{O}^l$ であったとする (図 5.3)．このとき，

(1) $O_{i_a}, \cdots, O_{i_k}$ が O_u の先行作業でない，

(2) O_{i_u} の直前の空き時間 w_{i_u} が p_u 以上である，

という 2 つの条件を満たす作業 $O_u \in \mathcal{O}^{k+1}$ および $O_{i_u} \in \mathcal{O}^k$ ($\mu_u = \mu_{i_u} = i$) を探し出し，この O_u を O_{i_u} の直前で処理する (割り込ませる) ことによって，最大完了時間 C_{max} を短縮できる可能性がある．そこで割り込み法では，各機械および各作業 $O_r \in \mathcal{O}^{k+1}$ について，上の手続きを繰り返すものとする．

[1] 部分問題ごとには，最適スケジュール (アクティブ・スケジュールの 1 つ) が得られている．

5.5.2 難易度の均一化

部分問題を生成するに当たって，各部分問題の難易度にはばらつきが少ないほうが望ましい．既に述べた一括分解法や逐次分解法において，すべての部分問題の難易度はその上限設定値 $\bar{r}d$ を越えることはないが，最後尾の部分問題 P_q の難易度 d_q が極端に小さくなることがある．P_q を構成するのは，最後に残った作業の集合となるからである．ここで述べる難易度の均一化とは，d_q が極端に小さい場合に，すべての部分問題で難易度がほぼ均一となるように作業数を調整することが目的である．なお，再分解を実行するか否かを $d_q/(\bar{r}d)$ の値によって決めるものとし，分解アルゴリズムを開始する前に基準となる定数 δ $(0 \leq \delta < 1)$ をあらかじめ設定しておくことにする．

P_q が生成された時点で，$d_q/(\bar{r}d)$ が δ を下回っていたとする．このとき，部分問題 $P_q, \cdots, P_{q-1}$ に含まれる作業の数を均等に増やし，全体として $(q-1)$ 個の部分問題に分解し直す．具体的には，P_1 に含まれる作業の数 N_q より，

$$\alpha = \left\lfloor \frac{N_q}{q-1} \right\rfloor \tag{5.8}$$

$$\beta = N_q - \alpha(q-1) \tag{5.9}$$

を求める ($\lfloor x \rfloor$ は x 以下の最大の整数)．そして，P_i $(i = 1, \cdots, \beta)$ については，

$$N' = N' + \alpha + 1, \tag{5.10}$$

P_i $(i = \beta + 1, \cdots, q-1)$ については，

$$N' = N' + \alpha \tag{5.11}$$

として，この更新された作業数 N' を基準に (難易度を無視して) 部分問題を再構成する方法を難易度の均一化法と呼ぶ．この方法を併用することにより，中途半端な分解で解を悪化させてしまう状況を回避することが可能となる．

5.6 計算例および考察

仕事間に一般型 (2.2 における β = prec) の先行関係制約を有する 13 仕事の問題で，

仕事に含まれる作業数 : 2 ~ 5
作業が処理される機械 : M_1 ~ M_6
作業の処理時間 : 1 ~ 12
仕事間の先行関係制約数 : 12

を一様乱数によって決定した 50 個の例題に対する計算例を通して，難易度に基づく分解法の有効性を評価する．

5.6.1 分枝限定法との比較

ここでは，最適解 (最適スケジュール) を求めるための分枝限定法を比較対象として取り上げる．なお，分解法による結果を評価するための尺度として，スケジュールの良さ R_c と求解に要する計算時間の比 R_t を，次のように定義しておく．

$$R_c = \frac{\text{分解法を適用した場合の } C_{\max}}{\text{分枝限定法を適用した場合の } C_{\max} \text{ (最適値)}} \tag{5.12}$$

$$R_t = \frac{\text{分解法による計算時間}}{\text{分枝限定法による計算時間}} \tag{5.13}$$

なお，式 (5.13) での分解法を適用した場合の計算時間には，難易度の計算など分解に要した時間も含めるものとする．また，分解法による求解に際しては，

- 部分問題 (P_1 を除く) の求解を始める時点で割り込み法を適用する，
- 難易度を均一化する場合に基準となる定数 δ は 0.6 とする，

こととした．

表 5.1 および図 5.4 に，3 種の分解アルゴリズムで，分解の基準となる部分問題の相対難易度の上限値 r を，0.4, 0.2, 0.1 および 0.05 に設定した場合の計算結果を示す．表 5.1 および 図 5.4 より，

(a) 難易度の上限値を低く設定するほど，計算時間は短縮されるがスケジュールの良さは低下する

ことが確認できる．これは，直観的にも明らかな結果であり，換言すれば 5.3 で定義した難易度の妥当性を示すものであると言える．また，

(b) 一括，逐次，反復と分解のアルゴリズムを複雑化していくにつれて，計算時間は長くなるが解は良くなる．

したがって，必要なスケジュールの良さ (精度) および許容される計算時間を考慮した上で，これらの方法を使い分けることが可能である．

5.6.2 リスト・スケジューリング法との比較

リスト・スケジューリング法は，何らかの基準 (優先規則) に基づいて作業に優先順序を付け，この順に空いている機械に割り当てていく方法であり，実際のスケジューリングで最もよく用いられている近似解法 (ヒューリスティクス) の 1 つである (3.2.4 参照)．リスト・スケジューリング法では，優先規則の違いによって多種多様なアルゴリズムを実現できるが，ここでは，

(1) 処理時間最小作業優先規則 (SPT 規則)，

(2) 最大後続時間最大作業優先規則 (MST 規則)

表 5.1 分枝限定法との比較

(a) 一括分解法 D_1

t	R_c	R_t
	平均値 (最良値-最悪値)	平均値 (最良値-最悪値)
0.4	1.022 (1.000-1.156)	0.719 (0.0027-1.5488)
0.2	1.048 (1.000-1.182)	0.104 (0.0030-0.4325)
0.1	1.072 (1.000-1.250)	0.072 (0.0009-0.2807)
0.05	1.074 (1.000-1.189)	0.059 (0.0006-0.1986)

(b) 逐次分解法 D_2

t	R_c	R_t
	平均値 (最良値-最悪値)	平均値 (最良値-最悪値)
0.4	1.004 (1.000-1.115)	1.016 (0.0028-1.5560)
0.2	1.040 (1.000-1.144)	0.236 (0.0047-1.3305)
0.1	1.070 (1.000-1.246)	0.113 (0.0008 1.3836)
0.05	1.075 (1.000-1.212)	0.007 (0.0010-0.2600)

(c) 反復分解法 D_3

t	R_c	R_t
	平均値 (最良値-最悪値)	平均値 (最良値-最悪値)
0.4	1.004 (1.000-1.115)	1.035 (0.0142-1.5560)
0.2	1.031 (1.000-1.142)	0.354 (0.0100-1.3305)
0.1	1.055 (1.000-1.153)	0.251 (0.0071-1.3836)
0.05	1.065 (1.000-1.183)	0.160 (0.0081-0.5188)

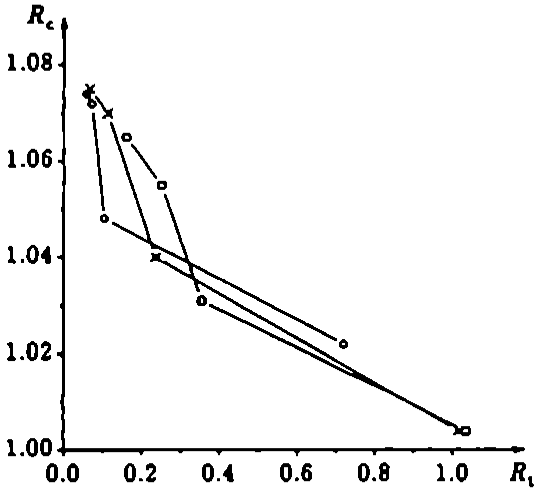


図 5.4　分枝限定法との比較

記号 ○, × および □ はそれぞれ，アルゴリズム D_1, D_2 および D_3 を 50 個の例題に適用した場合の結果の平均値を示す．

表 5.2 リスト・スケジューリング法との比較

方法	R_c	R_t
	平均値 (最良値-最悪値)	平均値 (最良値 最悪値)
分解法 (アルゴリズム D_3, $r = 0.2$)	1.031 (1.000-1.144)	0.637 (0.135-1.132)
リスト・スケジューリング法 (SPT 規則)	1.152 (1.000-1.343)	0.147 (0.023 0.254)
リスト・スケジューリング法 (MST 規則)	1.067 (1.000-1.235)	0.141 (0.021-0.246)

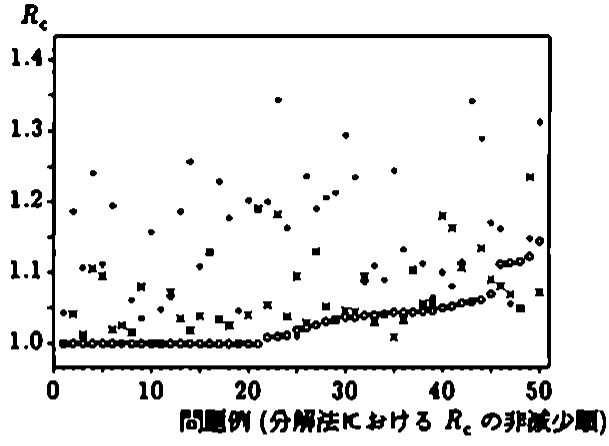


図 5.5 リスト・スケジューリング法との比較

記号 ○, + および × はそれぞれ, 分解法 (アルゴリズム D_3, $r = 0.2$), リスト・スケジューリング法 (SPT 規則) およびリスト・スケジューリング法 (MST 規則) による結果を示す.

の 2 つを用いることにした．

分解法とリスト・スケジューリング法との比較結果を表 5.2 および図 5.5 に示す．なお，分解法での相対難易度の上限値 ρ は 0.2 とし，またリスト・スケジューリング法についても，分解法の場合と同様に R_c および R_t を定義した．次のことがわかる：

(a) 分解法のほうが計算に多くの時間を要しているが，全般的に良い解が得られる．

(b) リスト・スケジューリング法では最適解の得られている問題例が少ない．

以上より，難易度に基づく分解法の，

(c) 計算時間をある程度必要とするが，最適に近いスケジュールが得られる，

(d) 難易度の上限値をパラメータとして，計算時間およびスケジュールの良さを調整できる，

という特徴を確認することができた．

5.7　むすび

問題の求解困難さを表す難易度を定義・数量化し，次に，部分問題の難易度がほぼ等しくなるような分解アルゴリズムを提案する．

本章では，まず問題の求解困難さを表す難易度を数量化し，次に部分問題の難易度がほぼ等しくなるような 3 種の分解アルゴリズム，すなわち一括，逐次，および反復分解アルゴリズムを提案した．計算例より，難易度と求解に必要な計算時間には高い相関があること，難易度が著しく減少するような分解を行うとスケジュールの良さが低下すること，および一括，逐次，反復と分解アルゴリズムを複雑化していくにつれてスケジュールが良くなることが明らかにされた．

今後，大規模な問題に対する分解法の有効性の評価，より正確な難易度の推定法，および最適スケジュールが得られるような分解 (最適分解) 方法の開発などについての検討が必要である．

第 6 章

遺伝アルゴリズムによる近似解法

6.1 まえがき

近年のシステム工学においては，極めて多量のしかも往々にして不確実あるいは不完全な，さらには時変的な情報に対処する新しい方法論が求められている．このような要求に対して，生体・生物の持つ独特の機能，すなわち環境適応的に自らのシステムを形成し，改良していく機能 (自己組織化機能) に注目して，その仕組みを工学的な課題に応用しようとする試みが活発化している．例えば，脳や神経回路網 (ニューラル・ネットワーク) における適応的で高度に並列化された情報処理方式の仕組みを解明し，それを最適化問題の求解に応用する手法の研究[36, 37]，および遺伝子情報に基づき，環境適応的である生物の進化過程を模擬した遺伝アルゴリズム (2.4.3 参照) を，最適化計算やルールベース・システムの学習などに適用しようとする研究[38] などである．

本章では，上述の生物機能的な方法論のうちで遺伝アルゴリズム (GA) を取り上げ，まずジョブショップ問題に対する GA の一構成方法を示す[39]．GA の構成法についてはこれまでにも報告されているが[40, 41, 42]，ここでは，問題の選択グラフ表現に基づく方法を示すとともに，種々の解法との比較を通してその有効性を評価する．次に，GA の問題点の 1 つである初期収束 (premature convergence) 現象を抑制して個体群の多様性を維持するために，近傍を考慮した GA の一並列化手法 (近傍モデル) を導入し，その解の良さおよび求解時間短縮に及ぼす効果を調べる[43, 44, 45]．

6.2 遺伝アルゴリズム

6.2.1 概要

2.4.3 でも述べたように，遺伝アルゴリズム (GA) は，自然界における生物の進化 (集団遺伝) モデル，すなわち世代を形成している個体の集合 (個体群) の中で，環境への適応度の高い個体が次世代により多く生き残り，また交叉および突然変異を起こしながら次の世代を形成していく過程を模した最適化法である[15, 16]．そこで，最適化問題の目的関数を適応度

に，解の候補を個体にそれぞれ対応させることになる．

6.2.2 基本構成

GA の基本的な構成について，以下に記す[16]．

(1) 個体の表現：

GA では，有限固定長の記号列を染色体に，個々の記号を遺伝子にそれぞれ対応させ，この記号列によって個体を表現する．記号列は，

$$1\ 0\ 0\ 1\ 0\ 1\ 1$$

のように 0, 1 のみの 2 値記号から構成されることが多い．

(2) 適応度の計算：

GA では，個体は非負の値をとる適応度により評価され，その最大化を図る．そこで，最適化問題の目的関数と適応度との対応規則を定めなければならない．

(3) 遺伝演算子の実現：

次の世代を決定するのが，遺伝演算子である．以下に，代表的なものを列挙する．

- Selection (選択あるいは淘汰)：
  世代 t の個体群 $\mathcal{P}(t)$ 中の各個体 x の適応度 $f(x)$ を計算し，次世代の個体群における個体 x の子 (offspring) の数の期待値を $f(x)/\bar{f}$ とする．ただし，$\bar{f}$ は個体群の適応度の平均値である．
- Crossover (交叉)：
  個体群内の個体をランダムにペアリングし，各ペアについて，(a) 分解位置 (cross-point) をランダムに 2 カ所選び，(b) 確率 p_c で部分記号列を入れ替える．一例を次に示す．

$$\begin{array}{ccc} 1\ 0\ |\ 0\ 1\ 0\ |\ 1 & & 1\ 0\ |\ 1\ 0\ 0\ |\ 1 \\ 1\ 1\ |\ 1\ 0\ 0\ |\ 0 & \longrightarrow & 1\ 1\ |\ 0\ 1\ 0\ |\ 0 \end{array}$$

- Mutation (突然変異)：
  各個体について，(a) 1 ビットをランダムに選び出し，(b) 確率 p_m で反転する．一例を次に示す．

$$1\ 0\ \underline{0}\ 1\ 0\ 1 \longrightarrow 1\ 0\ \underline{1}\ 1\ 0\ 1$$

(4) パラメータの設定：

GA を実現するに当たって，4 つのパラメータ：

N_g：計算終了までの世代数，

N_p : 個体群のサイズ (個体数),

p_c : 交叉を行う確率,

p_m : 突然変異を行う確率

の値をあらかじめ設定する必要がある. 以下では, これらのパラメータの設定による特定の GA の実現を, 4 項組 $G = (N_g, N_p, p_c, p_m)$ で表す.

(5) 初期個体群の生成 :
一様乱数を用いてランダムに記号列の値を決め, N_p 個の個体を生成する.

(6) 計算の全体的手順 :

1° パラメータを設定し, 初期個体群 $\mathcal{P}(1)$ を形成する. 世代 $t = 1$ とする.

2° $t < N_g$ であれば 3° へ. $t = N_g$ であれば 4° へ.

3° 遺伝演算子を適用して, 次世代 $t+1$ の個体群 $\mathcal{P}(t+1)$ を生成する. $t = t+1$ として 2° へ.

4° 終了.

6.2.3 GA の特徴

GA による最適値の探索には,

(1) 1 つの点 (個体) から他の点へと探索を進めるのではなく, 点の集合 (個体群) から集合へと探索を進めるので, 初期値に比較的依存しにくい,

(2) 適応度 (目的関数値) を利用するだけで勾配などの他の情報を使わないので, 目的関数の性質がよくわからないような問題についても (とりあえず) 適用できる,

(3) 確率的な遷移ルールに従って挙動するので, 局所最大点にとどまらずに大域的な最大点に到達し得る可能性が高い

といった特徴がある.

さらに, GA による計算の過程, すなわち適応度の高い個体がより多く生き残って次の世代を形成していく過程に注目すると, 個体を表現している記号列そのものだけでなく, 適応度を高くするような部分記号列, 例えば,

1 1 0 * * * *

(* は 0/1 のどちらでもよいことを示す) も同時に評価され, 個体群の中でこのような部分記号列の数が増えていくという見方もできる. 言い換えると, この部分記号列から構成される記号列全体が「裏で」同時並列的に評価されていることになる (implicit parallelism). そこで, この同時並列性がうまく働くように, 個体を表現し, また交叉演算子を構築することが重要となる.

一方，GA の問題点として指摘されているのは，計算の初期の段階で個体群中の個体の多様性が失われる初期収束 (premature convergence) 現象である．交叉演算を探索の中心におく GA において，探索空間の広さは個体の多様性によって決まるので，初期収束を避けるための工夫が必要となる．単純には p_c と p_m，特に p_m を比較的大きい値に設定しておくことによって，この初期収束現象の可能性を低減させることができるが，同時に，適応度の高い個体も消滅しやすくなってしまうのが難点である．

6.3 遺伝アルゴリズムの構成法

3.2.3 で記述したジョブショップ問題の選択グラフ表現に基づいて，遺伝アルゴリズム (GA) を構成する方法を示す．ここで，GA を構成するに当たって，6.2.2 における (1) と (2) の実現方法を決めなければならない．(3) ～ (6) は汎用的であるので，改めて規定する必要はない．

6.3.1 個体の表現方法 — encoding

問題の選択グラフ表現における選択弧対の解消方向を 0 と 1 で表し，この 0/1 の記号を選択弧対の数だけ並べた記号列で個体を表現することにする．たとえば，図 6.1 (a) の選択グラフでは，

$$\begin{array}{ccccccc} 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ d_{14} & d_{17} & d_{47} & d_{25} & d_{26} & d_{56} & d_{38} \end{array}$$

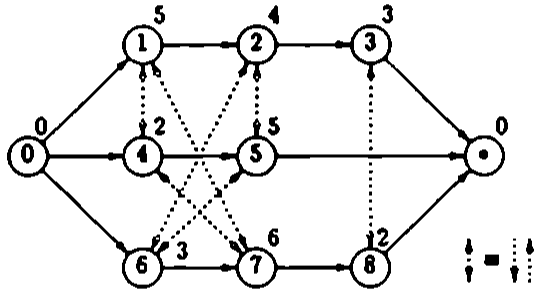
のように個体を表現することにする．なお，d_{ij} は節点 i, j 間の選択弧対を表す．選択弧対の解消方向は，0 が番号の小さい節点から大きい節点への有向弧を残すこと，1 がその逆とする．

6.3.2 適応度の計算方法 — decoding

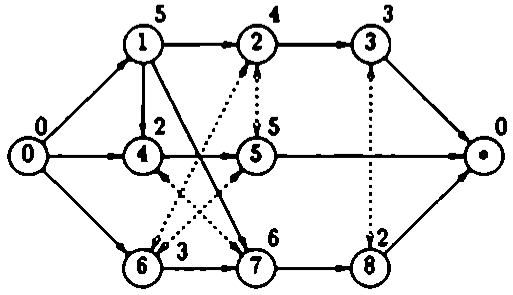
記号列からスケジュールに変換する際，単純に 0/1 の値で個々の選択弧対の解消方向を決定すると，択グラフに極めて高い確率で閉路が生じ，実行可能スケジュールすら得ることができない．そこで各記号の値を選択弧対の優先解消方向と捉え，個体とスケジュールとの対応，すなわち遺伝子形式 (genotype) から表現形式 (phenotype) への変換規則を以下のように定める．

0° すべての選択弧対を未解消にしておく．

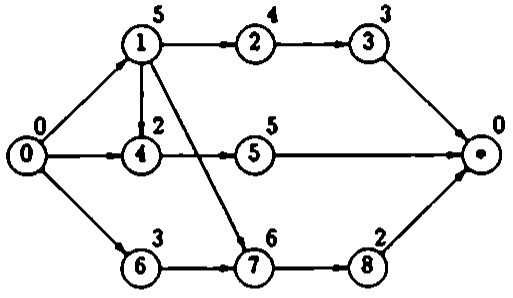
1° 現時点までの選択弧対の解消状況を示すグラフ (図 6.1 (b)) に対して，未解消の選択弧対を開放除去したグラフ (図 6.1 (c)) でクリティカル・パス計算を行い[5]，節点 (作業) の最早開始時刻 (スケジュール) を求める．



(a) 選択グラフ



(b) 選択弧対が部分的に解消されたグラフ



(c) 未解消の選択弧対を開放除去したグラフ

図 6.1 クリティカル・パス計算に用いられる種々のグラフ

有向弧および選択弧の重みを，その始点となる節点に付加してある．

2° このスケジュールでコンフリクトが起こっていなければ，節点 • の最早開始時刻が C_{max} を与える．その時，終了．コンフリクトが起こっていれば，最も早い時刻に生じている 2 節点間の選択弧対を，対応する記号の値を参照して解消し，1° へ．

例えば，図 6.1 の例題において個体が

0 0 1 0 1 0 1

のときのクリティカル・パス計算は，表 6.1 のように行われる．このとき，最終的な選択弧対の解消状況は図 6.2 に示すようになる．

さらに，個体が表すスケジュールの C_{max} から，C_{max} の上界値 C^U_{max} および下界値 C^L_{max} を用いて，適応度 $f(C_{max})$ を次式で計算する:

$$f(C_{max}) = \frac{C^U_{max} - C_{max}}{C^U_{max} - C^L_{max}} . \tag{6.1}$$

ただし，C^U_{max} および C^L_{max} は，

C^U_{max} = 全作業の処理時間の和，

C^L_{max} = 選択弧対をすべて開放除去したグラフ上でのクリティカル・パスの長さ

により求める．

6.3.3 個体表現における冗長性

6.3.1 および 6.3.2 で定義された個体の表現方法では，参照されない記号がいくつか含まれている (ただし，ある記号が参照されるか否かは，実際にクリティカル・パス計算をしてみないとわからない)．図 6.1 の例では，4, 5 および 6 番目の記号 (それぞれ d_{25}, d_{26} および d_{56} に対応) が参照されず，

0 0 1 • • • 1

はすべて同一のスケジュールを表すことになる．このような個体表現における冗長性，すなわち個体 (genotype) としては異なるものが同じスケジュール (phenotype) を表すのを許容することによって，いかなる個体をも実行可能スケジュールに対応させることができるようになる．

6.3.4 部分記号列と作業順序

6.3.1 の個体表現では，機械ごとに，そこで処理される作業間の順序 (選択弧対の解消方向) が連続した部分記号列に対応している．たとえば，図 6.1 の例では，

$$\underbrace{0\ 0\ 1}_{M_1}\ \underbrace{0\ 1\ 0}_{M_2}\ \underbrace{1}_{M_3}$$

表 6.1 クリティカル・パス計算の例

解消された選択弧対	最早節点時刻 (1 - •)									コンフリクト
—	0	5	9	0	2	0	3	9	12	(1, 4)
d_{14} (1 → 4)	0	5	9	5	7	0	3	9	12	(1, 7)
d_{17} (1 → 7)	0	5	9	5	7	0	5	11	13	(4, 7)
d_{47} (7 → 4)	0	5	9	11	13	0	5	11	18	(3, 8)
d_{38} (8 → 3)	0	5	13	11	13	0	5	11	18	

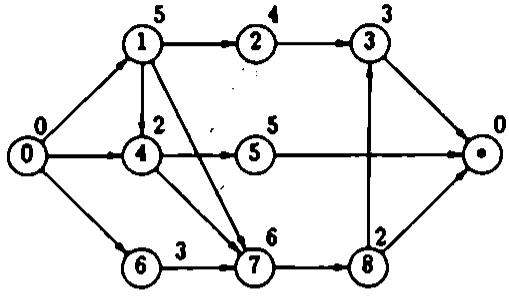


図 6.2 選択弧対の最終解消状況

点線は，コンフリクトを起こさなかった対．すなわち対応する記号が参照されなかった選択弧対を表す．

のように対応している．このとき，各機械上での作業順序を表す部分記号列が交叉によって壊されにくく，6.2.3 で述べた同時並列性がうまく機能すると期待できる．

6.4 遺伝アルゴリズムの近傍モデル

6.4.1 近傍モデルの導入と遺伝演算子の適用法

6.2 での GA では，個体群全体において適応度による選択が行われる (全体モデルと呼ぶ) ので適応度の高い個体が急速に増え，早い時期に個体群の多様性が減少してしまう可能性が高い．そこで各個体についてその近傍を定義し，近傍ごとに選択が局所的かつ並列的に行われるようにする．これを近傍モデルと呼ぶ．具体的には，図 6.3 のように個体を 2 次元の閉じた平面 (トーラス) 上に配置することにして個体群に xy 座標を設け，ある個体からの距離が 1 以内にある個体の集合をその近傍と定義する．

この近傍モデルでは，選択の局所化によって，ある個体の適応度が極端に高い場合でも，その影響は隣り合う近傍間の重なりを通して徐々にしか個体群全体に波及しないので，適応度の高い個体が急速に増加することが抑制される．この結果，個体群の多様性が比較的高く保たれることになる．近傍モデルでの遺伝演算子の構成を以下に示す．

- 選択：
  個体群全体での適応度の最大値，最小値，平均値および近傍内での適応度の平均値を，それぞれ f_g^{M}, f_g^{m}, $\bar{f}_g$ および $\bar{f}_n$ とし，さらに閾値乗数 H を用いて個体 x の子の数を，

  $$2\ (\text{増殖}), \quad f(x) - \bar{f}_n > H(f_g^{\mathrm{M}} - \bar{f}_g) \text{ のとき}, \tag{6.2}$$

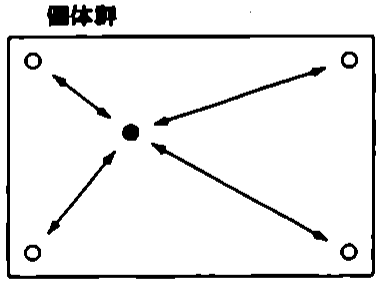
  $$0\ (\text{死滅}), \quad f(x) - \bar{f}_n < H(f_g^{\mathrm{m}} - \bar{f}_g) \text{ のとき}, \tag{6.3}$$

  $$1\ (\text{生存}), \quad \text{それ以外のとき} \tag{6.4}$$

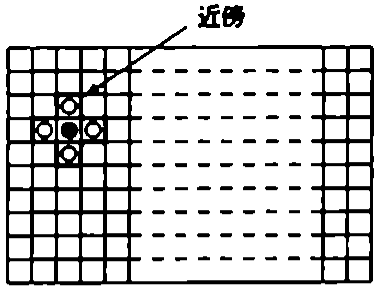
  とする．ただし 1 つの近傍内に増殖および死滅の条件を満足する個体がある場合にのみ，増殖する個体が死滅する個体に取って替わるものとする．一方，増殖 (あるいは死滅) の条件を満たす個体が複数個ある場合には，最も $f(x)$ の高い (低い) ものを選ぶことにする．

  さらに，選択は各近傍に対して独立に行うものとする．ただし 1 つの個体は，それを中心とする 5 個の近傍に含まれるので，選択を行った結果これら 5 つの内容が一致しない場合が生じ得る．このような場合，最も適応度の高いものを選んで次の世代に残すことにする．

- 交叉：
  各近傍において，中心にある個体と，残りの 4 個体の中からランダムに選ばれた個体とをペアリングするものとする．さらに，交叉が行われる近傍の順序を決めておき，



(a) 全体モデル



(b) 近傍モデル

図 6.3 全体モデルと近傍モデル

近傍モデルでは，上辺と下辺および左辺と右辺が，それぞれつながっている．

ある近傍でペアとして選ばれた個体のいずれかが既に他の近傍において選ばれている場合には，その近傍では交叉を行わないものとする．

- 突然変異：
  各近傍の中心の個体に対してのみ突然変異を行う．

なお，Mührenbein[46] や Schleuter[47] らによっても近傍ごとに遺伝演算子を並列的に適用する方法が提案されているが，本論文の方法とは，個体の並べ方 (近傍定義の方法) および遺伝演算子の適用方法が異なる．

6.4.2 初期個体群の生成方法

GA の探索過程において多様性を維持することは重要であるが，同時に適応度の高い個体を増殖させることも大切である．そこで，なるべく適応度の高い個体から成る初期個体群を得るために，以下に示すような初期個体群生成ルーチンを構成する．

0° 適当な値 α (≥ 1) を設定し，$\alpha \times N_p$ 個の個体をランダムに生成して C とする．

1° C から適応度の高い個体を N_p 個選び出し，初期個体群 $\mathcal{P}(1)$ とする．

なお以下では，6.2.2 (4) の 4 つのパラメータに，

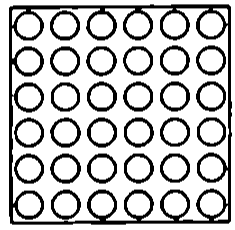
M ：モデル (M^g: 全体モデル, M^n: 近傍モデル)，
$I(\alpha)$ ：初期個体群の生成方法 (I^r: ランダム, I^f: 6.4.2 の適応度を用いる方法)，
H ：閾値乗数

の 3 パラメータを加えた 7 項組 $G = (M, N_g, N_p, p_c, p_m, I(\alpha), H)$ によって，特定の GA の実現を表すことにする．

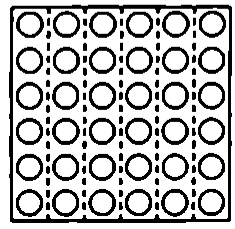
6.5 アルゴリズムの並列化

近傍モデルを用いた GA の大きな利点の 1 つは，アルゴリズムを比較的容易に並列化できること，すなわち並列計算機上にインプリメントできることである．ここでは，MIMD(Multi-Instruction Multi-Data) 型の並列計算機トランスピュータ (transputer) を使用した場合について述べる．

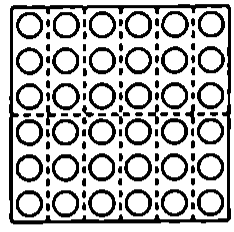
トランスピュータは，ホスト・コンピュータとの通信を行うための 1 個のプロセッサ (以下 root と呼ぶ) および通信リンクによって結ばれたいくつかのプロセッサを持っている．リンクによる通信は双方向であるが，1 プロセッサ当りのリンクは最大 4 本という制約がある．すなわちトランスピュータでは，あるプロセッサと直接通信可能なプロセッサは 4 個までであり，通信オーバヘッドを少なくするためには，広範囲な通信をなるべく抑える工夫が必要である．



(a) 1 プロセッサ



(b) 6 プロセッサ



(c) 12 プロセッサ

図 6.4 個体のプロセッサへの割当て

$N_p = 36$ の場合の例を示す．破線で区切られた部分個体群が同一のプロセッサに割り当てられる．

6.4 で述べた近傍モデルをトランスピュータ上で実現するに当たって，図 6.4 のように個体群全体をいくつかのブロック (部分個体群) に分割し，それぞれのブロックに含まれる個体を同一のプロセッサへ割り当てることにする．ただし，式 (6.2) ～ (6.4) におけるように個体群全体の適応度を参照して選択を行うことは通信量の増大を伴うので，近傍内の適応度のみを参照するように選択演算子を変更する．

- 選択：
  世代 t において，近傍内での適応度の最大値 $f_n^M(t)$ および最小値 $f_n^m(t)$ が，

$$f_n^M(t) - f_n^m(t) > H(F^M(t) - F^m(t)) \tag{6.5}$$

  を満たすとき，$f_n^M(t)$ および $f_n^m(t)$ の子の数をそれぞれ 2 (増殖)，0 (死滅) とする．ただし $F^M(t)$ は，

$$F^M(t) = \begin{cases} \text{初期個体群での適応度の最大値} & (t = 1) \\ \dfrac{F^M(t-1) + f_n^M(t-1)}{2} & (t > 1) \end{cases} \tag{6.6}$$

  によって求められる個体群全体での適応度の最大値の推定値，同様に $F^m(t)$ は，

$$F^m(t) = \begin{cases} \text{初期個体群での適応度の最小値} & (t = 1) \\ \dfrac{F^m(t-1) + f_n^m(t-1)}{2} & (t > 1) \end{cases} \tag{6.7}$$

  による個体群全体での適応度の最小値の推定値を表し，H は閾値乗数である．なお $f_n^M(t)$ あるいは $f_n(t)$ が複数個ある場合には，適当に 1 つを選ぶことにする．また，増殖あるいは死滅しない個体の子の数は 1 (生存) とする．
  さらに，各近傍の中心にある個体が，その近傍内で最小の適応度を持つ場合に限って選択を行い，中心の個体のみを次世代に残すものとする．

さらに，通信量を少なくするために交叉演算子を以下のように変更する．

- 交叉：
  各個体について，交叉の相手を近傍内から 1 つ選び出してペアリングし，確率 p_c で分解・再構成する．ただし近傍内から選ばれた個体については，再構成されたものとの置換えはしない．

このように，トランスピュータの特徴を考慮して並列計算用に変更された近傍モデルを，以下では完全近傍モデル (MP) と呼ぶ[1]．

[1] 完全近傍モデルでは，全体モデルおよび近傍モデルにおけるものと p_c の意味が異なる．すなわち，これら 2 つのモデルでは，適当にペアリングされた $\lfloor N_p/2 \rfloor$ 組のそれぞれについて確率 p_c で交叉するのに対して，完全近傍モデルでは，N_p 個の個体それぞれに対して確率 p_c で交叉する．

6.6 計算例および考察

6.6.1 全体モデルと他の解法との比較

仕事間に先行関係のない 8 仕事の問題で,

仕事に含まれる作業数 : 2 ～ 4,
作業が処理される機械 : M_1 ～ M_6,
作業の処理時間 : 1 ～ 12

を一様乱数によって決定した 100 個の例題に対し, 厳密解法である分枝限定法 (BAB) および GA の全体モデルを適用した. ここで分枝限定法は, 3.2.3 で述べたコンフリクト解消法に基づく分枝操作, 下界値による限定操作, および深さ優先探索を用い, GA では, パラメータを $G = (M^6, 50, 20, 0.6, 0.005, I', -)$ と設定し, 計算時間には Sony News 1850 を使用した.

図 6.5 に, GA によるスケジュールの良さ R :

$$R = \frac{\text{GA を適用した場合の } C_{max}}{\text{BAB を適用した場合の } C_{max} \text{ (最適値)}}. \tag{6.8}$$

および BAB と GA との計算時間の比較を示す. ただし 100 個の例題のうち, 分枝限定法 (生成頂点数 200,000 で計算打ち切り) で最適解 (最適スケジュール) が求まらなかった 19 個の例題に対する結果は除外してある. 図より, 次のことがわかる:

(a) 計算に使用した問題の見かけ上の規模はほとんど同じであるが, BAB では計算時間のばらつきが大きい. これに対して GA では, 計算時間がほぼ一定であり, ほとんどの問題例で最適解が得られている.

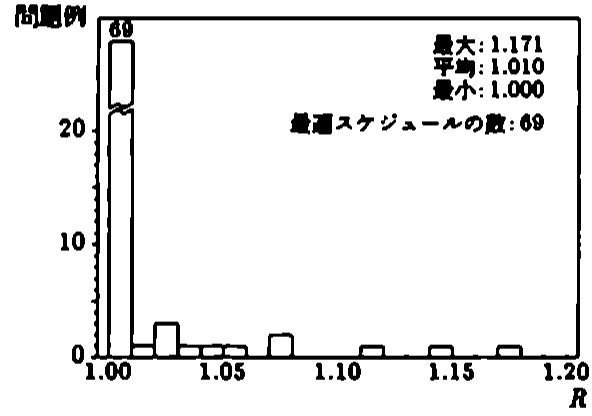
次に, BAB に要する計算時間が最も長いにもかかわらず GA で良い解が得られた問題例 P_1 と, その逆に BAB では簡単に解けるが GA では良い解が得られなかった問題例 P_2 に対する最適スケジュール (の 1 つ) を図 6.6 に示す. これより,

(b) P_1 はクリティカルな作業が 1 つの機械に集中し (図 6.6 (a)), 相異なる最適解が多い問題例である. それ故, GA のように探索空間を絞り込む方法でも容易に最適解が得られるが, 分枝限定法では終端効率が低下し計算に時間がかかる,

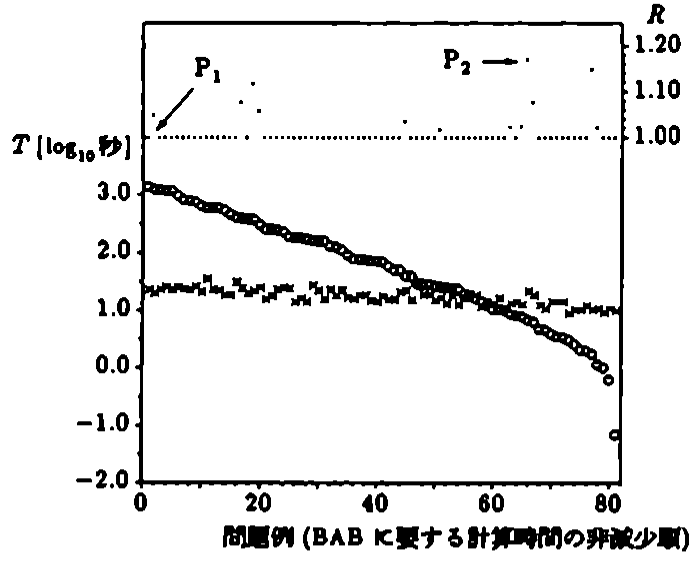
(c) これとは対照的に P_2 では, クリティカルな作業がほとんどすべての機械に分散しているため (図 6.6 (b), (c)) 相異なる最適解が少なく, 探索空間を絞り込みすぎると最適解が得にくくなる

と見ることができる.

最後に, これら 81 個の問題例に対して, ヒューリスティクスの 1 つであるリスト・スケジューリング (List Scheduling: LS) 法を適用した結果を表 6.2 および図 6.7 に示す. なお LS 法では, 3 種類の優先規則:



(a) 解の良さ R のヒストグラム

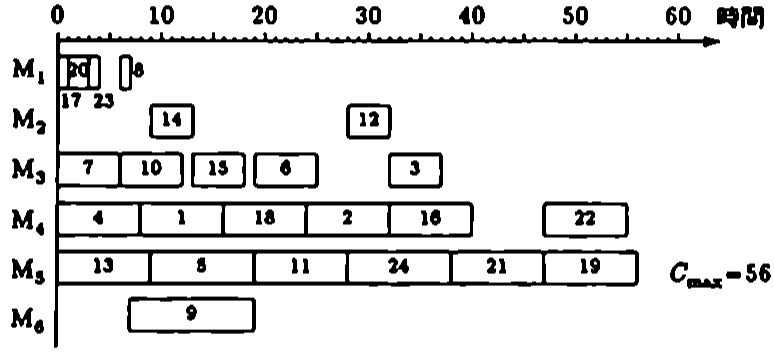


(b) 問題例ごとの計算時間 T と解の良さ R

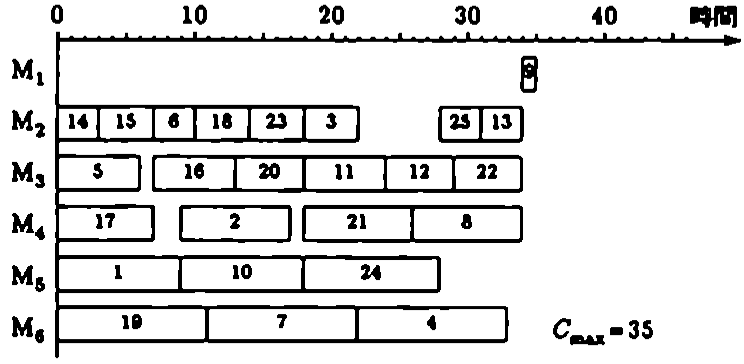
記号 ○, × および • はそれぞれ, BAB に要する計算時間, GA に要する計算時間 T, および GA によって得られるスケジュールの良さ R を示す.

図 6.5 分枝限定法との比較

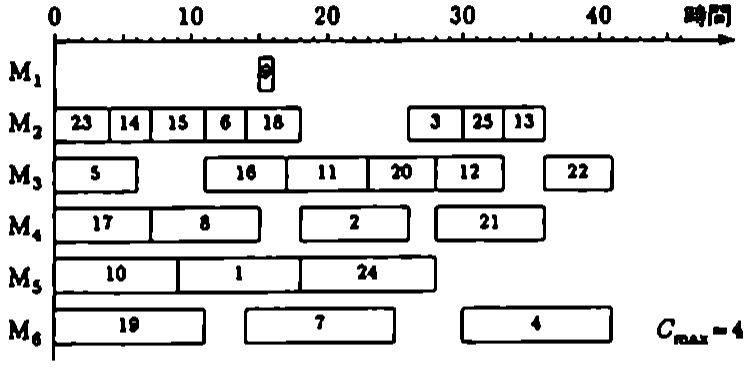
100 個の例題のうち, BAB (生成頂点数 200,000 で計算打ち切り) で最適解が求まらなかった 19 個の例題は除外してある.



(a) 最適スケジュール (問題例 P_1)



(b) 最適スケジュール (問題例 P_2)



(c) GA によるスケジュール (問題例 P_2)

図 6.6 最適スケジュール

表 6.2 リスト・スケジューリング法との比較

方法	パラメータ	R	T
		平均値 (最良値 最悪値)	平均値 [秒]
GA	(M^6, 50, 20, 0.6, 0.005, I', −)	1.010 (1.000 1.171)	18
	(M^6, 50, 40, 0.6, 0.005, I', −)	1.002 (1.000 1.086)	35
	(M^6, 100, 40, 0.6, 0.005, I', −)	1.003 (1.000-1.070)	73
LS	SPT 規則	1.135 (1.000-1.529)	0.19
	LPT 規則	1.031 (1.000 1.179)	0.17
	MST 規則	1.019 (1.000-1.178)	0.15

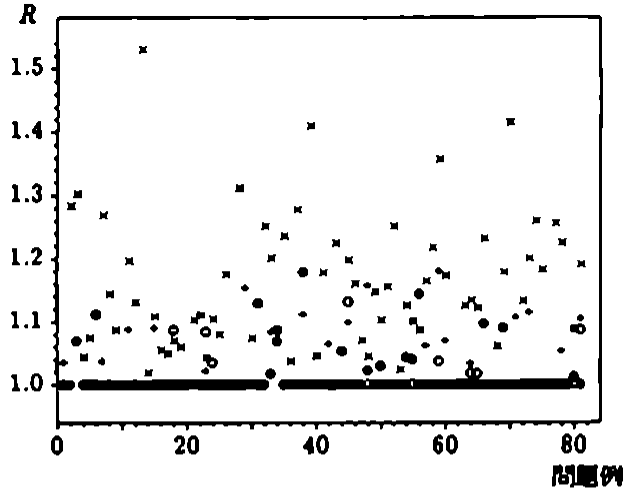


図 6.7 リスト・スケジューリング法との比較

- • : G = (M^6, 100, 40, 0.6, 0.005, I', −),
- × : SPT 規則,
- \+ : LPT 規則,
- ∘ : MST 規則.

(1) 処理時間最小作業優先規則 (SPT 規則),

(2) 処理時間最大作業優先規則 (SPT 規則),

(3) 最大後続時間最大作業優先規則 (SPT 規則),

を使用した. これらより, 以下のことがわかる:

(d) GA と LS で計算に要する時間は, 比較にならないぐらいの差がある. しかし, LS では適する優先規則が問題例ごとに異なるのに対して, GA により得られる解は安定して良い.

6.6.2 近傍モデルによる多様性の維持効果

6.6.1 での問題例 P_2 (C_{max} の最適値 = 35) に対して, GA の全体モデル M^a および近傍モデル M^n を適用した結果を表 6.3 に示す. なお, 計算には Toshiba AS-4060 を使用し, パラメータの各設定値について初期個体群を変えて 100 回計算を行った. 同表の結果に考察を加えれば, 以下のことがわかる:

(a) 近傍モデルで閾値乗数 H を適切に選べば, 同程度の計算時間で, 全体モデルに比べて良い解が得られる割合を高め得るとともに, 悪い解の割合を減らすことができる.

(b) 初期個体群の生成に関しては, 全体モデルおよび近傍モデルとも適応度による方法が有効である. 特に近傍モデルでは, この初期化方法によって解が良くなるだけでなく, 適切な閾値乗数の範囲が広くなる.

(c) 適応度を考慮した場合, 初期個体群の多様性は他の方法より低くなる. しかし個体群内での適応度の差が小さいため, 各個体が選択によって消滅しにくく, 多様性が長期間維持され得るという見方もできる.

さらに, GA の探索過程での平均適応度 $\bar{f}$ の変化を図 6.8 に示す. これより,

(d) 近傍モデル (図中 □) では, 適応度の低い個体も残されていて, 個体群の平均適応度 $\bar{f}$ が比較的低い値になっていること,

が確認できる. このことは, 個体群の多様性が維持されていることを意味するものである.

さらに, 図 6.9 (a) は G = (M^a, 400, 36, 0.6, 0.005, I', −), また同図 (b) は G = (M^n, 400, 36, 0.6, 0.005, I', 0.8) (図 6.8 の + および □ にそれぞれ対応) による計算途中の個体群の模様を, 各個体の適応度を円の半径に比例させて示したものであるが, この図からも, 近傍モデルを用いることによって個体群の多様性が適切に維持されていることがわかる. これらの結果より, GA では計算の初期だけでなく, 長期間にわたって個体群の多様化を図ることが重要であると言える.

近傍モデルによる個体群多様性の維持効果を示すこれらの結果は, ランダムに作成したいくつかの問題例に対しても確認されたが, 今後, より一般的に, 多様化の効果と重要性を示

表 6.3 全体モデルと近傍モデルの比較

パラメータ	C_{max}	T
	平均値 (最良値-最悪値), 最良値の数, 最悪値の数	平均値 [秒]
$G_g(100, I^f(10))$	36.92 (35-40), 26, 1	21
$G_g(400, I^f(10))$	35.45 (35-38), 80, 6	85
$G_n(I^r, 0.5)$	35.51 (35-38), 79, 10	71
$G_n(I^r, 0.6)$	35.38 (35-38), 83, 5	71
$G_n(I^r, 0.7)$	35.27 (35-38), 87, 3	71
$G_n(I^r, 0.8)$	35.26 (35-38), 87, 4	69
$G_n(I^r, 0.9)$	35.53 (35-38), 70, 3	65
$G_n(I^f(10), 0.5)$	35.18 (35-38), 92, 4	74
$G_n(I^f(10), 0.6)$	35.25 (35-38), 89, 4	74
$G_n(I^f(10), 0.7)$	35.19 (35-38), 91, 3	74
$G_n(I^f(10), 0.8)$	35.15 (35-38), 93, 3	72
$G_n(I^f(10), 0.9)$	35.36 (35-37), 75, 11	67

$G_g(N_g, I(\alpha)) = (M^g, N_g, 36, 0.6, 0.005, I(\alpha), -)$.
$G_n(I(\alpha), H) = (M^n, 400, 36, 0.6, 0.005, I(\alpha), H)$.

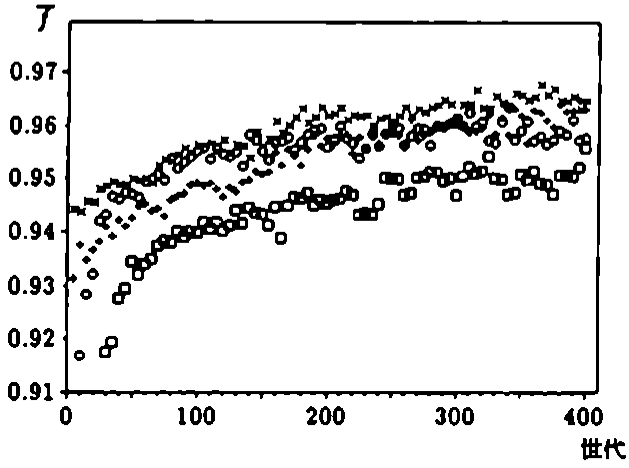
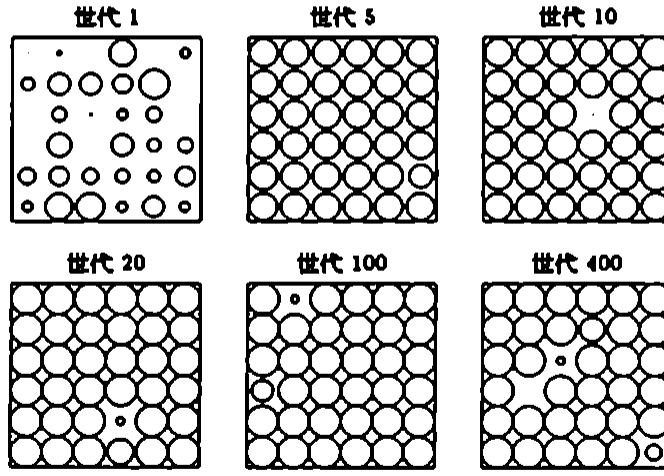
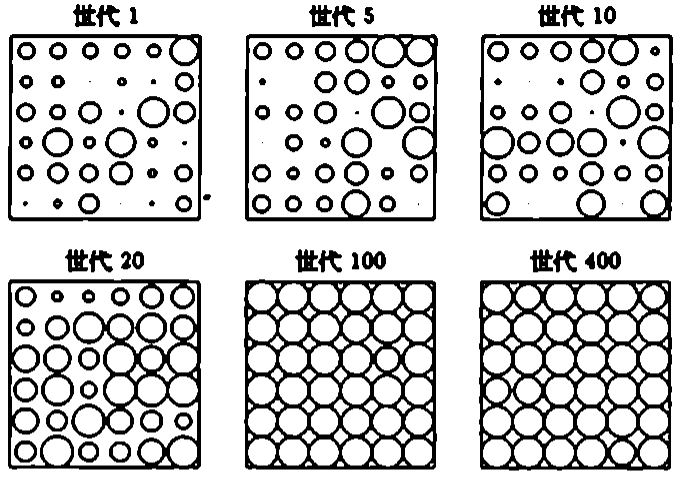


図 6.8 個体群の平均適応度の変遷

+ : G = (M^{s}, 400, 36, 0.6, 0.005, I^{r}, −).
× : G = (M^{s}, 400, 36, 0.6, 0.005, I^{f}(10), −).
∘ : G = (M^{n}, 400, 36, 0.6, 0.005, I^{r}, 0.5).
□ : G = (M^{n}, 400, 36, 0.6, 0.005, I^{r}, 0.8).



(a) 全体モデル $G = (\mathrm{M}^{\mathrm{s}}, 400, 36, 0.6, 0.005, \mathrm{I}^{\mathrm{r}}, -)$



(b) 近傍モデル $G = (\mathrm{M}^{\mathrm{n}}, 400, 36, 0.6, 0.005, \mathrm{I}^{\mathrm{r}}, 0.8)$

図 6.9 個体の適応度の変遷

各個体を，その半径を適応度に比例させた円で表す．

すとともに，近傍モデルおよび閾値による選択の有効性を評価する必要がある．

6.6.3 並列化による計算時間の短縮効果

6.6.2 で用いた 6 機械，8 仕事，25 作業の問題例 P_2 に対して，まず Toshiba AS-4060 を用いた直列計算により，近傍モデルおよび完全近傍モデルによって得られる解の良さを比較した．その結果を表 6.4 に示す．さらに同じ例題に対して，ホスト・コンピュータ Toshiba AS-3260 上に実装されたトランスピュータ T800 を使用した場合の計算時間を表 6.5 に示す．並列計算では，図 6.10 のように接続された 1, 6 あるいは 12 個のプロセッサを使用して個体を図 6.4 のように各プロセッサに割り当て，それぞれの場合について初期個体群を変えて 10 回ずつ計算を行った．表 6.4 および表 6.5 より，以下のことがわかる:

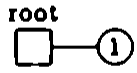
(a) 完全近傍モデルでは，並列化により計算時間の短縮を図ることが可能であるが，適応度に関する情報を近傍からのものに限定するために，近傍モデルに比べて解の良さは低下する．

ただし，トランスピュータには共有メモリがなく，遺伝演算子を適用する際に必ずプロセッサ間通信による情報交換が必要となるので，計算時間は理想値 (1/プロセッサ数) までは減少しない．

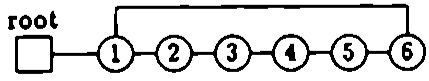
これらの結果より，利用できる情報を近傍内のみに限定した完全近傍モデルでは，並列計算によって計算時間は短縮されるが，得られる解の良さは全体の情報を利用できる近傍モデルよりも若干低下することがわかる．すなわち，トランスピュータのように広範囲な通信が困難な並列計算機上で近傍モデルを実現する場合には，近傍からの局所的な情報をもとに，いかにして全体として効率的な動作 (探索) をさせるかということが重要である．

6.7 むすび

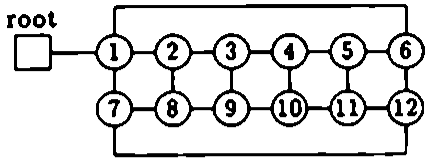
本章では，まず，生物の進化過程を模倣した遺伝アルゴリズム (GA) のジョブショップ問題に対する一構成手法を提案し，計算例を通してこれまでの代表的な解法と比較することにより，最適化の近似解法としての GA の有効性を示した．次に，近傍モデルを導入することによって個体群の多様性を維持する方法を提案するとともに，このアルゴリズムを MIMD 型の並列計算機 (トランスピュータ) 上で実現する方法を示した．ジョブショップ問題に対する計算例を通して，近傍モデルおよび適応度に基づく初期化により安定して良い解が得られること，およびアルゴリズムの並列化によって計算時間が格段に短縮できることを確認した．今後，完全近傍モデルにおける遺伝演算子の構成方法，GA の階層化によるパラメータ調整の自律化，および個体にスケジュール生成規則を対応させたクラシファイア・システム (classifier system) [16] によるオンライン・スケジューリングの実現方法などについての検討



(a) 1 プロセッサ



(b) 6 プロセッサ



(c) 12 プロセッサ

図 6.10 プロセッサの接続

円がプロセッサを，実線がプロセッサ間の通信リンクを表す．

表 6.4　近傍モデルと完全近傍モデルの比較

パラメータ	試行回数	C_{max}
		平均値 (最良値-最悪値), 最良値の割合 [%]
$G_n(0.8)$	100	35.15 (35-38), 93
$G_n(0.9)$	100	35.36 (35-37), 75
$G_p(0.8)$	50	35.42 (35-38), 80
$G_p(0.9)$	50	35.42 (35-38), 82

$G_n(H) = (M^n, 400, 36, 0.6, 0.005, I^r(10), H)$,
$G_p(H) = (M^p, 400, 36, 0.3, 0.005, I^r(10), H)$.

表 6.5　完全近傍モデルによる結果

プロセッサ数	計算時間		高速化率
	[秒]	1 プロセッサの場合との比	
1	593	1.000 (1.000)	1
6	128	0.216 (0.167)	4.63
12	90	0.152 (0.083)	6.58

括弧内の値は 1/プロセッサ数.

が望まれる．さらに，近傍モデルによる多様性維持の有効性に対する解析的な裏付けも必要であると考えられる．

第 7 章

結 論

本研究では，種々の状況に応じて広範なモデル化が行われているスケジューリング問題のうちでジョブショップ型の問題を取り上げ，モデルの拡張という観点からバッファを考慮したスケジューリング問題の定義，モデル化手法および解法の開発を行うとともに，ジョブショップ問題に対する実用的な解法として，分解による近似解法および部分的探索に基づく近似解法を提案した．以下，本論文の各章で得られた結論を整理する．

第 2 章では，従来のスケジューリング理論で対象とされる問題を体系的に整理し，本研究で対象とするジョブショップ・スケジューリング問題の位置付けを明確にした．さらに，特定の問題に適用可能な最適解法 (アルゴリズム)，および一般のスケジューリング問題に対して汎用的に適用できる組合せ最適化手法について述べた．

第 3 章では，本研究で取り上げるジョブショップ・スケジューリング問題を定義するとともに，この問題に対する代表的な解法 (最適解法および近似解法) を紹介した．

第 4 章では，バッファを考慮したスケジューリング問題を定義し，混合整数計画問題への定式化を通して，選択グラフ，タイム・ペトリネット (TPN) およびガントチャートに基づく 3 種のモデル化手法を提案した．選択グラフ・モデルにより，バッファを考慮した問題に対する従来のスケジューリング理論における既存解法の適用を可能とした．TPN モデルでは，一括スケジューリングおよび繰返しスケジューリングのための分枝限定法を構成した．ガントチャート・モデルは，スケジュール生成過程がわかり易いという特徴を有するものであり，このモデルを用いて，分枝限定法およびリスト・スケジューリング法を構成した．

計算例を通して 3 種のモデル上での分枝限定法による結果を比較したところ，最適 (あるいは最適に極めて近い) スケジュールを求めるためには，TPN モデル上での分枝限定法が有効であることを確認した．さらに，バッファの導入がスケジュールに与える効果については，TPN モデル上での分枝限定法を用いて検討した結果，バッファを設けることによって，一括スケジューリングでは最大完了時間が，また繰返しスケジューリングでは繰返し周期が短縮されることを確認した．最後に，ガントチャート・モデル上でのリスト・スケジューリング法による結果より，すべての作業を平均的に処理していくような優先規則がバッファを考慮したスケジューリング問題に対して比較的有効であることが明らかにされた．

第 5 章では，ジョブショップ・スケジューリング問題に対する分解による近似解法を考案

した．まず，部分問題の求解困難さを事前に見積もるため，問題の難易度を数値化する手法を提案し，次にこれに基づき，3種の分解アルゴリズムを提案した．計算例より，難易度と求解に要する時間との間の相関，および難易度を著しく減少させるような分解を行うと解の良さが低下することが確認された．さらに，3種の分解アルゴリズムについて，一括，逐次，反復型とアルゴリズムを複雑化していくにつれて，計算時間の増加は伴うが，より良いスケジュールが得られることがわかった．

第6章では，部分的探索法として，生物の進化過程を模倣した遺伝アルゴリズム(GA)を取り上げ，ジョブショップ問題に対するGAの一構成手法を提案した．計算例を通して，これまでの代表的な解法である分枝限定法およびリスト・スケジューリング法と，計算に要する時間および得られる解の良さの点で比較することにより，近似解法としてのGAの有効性を示した．次に，GAの近傍モデルを導入することによって個体群の多様性を維持する方法を提案するとともに，このアルゴリズムを並列計算機トランスピュータ上で実現する方法を示した．計算例を通して，近傍モデルによって良い解が安定して得られること，およびアルゴリズムの並列化により計算時間が格段に短縮できることを確認した．

第1章で述べた理想問題と現実問題との間のギャップを埋めるべく，本研究では，バッファを考慮することによるモデル面からの接近，および近似解法を構成することによる解法面からの接近を試みたが，そのギャップはまだまだ大きいものと考えられる．モデル化の観点からは，種々の問題を統一的/汎用的に扱えるような枠組みが必要であるが，これは，現実に発生する問題をどこまで抽象化できるかにかかっている．これに関しては，そのアプローチ手法から検討していくことが今後の課題である．一方，実用的な解法面に関して，本研究で提案した近似解法の有効性は，すべて計算例を通して経験的に確認されたものであり，その理論的な解析は全く行われていない．特にパラメータの設定等に関する指針もほとんどない状況である．これらに関して，理論的/解析的な研究を進めていくことが今後の課題である．

謝辞

本研究を遂行するにあたり，終始懇切なる御指導および御鞭撻を賜った京都大学工学部 西川禅一教授に謹んで感謝を申し上げます．また，懇切なる御指導および御討論を戴きました京都大学工学部 倉光正巳講師，手塚哲央助手，喜多一助手に厚く感謝の意を表します．

京都工芸繊維大学工芸学部 三宮信夫教授には，研究途上において度々有益な御討論および御教示を賜りました．謹んで感謝を申し上げます．

スケジューリング理論に関して御教授戴きました京都大学工学部 茨木俊秀教授，日頃有益な御教示を賜わりました松下電工株式会社 野村淳二博士には，謹んで感謝を申し上げます．

さらに，日頃お世話になりました西川研究室の方々に感謝します．特に，大学院生 山口浩司君，路広平君，ビタック・プルッティサーリコン君，および学部学生 出野徹君，阿武純君，牧淳人君，西崎浩司君には，計算結果について御協力戴きました．心から感謝いたします．

最後に，京都大学工学部 荒木光彦教授には，本論文をまとめるにあたり，多大な御配慮および御鞭撻を賜りましたことを深謝いたします．

参考文献

1) R. L. Graham, E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan, Optimization and Approximation in Deterministic Sequencing and Scheduling: A Survey, *Ann. Discrete Math.*, vol. 5, pp. 287–326, 1979.

2) E. L. Lawler, J. K. Lenstra and A. H. G. Rinnooy Kan, Recent Development in Deterministic Sequencing and Scheduling: A Survey, in *Deterministic and Stochastic Scheduling*, D. Reidel Pub. Comp., pp. 35–73, 1982.

3) K. P. White, Jr., Advances in the Theory and Practice of Production Scheduling, in *Control and Dynamic Systems 37: Advances in Industrial Systems*, Academic Press, pp. 115–157, 1990.

4) R. W. Conway, W. L. Maxwell and L. W. Miller, *Theory of Scheduling*, Addison-Wesley, 1967 (関根智明訳, スケジューリングの理論, 日刊工業新聞社, 1971).

5) 鶴島一郎, スケジューリング理論, 森北出版, 1974.

6) S. French, *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*, John Wiley & Sons, 1982.

7) 伊理正夫・野崎昭弘・野下浩平, 計算の効率化とその限界, 数学セミナー増刊－入門現代の数学 13, 日本評論社, 1980.

8) 西川禎一・三宮信夫・茨木俊秀, 最適化, 岩波書店, 1982.

9) 茨木俊秀, 組合せ最適化－分枝限定法を中心として－, 産業図書, 1983.

10) N. Metropolis, et al., Equation of State Calculation by Fast Computing Machines, *J. Chemical Physics*, vol. 6, no. 21, pp. 1087–1092, 1953.

11) S. Kirkpatrick, C. D. Gelatt and Jr. M. P. Vecchi, Optimization by Simulated Annealing, *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

12) P. J. M. Laarhoven and E. H. L. Aarts, *Simulated Annealing: Theory and Applications*, D. Reidel Pub. Comp., 1987.

13) E. Aarts and J. Korst, *Simulated Annealing and Boltzmann Machines*, Jon Wiley & Sons, 1989.

14) S. Geman and D. Geman, Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images, *IEEE Trans. Pattern Analysis and Machine Intelli-*

gence, PAMI-6, pp. 721-741, 1984.

15) J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.

16) D. E. Goldberg, *Genetic Algorithm in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

17) 今野 浩・鈴木久敏, 整数計画法と組合せ最適化, 日科技連, 1982.

18) B. J. Lageweg, J. K. Lenstra and A. H. G. Rinnooy Kan, Job-Shop Scheduling by Implicit Enumeration, *Management Science*, vol. 24, no. 4, pp. 441-450, 1977.

19) J. R. Barker and G. B. McMahon, Scheduling the General Job-Shop, *Management Science*, vol. 31, no. 5, pp. 594-598, 1985.

20) J. H. Blackstone, D. T. Phillips and G. L. Hogg, A State-of-the-art Survey of Dispatching Rules for Manufacturing Job Shop Operations, *Int. J. Prod. Res.*, vol. 20, no. 1, pp. 27-45, 1982.

21) 石井博昭, スケジューリング問題の近似解法, オペレーションズ・リサーチ, vol. 31, no. 10, pp. 1360-1366, 1983.

22) 水野幸男, 在庫管理入門, 日科技連, 1974.

23) 門田安弘, トヨタシステム, 講談社, 1985.

24) 黒田 充・田部 勉・園川隆夫・中根義一郎, 生産管理, 朝倉書店, 1980.

25) 西川禅一・玉置 久, 在庫容量を考慮したスケジューリング, 第30回自動制御連合講演会, pp. 457-460, 1987.

26) 人見勝人, 生産システム工学, 共立出版, 1974.

27) 畠 広平・玉置 久・西川禅一, バッファを考慮したスケジューリング問題と分枝限定法, 第33回システム制御情報学会研究発表講演会, pp. 129-130, 1989.

28) 西川禅一・玉置 久, 仕掛り在庫を考慮したスケジューリング, 第31回自動制御連合講演会, pp. 527-530, 1988.

29) J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice Hall, 1981 (市川惇信・小林重信訳, ペトリネット入門, 共立出版, 1984).

30) C. V. Ramamoothy and G. S. Ho Performance Evaluation of Asynchronous Concurrent Systems Using Petri Nets, *IEEE Trans. Software Engineering*, vol. SE-6, no. 5, pp. 440-449, 1980.

31) 新井英哲・藤森 敦・久村富持, 繰返しスケジューリング問題へのタイム・ペトリネットの応用と緊急停止時の復旧方策, 計測自動制御学会論文集, vol. 22, no. 9, pp. 955-961, 1986.

32) P. ピタック・玉置 久・西川禅一, バッファを考慮したスケジューリング問題に対するリスト・スケジューリング法, 第33回システム制御情報学会研究発表講演会, pp. 129-130, 1989.

33) S. A. Kantsedal, A Decomposition Approach to the Solution of Large-Scale Scheduling Problems, *Automation and Remote Control*, vol. 44, no. 10, pp. 1360-1366, 1983.

34) J. Adams, E. Balas and D. Zawack, The Shifting Bottleneck Procedure for Job Shop Scheduling, *Management Science*, vol. 34, no. 3, pp. 391-401, 1988.

35) 西川禅一・玉置 久・牧 淳人, ジョブショップ・スケジューリング問題に対する分解法, 計測自動制御学会論文集, vol. 27, no. 5, pp. 607-613, 1991.

36) システム/制御/情報, vol. 35, no. 1 (ニューラルネットワークの応用特集号), システム制御情報学会, 1991.

37) 計測と制御, vol. 30, no. 4 (特集 ニューラルネット), 計測自動制御学会, 1991.

38) 第10回システム工学部会研究会資料 - Genetic Algorithm をめぐって -, 計測自動制御学会, 1992.

39) 西川禅一・玉置 久, ジョブショップ・スケジューリング問題に対する遺伝アルゴリズムの一構成法, 計測自動制御学会論文集, vol. 27, no. 5, pp. 593-599, 1991.

40) L. Davis, Job Shop Scheduling with Genetic Algorithms, *Proc. 1st Int. Conf. Genetic Algorithms*. Lawrence Erlbaum Associates, pp. 136-140, 1988.

41) R. Nakano, Conventional Genetic Algorithm for Job Shop Problems, *Proc. 4th Int. Conf. Genetic Algorithms*, Morgan Kaufman, pp. 474-479, 1991.

42) T. Yamada and R. Nakano, A Genetic Algorithm Applicable to Large-Scale Job-Shop Problems, *Proc. 2nd Conf. Parallel Problem Solving from Nature*, North-Holland, pp. 281-290, 1992.

43) 西川禅一・玉置 久・P. ピタック, 遺伝アルゴリズムにおける個体群多様化の効果, 第2回自律分散システムシンポジウム, pp. 45-50, 1991.

44) 西川禅一・玉置 久・P. ピタック・西崎浩司, 遺伝アルゴリズムの並列化とそのジョブショップ型スケジューリング問題への応用, 第35回システム制御情報学会研究発表講演会, pp. 403-404, 1991.

45) H. Tamaki and Y. Nishikawa, A Paralleled Genetic Algorithm based on a Neighborhood Model and Its Application to the Jobshop Scheduling, *Proc. 2nd Conf. Parallel Problem Solving from Nature*, North-Holland, pp. 573-582, 1992.

46) H. Mührenbein, Parallel Genetic Algorithms: Population Genetics and Combinatorial Optimization, *Proc. 3rd Int. Conf. Genetic Algorithms*, Morgan Kaufman, pp. 416-421, 1989.

47) M. Gorges-Schleuter, ASPARAGOS An Asynchronous Parallel Genetic Optimization Strategy, *Proc. 3rd Int. Conf. Genetic Algorithms*, Morgan Kaufman, pp. 422-427, 1989.

本研究に関する発表

論文発表

(1) 西川眹一・玉置 久, ジョブショップ・スケジューリング問題に対する遺伝アルゴリズムの一構成法, 計測自動制御学会論文集, vol. 27, no. 5, pp. 593-599, 1991.

(2) 西川眹一・玉置 久・牧 淳人, ジョブショップ・スケジューリング問題に対する分解法, 計測自動制御学会論文集, vol. 27, no. 5, pp. 607-613, 1991.

(3) H. Tamaki and Y. Nishikawa, Maintenance of Diversity in a Genetic Algorithm and an Application to Jobshop Scheduling, *Proceedings of the IMACS/SICE International Symposium on Robotics, Mechatronics and Manufacturing Systems* 1992, pp. 869-874, 1992.

(4) H. Tamaki and Y. Nishikawa, A Paralleled Genetic Algorithm based on a Neighborhood Model and Its Application to the Jobshop Scheduling, *Proceedings of the 2nd Conference on Parallel Problem Solving from Nature*, North-Holland, pp. 573-582, 1992.

(5) 西川眹一・玉置 久, 近傍モデルによる遺伝アルゴリズムの並列化手法とそのジョブショップ・スケジューリング問題への応用, 計測自動制御学会論文集, vol. 29, no. 5, 掲載予定, 1993.

(6) Y. Nishikawa, H. Tamaki, H. Kita and N. Shimizu, On a Self-Organizing Property of Genetic Algorithms, *Proceedings of the International Symposium on Autonomous Decentralized Systems*, to be published, 1993.

(7) 西川眹一・玉置 久, バッファ容量を考慮したジョブショップ・スケジューリング— 選択グラフ・モデル, 計測自動制御学会論文集投稿準備中.

(8) 西川眹一・玉置 久, バッファ容量を考慮したジョブショップ・スケジューリング タイム・ペトリネット・モデル, 計測自動制御学会論文集投稿準備中.

(9) 西川眹一・玉置 久・P. ビタック, バッファ容量を考慮したジョブショップ・スケジューリング— ガントチャート・モデル, 計測自動制御学会論文集投稿準備中.

口頭発表

(1) 西川禅一・玉置 久, ジョブショップ型スケジューリングにおける在庫の影響に関する考察, 第31回システムと制御研究発表講演会, pp. 227-228, 1987.

(2) 西川禅一・玉置 久, 在庫容量を考慮したスケジューリング, 第30回自動制御連合講演会, pp. 457-460, 1987.

(3) 西川禅一・玉置 久・山口浩司, 仕掛り在庫を考慮したジョブショップ型スケジューリングの研究, 第32回システムと制御研究発表講演会, pp. 321-322, 1988.

(4) 西川禅一・玉置 久・出野 徹, ジョブショップ型スケジューリング問題に対する分解法について, 第32回システムと制御研究発表講演会, pp. 323-324, 1988.

(5) Y. Nishikawa and H. Tamaki, A Study on a Decomposition Method for Jobshop Scheduling, *The 13th International Symposium on Mathematical Programming*, p. 32, 1988.

(6) 西川禅一・玉置 久, 仕掛り在庫を考慮したスケジューリング, 第31回自動制御連合講演会, pp. 527-530, 1988.

(7) 阿武 純・玉置 久・西川禅一, スケジューリング問題に対する分解法についての考察, 第33回システム制御情報学会研究発表講演会, pp. 125-126, 1989.

(8) P. ビタック・玉置 久・西川禅一, バッファを考慮したスケジューリング問題に対するリスト・スケジューリング法, 第33回システム制御情報学会研究発表講演会, pp. 127-128, 1989.

(9) 路 広平・玉置 久・西川禅一, バッファを考慮したスケジューリング問題と分枝限定法, 第33回システム制御情報学会研究発表講演会, pp. 129-130, 1989.

(10) 西川禅一・玉置 久, Genetic Algorithm のスケジューリング問題への応用, 第1回自律分散システムシンポジウム, pp. 117-122, 1990.

(11) 西川禅一・玉置 久・牧 淳人, ジョブショップ型スケジューリングにおける分解法, 第34回システム制御情報学会研究発表講演会, pp. 385-386, 1990.

(12) 西川禅一・玉置 久・P. ビタック, 遺伝アルゴリズムによるスケジューリング問題の解法, 第33回自動制御連合講演会, pp. 413-414, 1990.

(13) 西川禅一・玉置 久・P. ビタック, 遺伝アルゴリズムにおける個体群多様化の効果, 第2回自律分散システムシンポジウム, pp. 45-50, 1991.

(14) 西川禅一・玉置 久・渡辺 豊, ジョブショップ型スケジューリング問題に対するボトルネック移動法の並列化, 第35回システム制御情報学会研究発表講演会, pp. 393-394, 1991.

(15) 西川禅一・玉置 久・P. ビタック・西崎浩司, 遺伝アルゴリズムの並列化とそのジョブ

ショップ型スケジューリング問題への応用, 第35回システム制御情報学会研究発表講演会, pp. 403-404, 1991.

(16) 喜多 一・玉置 久・西川禎一, ニューラルネットと Genetic Algorithm, 第30回計測自動制御学会学術講演会, pp. 859-860, 1991.

(17) 玉置 久・西川禎一, 遺伝アルゴリズムのスケジューリング問題への応用とその近傍を考慮した並列化, 第8回国際研シンポジウム, pp. 220-231, 1991.

(18) 西川禎一・玉置 久, 遺伝アルゴリズムの近傍モデルとそのスケジューリング問題への応用, 第34回自動制御連合講演会, pp. 345-346, 1991.

(19) 玉置 久・喜多 一・西川禎一, 遺伝アルゴリズムのダイナミクスに関する基礎的考察, 第3回自律分散システムシンポジウム, pp. 97-102, 1992.

(20) 玉置 久・喜多 一・西川禎一, 遺伝アルゴリズムにおける遺伝子型のロバスト性に関する考察, 第10回計測自動制御学会システム工学部会研究会, pp. 49-56, 1992.

(21) 玉置 久・喜多 一・西川禎一, 遺伝アルゴリズムのダイナミクスについて, 第31回計測自動制御学会学術講演会, pp. 783-784, 1992.

(22) 玉置 久・西川禎一, 近傍モデルに基づく並列化遺伝アルゴリズムの淘汰規則に関する考察, 第31回計測自動制御学会学術講演会, pp. 285-286, 1992.

(23) 玉置 久・喜多 一・清水延彦・西川禎一, 遺伝アルゴリズムのダイナミクスと自己組織性, 第2回日本機械学会 FAN シンポジウム, pp. 401-406, 1992.

(24) 玉置 久・喜多 一・清水延彦・西川禎一, 遺伝アルゴリズムにおける符号化についての考察, 第4回自律分散システムシンポジウム, pp. 99-104, 1993.