資源制約付きスケジューリング問題に対する拡張モデルとメタ解法

修士学位論文

資源制約付きスケジューリング問題に対する拡張モデルとメタ解法

平成１０年度

（１９９８）

東京商船大学大学院
商船学研究科
流通情報工学専攻

石川友保

目 次

第 1 章

序論

資源制約付きスケジューリング問題(Resource Constrained Scheduling Problem)とは，スケジューリング問題の1つであり，使用可能量の上限を持つ資源（原材料・人・機械など）を使用し，作業（切断・組立など）の間に定義された先行順序関係を満たしつつ，最適な作業の処理順および資源の作業への割り当てを決めるスケジューリング問題である．目的関数は，すべての作業が完了する時刻の最小化（最大完了時刻最小化）や作業の開始時刻に依存する費用や資源の超過のペナルティ費用などの和の最小化（総費用最小化）をとる．資源制約付きスケジューリング問題は，フローショップ問題やジョブショップ問題をはじめ，多くのスケジューリング問題を定式化できるという高い汎用性を持っている．

近年，実際の現場でスケジュールを計画する際，考慮すべき要素が多様化してきており，人の手によって（計算機を使わないで）効率の良い計画を立てることが困難となってきている．そのため計算機上でスケジュールの計画・管理を行うシステム（手法）が考えられている．

本研究ではそれらのシステム（手法）をスケジューリング・システムと呼ぶこととする．既存のスケジューリング・システムを大別すると，スケジュール管理システムとスケジュール最適化システムの2つに分類できる．

スケジュール管理システムとは，スケジュールを管理することを目的とし，作業の資源への割当状況や，資源の超過などを管理できる手法である．この手法を実現したアプリケーション・ソフトをスケジュール管理ソフトと呼ぶ．

スケジュール最適化システムとは，資源制約付きスケジューリング問題を基礎に最適化を導入した手法であり，通常それぞれの問題に特化される．

現実のスケジュールの計画・管理を行うシステムには，

- 様々な付加条件を容易に追加できる
- 作業の効率を高めることができる
- 現場の作業員が使用できる

といった条件が必須であると思われるが，既存のシステムにおいてそれらをすべて満たすものは少ない．

第 2 章

研究の目的と方法

2.1 研究の目的

スケジュールを計画するシステム（手法）は，理論的な解法とは異なって，作業員の勤務時間の制限や使用できる機械数の制限などの制約条件を容易に追加できるように，実際の作業現場の実状を組み入れられるようにすることが重要である．このような資源制約を加味して，作業を処理する順序を，スケジュール全体の効率が良くなるように決めるためには，最適化手法の導入が必要となる．

既存のスケジューリング・システムには，制約条件の追加が容易であり（つまり汎用性が高い）かつ最適化が導入されているシステムは少ない．

本研究では，実際の現場でスケジュールを計画するシステムとして，(1)拡張モデルを用いるシステムと，(2)市販のスケジュール管理ソフトとメタ解法を組み合わせるシステムを，提案することを目的としている．この2つのシステムは，いずれも，制約条件を容易に追加でき，かつ最適化手法を導入している．

この2つのシステムは，関連する既存研究と比較して，以下のような特徴がある．すなわち，数理計画ソルバーCPLEX（ILOG社）と，スケジュール管理ソフトProject98（Microsoft社）という既存のアプリケーション・ソフトを効率良く利用して，スケジュールの精度を高めようという工夫である．

拡張モデルを用いるシステムでは，制約条件を容易に追加できるように定式化を行い，しかも既存のアプリケーション・ソフトである数理計画ソルバーCPLEXを用いている．これにより，制約条件を容易に追加できるという困難な課題に対して，新規にプログラムを作成する必要もなく，短時間でスケジュールを計画することができる．

メタ解法との組合せは，市販のスケジュール管理ソフトProject98によって計画されたスケジュールをもとに，近似解法の1つであるメタ解法を用いて改善する手法である．このため，スケジュールの計画それ自体には大きな労力を必要とせずに，メタ解法を用いて精度を向上させることができる．

2.2 研究の方法

本研究の方法は以下のとおりである.

- 資源制約付きスケジューリング問題の基本形を示す.
- スケジューリング・システムの定義を述べる.
- スジューリング問題と本研究で用いる最適化の手法に関する従来の研究を示す.
- 本研究で用いるアプリケーション・ソフトの概要を述べる.
- 本研究で提案するスケジューリング・システムの概要とアルゴリズムを示す.
- 提案するシステムを実装（プログラミング）し，ベンチマーク問題に対する計算実験を行い，システムの有用性を調べる.

2.3 論文の構成

本論文の構成は以下のとおりである.

第3章では，資源制約付きスケジューリング問題の定義，および基本形の定式化を示す.

第4章では，スケジューリング・システムの定義を述べる.

第5章では，スケジューリング問題と本研究で用いる最適化の手法に関する従来の研究を示す.

第6章では，本研究で用いるアプリケーション・ソフトの概要を述べる.

第7章では，拡張モデルを用いるシステムの概要とアルゴリズムを示す.

第8章では，市販のスケジュール管理システムとメタ解法を組み合わせるシステムの概要とアルゴリズムを示す.

第9章では，第7章と第8章で提案したシステムを組み合わせるシステムの概要とアルゴリズムを示す.

第10章では，提案したシステムに対する計算実験の結果を示す.

第11章では，結論と今後の課題を述べる.

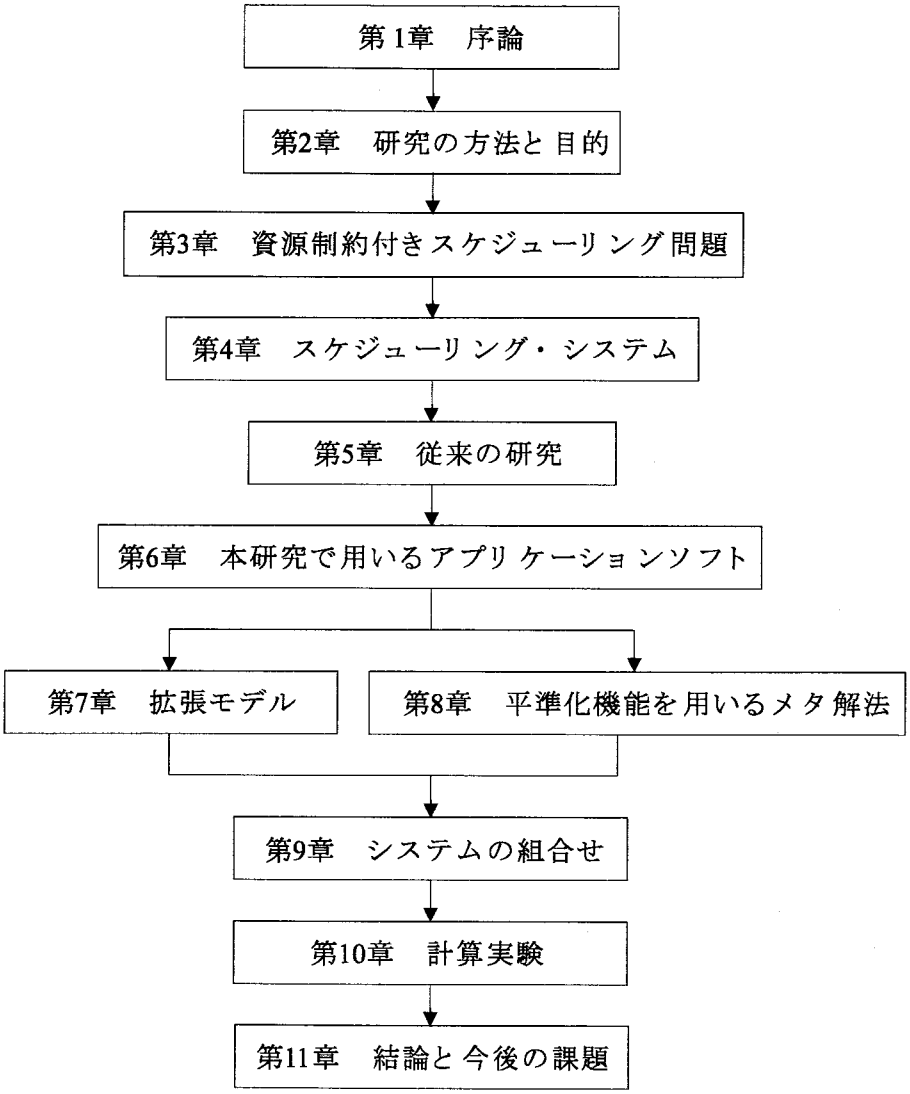
第 1章　序論
第2章　研究の方法と目的
第3章　資源制約付きスケジューリング問題
第4章　スケジューリング・システム
第5章　従来の研究
第6章　本研究で用いるアプリケーションソフト
第7章　拡張モデル
第8章　平準化機能を用いるメタ解法
第9章　システムの組合せ
第10章　計算実験
第11章　結論と今後の課題

図 2.1: 論文の構成

第 3 章

資源制約付きスケジューリング問題

本章では，スケジューリング問題と資源制約付きスケジューリング問題の定義，および資源制約付きスケジューリング問題の基本形の定式化を示す．

3.1 スケジューリング問題の定義

スケジューリング問題とは，予定された作業の最適な処理順を決める問題の総称である [1]．目的関数や制約条件の与え方により，様々な応用が考えられる．

ここでは，スケジューリング問題を検討するとき用いられる，用語，目的，制約条件について代表的なものを挙げる．また，代表的なスケジューリング問題を挙げる．

3.1.1 用語

ここで，スケジューリング問題で使われる代表的な用語を整理する．用語の具体的な例として，製鉄工場内のスケジュールの場合を当てはめて考える（表3.1）．

作業 計画期間中に行わなければならない活動である．例えば，鉄の溶解，鋳型への注入，切断，などである．

資源 作業を処理するために必要な要素である．例えば，鉄鉱石などの原材料や，作業員，切断機械などである．

先行順序 作業間の前後関係を表す．例えば，鉄の溶解を終了しなければ，鋳型への注入は開始できないなどである．

仕事 先行順序関係を持つ作業の集合である．例えば，ある製品をつくる一工程である．

表 3.1: スケジューリング問題で使われる代表的な用語

用語	製鉄工場内スケジューリングの例	配送のスケジューリング例
作業	鉄の溶解，鋳型への注入，切断など	各顧客への配送など
資源	原材料（鉄鉱石，燃料），作業員，機械など	輸送機械（自動車・飛行機），貨物など
先行順序	鉄の溶解→鋳型への注入	顧客iへの配送→顧客jへの配送など
仕事	ある製品をつくる一工程など	同じ配送圏の顧客への配送など

3.1.2 スケジューリング問題の目的

スケジューリング問題では，様々な目的に対する最適化を考慮する．ここではそれらの目的の中で代表的なものを挙げる．

- 最大完了時刻最小化
  すべての作業が完了する時刻を最小化する問題
- 延べ処理時間最小化
  すべての作業の処理時間の合計を最小化する問題
- 総費用最小化
  計画期間中にかかるすべての費用を最小化する問題．費用としては，作業の開始時刻に依存する費用や資源の超過のペナルティ費用，段取り費用などがある．
- 最大納期ずれ最小化
  それぞれの作業の完了時刻に対する納期からのずれ（納期ずれ時間；完了時刻と納期の差の絶対値）のうち最大のものを最小化する問題．
- 最大納期遅れ最小化
  納期に間に合わなかった作業の完了時刻に対する納期からの遅れ（納期遅れ時間；納期より早く作業が完了すれば0）のうち最大のものを最小化する問題．
- 純利益最大化
  作業を処理することにより得られた利益から，計画期間中にかかるすべての費用を引いた純利益を最小化する問題．

3.1.3 スケジューリング問題の制約条件

スケジューリング問題では，様々な制約条件により制約された最適化を考慮する．ここではそれらの制約条件の中で代表的なものを挙げる．

- 作業遂行条件
  すべての作業を計画期間中に処理することを制約する．

- 資源制約
  すべての資源について，資源の使用量はその使用可能量の上限を破らない．

- 先行順序制約
  ある作業の処理が完了するまで，他のある作業の処理が開始できないことを表す．

3.1.4 代表的なスケジューリング問題

スケジューリング問題には目的関数や制約条件の違いから，様々な応用問題がある．ここではそれらスケジューリング問題の応用の中で代表的なものを挙げる．

- ジョブショップ問題
  ジョブショップ問題とは，仕事という概念を導入したスケジューリング問題である．仕事の中の作業はすべて異なった資源が割り当てられている．

- フローショップ問題
  フローショップ問題とは，ジョブショップ問題の特殊形であり，すべての仕事において，仕事の中の作業が資源を用いる順序が同じである．

- 資源制約付きスケジューリング問題
  3.2参照

3.2 資源制約付きスケジューリング問題の定義

資源制約付きスケジューリング問題(Resource Constrained Scheduling Problem)とは，使用可能量の上限を持つ資源を使用するとき，作業間に定義された先行順序関係を満たす，最適な作業の処理順，および資源の作業への割り当てを決めるスケジューリング問題である．

スケジューリング問題には様々な種類があるが，それらのほとんどが資源制約付きスケジューリング問題に一般化できる．

このような適用範囲の広さから，数多くの研究がされている．

本研究では，資源制約付きスケジューリング問題に対する時刻添え字付き定式化を示す．

3.3 時刻添え字付き定式化

時刻添え字付き定式化とは，連続時間を離散化する定式化である（図3.1）．連続時間の$t-1$以上t未満の時間を，時刻tと仮定する．すなわち，計画期間は時刻の集合となる．

この定式化の特徴は，新たな制約条件を加えることが容易であることにある．

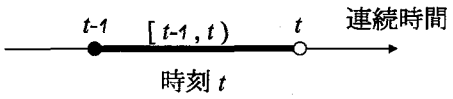

図 3.1: 連続時間の離散化

3.4 資源制約付きスケジューリング問題の定式化

以下に，資源制約付きスケジューリング問題の基本形の構成要素ならびに定式化を示す．

3.4.1 集合

$\mathcal{J}$: 作業の集合

$\mathcal{R}$: 資源の集合

Prec: 作業間の先行順序関係を表す集合．$\mathcal{J} \times \mathcal{J}$ の部分集合であり，$(j,k) \in$ Prec ならば，作業 j の処理が終了するまで作業 k の処理が開始できないことを表す

3.4.2 入力データ

T: 計画期間

p_j: 作業 j の処理時間

C_{jt}: 作業 j を時刻 t に開始したときにかかる費用

R_{jr}: 作業 j の処理に要する資源 r の量

RUB_{rt}: 時刻 t における資源 r の使用可能量の上限

3.4.3 変数

x_{jt}: 作業 j を時刻 t に開始するとき 1，それ以外のとき 0 を持つ 0-1 変数

3.4.4 定式化

以下に資源制約付きスケジューリング問題の基本形の定式化を示す．目的関数は総費用最小化である．

$$\text{最小化} \quad \sum_{j \in \mathcal{J}} \sum_{t=1}^{T-p_j+1} C_{jt} x_{jt}$$

$$\text{条件} \qquad \text{作業遂行条件}$$

$$\text{資源制約}$$

先行順序制約

変数の 0-1 条件

作業遂行条件:

$$\sum_{t=1}^{T-p_j+1} x_{jt} = 1 \quad \forall j \in \mathcal{J} \tag{3.1}$$

すべての作業は，計画期間中に必ず1度処理されなければならないことを表す．作業jが開始される時刻はちょうど1度であることから，上式を得る．

資源制約:

$$\sum_{j \in \mathcal{J}} R_{jr} \sum_{s=\max\{t-p_j+1,1\}}^{\min\{t,T-p_j+1\}} x_{js} \leq RUB_{rt} \tag{3.2}$$
$$\forall r \in \mathcal{R}, t = 1, \cdots, T$$

すべての資源rにおいて，ある時刻 t に処理中である作業の資源使用量R_{jr}の合計が，時刻tの資源使用可能量の上限RUB_{rt}を超えないことを表す．時刻 t に処理中の作業は，時刻 $t - p_j + 1$ から時刻 t の間に処理を開始した作業であることから，上式を得る．

資源r を使用できる時刻が b_r から f_r の間と定められている場合には，

$$RUB_{rt} = 0 \quad \forall t < b_r \text{ または } t > f_r$$

と設定する．

先行順序制約:

$$x_{jt} \leq \sum_{s=t+p_j}^{T-p_k+1} x_{ks} \quad \forall (j,k) \in \mathrm{Prec}, \tag{3.3}$$
$$t = 1, \cdots, T - p_j + 1$$

作業j の処理が完了するまで，作業k の処理が開始できないことを表す．作業j が時刻 t に作業を開始したとき（すなわち $x_{jt} = 1$ のとき），作業k の開始時刻 s は，時刻 $t + p_j$ 以降でなければならないことから，上式を得る．

変数の 0-1 条件

$$x_{jt} \in \{0, 1\} \quad \forall j \in \mathcal{J}, t = 1, \cdots, T - p_j + 1 \tag{3.4}$$

変数x_{jt}が，0または1のみを持つことを表す．

3.5 資源制約付きスケジューリング問題の図による表現

資源制約付きスケジューリング問題を，ガントチャート（図3.2）とリソースグラフ（図3.3)の2種類の図を用いて表現する．

ガントチャート（図3.2）では，各作業の属性や作業間の関係を表現できる．

図3.2では，縦軸に作業，横軸に時刻がとられている．作業は，J_1，J_2，J_3の3つである．横棒は作業を表し，棒の長さは 作業の処理時間を意味する（作業J_1の処理時間は2)．棒の位置が処理を行う時刻を表す（作業J_2は時刻3から時刻4の間で処理される)．棒の中に書かれた文字は，作業を処理するとき必要な資源の名前とその使用量を表す（作業J_2を処理するとき，資源R_2が1必要)．また棒と棒（すなわち作業と作業）をつなぐ矢印（→）は，先行順序を表す（作業J_1の処理が終了しなければ，作業J_2の処理は開始できない)．

リソースグラフ（図3.3）では，資源の使用可能量の上限と，各時刻における資源の使用量を資源ごとに表現できる．

図3.3では，縦軸に資源の量，横軸に時刻がとられている．棒が各時刻の資源を表し，棒の長さが資源の使用量をあらわす．また，棒の中に書かれた文字は，資源を使用中の作業を表す（時刻1において，資源R_1の使用量は1，使用中の作業はJ_1)．

一方，時間軸に平行に書かれた直線が資源の使用可能量の上限を表す（資源R_1，R_2の使用可能量の上限は，それぞれ2，3)．すなわち，棒が直線の上に出たならば，資源の使用量が使用可能量の上限を破っていることを表す．

ここでは，資源の使用可能量の上限を直線で表したが，資源の使用可能量の上限が時間により変化する場合は，階段関数となる．

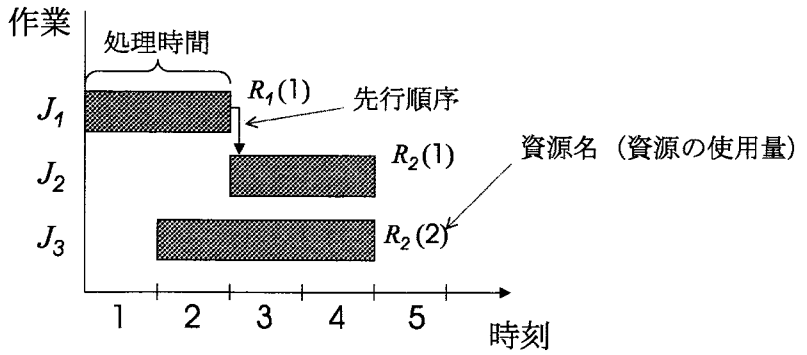

図 3.2: ガントチャート

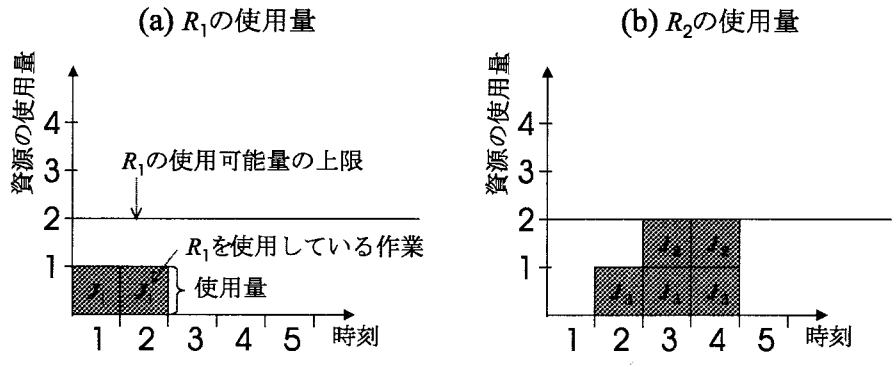
(a) R_1の使用量
資源の使用量
4
3
2
1
R_1の使用可能量の上限
R_1を使用している作業
使用量
1 2 3 4 5
時刻
(b) R_2の使用量
資源の使用量
4
3
2
1
1 2 3 4 5
時刻

図 3.3: リソースグラフ

第 4 章

スケジューリング・システム

本章では，スケジューリング・システムの定義を示す．

4.1 スケジューリング・システムの定義

本研究では，スケジュールを計画・管理を計算機上で行うためのシステム（手法）を，スケジューリング・システムと呼ぶこととする．計算機上を利用することにより，規模が大きく，複雑なスケジュールの計画・管理も行うことが可能となる．

既存のスケジューリング・システムを汎用性の広さなどの特徴により大別すると，スケジュール管理システムとスケジュール最適化システムの2つに分類できる（表4.1）．

表 4.1: スケジューリング・システムの一般的特徴

項目	スケジュール管理システム	スケジュール最適化システム
汎用性	高い	低い
最適化	導入されていない	導入されている
開発者	企業	研究者

4.2 スケジュール管理システム

スケジュール管理システムとは，スケジュールの管理を目的とし，計画から進捗状況の確認などを行えるシステムである．スケジュールを管理することに重点を置いているため，最適化は導入されていない．また，企業において作られたスケジューリング・システムに多い．

このシステムをパッケージ化したアプリケーション・ソフトが多く販売されており，誰でも購入することが可能である．パッケージ化されたスケジュール管理システムを，スケジュール管理ソフトと呼ぶこととする．スケジュール管理ソフトは，そのほとんどが

GUI（Graphical User Interface）環境が整っており，誰にでも使いやすいという特徴を持っている．

既存のスケジュール管理ソフトには，Project98（Microsoft社），PJMS/R4.0（日立システム）などがある（表4.2）．

表 4.2: 既存のスケジュール管理ソフト

スケジュール管理ソフト名	販売元
Project98	Microsoft
ILOG Scheduler	ILOG
PJMS/R4.0	日立システム
ASPROVA	スケジューラー研究所
しんすけ	西岡靖之

4.3 スケジュール最適化システム

スケジュール最適化システムとはスケジュールの最適化を目的するシステムである．このシステムは，企業の要請により研究者が作成したシステムに多い．通常それぞれの問題に対し特化されているため，汎用性は低い．また，ほとんどのスケジュール最適化システムには，使いやすいGUI環境が整っていない．個々の研究者によって作成されたスケジューリング・システムに多い．

近年，野々部・茨木[6]により，汎用性の高いシステムも開発されている．

第 5 章

従来の研究

本章では，本研究で用いる解法に関する従来の研究を示す．

5.1 資源制約付きスケジューリング問題

資源制約付きスケジューリング問題は，多くのスケジューリング問題を一般化することが可能であり，この汎用性の高さから，近年注目されている問題である．

近年の資源制約付きスケジューリング問題に関する研究に，野々部・茨木 [6] がある．

5.2 拡張モデル

本研究では，資源制約付きスケジューリング問題の基本形を，制約条件の追加などにより拡張したモデルのことを，拡張モデルと呼ぶ．

先行順序関係を一般化した研究に Sausa ら [7] の研究がある．

5.3 メタ解法

数理計画問題の中に，離散型の最適化を行う問題として，組合せ最適化問題がある．

組合せ最適化問題の解法は，厳密解法と近似解法に分けられる．厳密解法は最適性の保証のある解法だが計算に時間がかかり，一方近似解法は最適性の保証がない解法だが厳密解法に比べ計算時間は短い．

近年では，精度保証（例えば，最適値の 1.2 倍を超えない値を必ず得ることができる）がある近似解法に関する研究が多くされている．

メタ解法（または，メタヒューリスティック，メタ戦略，モダンヒューリスティック）とは，組合せ最適化問題を解くための近似解法を有機的（相互に関連を持つよう）に結合させたものであり，従来の近似解法を超えた新しい解法である．

代表的なメタ解法に，局所探索法，遺伝的アルゴリズムがある．

5.3.1 局所探索法

局所探索法（Local Search）とは，最も単純なメタ解法である．

局所探索法の基本構造は極めて簡単であり，構造を説明するとき暗い夜道を懐中電灯を頼りに山登りをする人に例えられることから，丘登り法（Hill Climbing Method）とも呼ばれる．ここでは，最小化問題を考えて，なるべく標高の低い地点を探す登山者（下山者）を例に用いて局所探索法の概要を説明する．

いま，登山者は真夜中に山の中にいるものとする．ここでの目的は，限られた時間内になるべく標高の低い地点に到着することである．夜道は暗いので，現在自分のいる場所のまわり以外は見ることができない（懐中電灯でまわりだけは見ることができ，懐中電灯で照らせる範囲を近傍と呼ぶ）．現在地点よりも標高が低い地点がまわりにあれば，その地点へ移動する．照らした範囲（近傍）内に，現在地点よりも標高が低い地点がまわりになければ，あきらめて寝袋を出して寝る．そのときの標高が登山者のスコア（目的関数値）になる．

簡単にわかるように，この方法ではいつも最も低い地点で眠れるとは限らない．我々の考えている山は，いくつもの谷があるからである．局所探索法が捉えられてしまう谷が局所最適解に対応する．

局所探索法に関する研究は多くされており，多出発局所探索法（Multiple Start Local Search）や反復局所探索法（Iterated Local Search）などの応用もされている．

5.3.2 遺伝的アルゴリズム

遺伝的アルゴリズム（Genetic Algorithm）は複数の実行可能解（すべての制約条件を満たす解）を保持し，その解を生物進化の過程を模写したアルゴリズム用いて改善していく方法の総称である．古典的な遺伝的アルゴリズム[2]では，解を0，1のビット列として表していたが，最近では問題の構造にあわせた任意の方法で解を表現する遺伝的アルゴリズムの変形（進化的アルゴリズムと呼ばれる）も多く用いられている[5]．

遺伝的アルゴリズム特有の用語を用いると，保持する実行可能解を個体と呼び，個体の集合を集団と呼ぶ．個体は有限の数列で表され，その数列の1つの要素を遺伝子と呼ぶ．

基本的な遺伝的アルゴリズムでは，選択（selection）により選ばれた個体に対し，交叉（crossover），突然変異（mutation）と呼ばれる2つの操作を用いて新たな個体を生成し，集団に加える．さらに各個体の適応度を算出し，適応度の低い個体を淘汰（集団から削除）する．このような操作を繰り返し行うことにより，より良い集団を得る解法である．一連の操作を繰り返す回数は，世代数と呼ばれる．

個体jの適応度$Adapt_j$は，

$$Adapt_j = k \times f_{max} - f_{min} + f_j \tag{5.1}$$

と計算される．ただし，kは重み（一定），f_{max}（f_{min}）は集団内の個体が持つ目的関数値の最大値（最小値），f_jは個体jが持つ目的関数値である．

1. 選択

   集団の中から個体を選ぶ操作である．この操作により選ばれた個体に対し，交叉，

突然変異を行う．代表的な選択方法として，ルーレット選択がある．ルーレット選択とは，適応度の高い個体を優先して選択する方法である．

2. 交叉

   交叉は，2つの個体の遺伝子を組み替えて，新たな個体を生成する操作である（図5.1）．代表的な交叉方法に，単純交叉，複数点交叉，一様交叉がある．どのような交叉が良いかについては，Syswerda[3]やSpearsら[4]の研究がある．

3. 突然変異

   突然変異は，1つの個体の遺伝子を一定の確率で変化させ，新たな個体を生成する操作である（図5.2）．最も単純な突然変異方法は，個体内の1つの遺伝子のみを一定の確率で変化させる方法である．

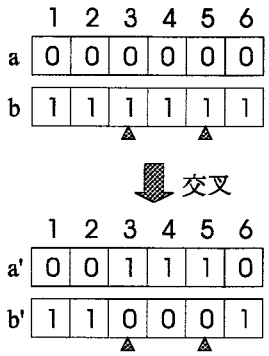

図 5.1: 交叉

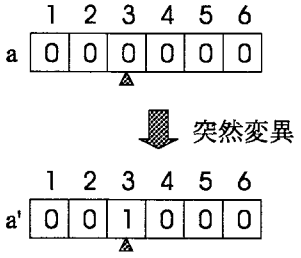

図 5.2: 突然変異

5.4 その他の解法

5.4.1 線形計画緩和法

緩和法とは，制約条件の一部を無視（緩和）した問題を解いて，原問題の下界（最小化問題の場合）を得る方法であり，整数計画問題の整数条件を無視（緩和）する場合を線形計画緩和法という．

例えば，ある整数計画問題が0-1変数xを持つとき，

$$0 \leq x \leq 1$$

というように，xを0から1の間の実数値をとる変数にすることである．

5.4.2 α-Point法

α-Point法とは，線形計画緩和法により原問題の近似解を求め，その値をαという定数を用いて制御することにより，より良い解を求める解法である[7]．

例えば，ある整数計画問題が0-1変数$x_t(t = 1, \cdots, T)$を持つとし，x_tには，

$$\sum_{t=1}^{T} x_t = 1$$

という条件があるとする．

このx_tを線形計画緩和し，0から1までの実数を持つ変数とした場合の問題を解くと近似解が生成される．原問題ではx_tは整数であるため，線形計画緩和により算出された解は，原問題の解として直接用いることはできない．そこで，α-Point法やその他の何らかの方法を用いて線形計画緩和により得た解を変換し，原問題の解を生成する必要がある．

α-Point法では，線形計画緩和による解を表す変数x'_tと，関数z_tを導入する．また，定数$\alpha(0 < \alpha \leq 1)$を設定する．関数$z_t$は，

$$z_t = \sum_{s=1}^{t} x_s \qquad t = 1, \cdots, T$$

と設定される．

関数z_tが，

$$z_{t'-1} < \alpha \text{かつ} z_{t'} \geq \alpha$$

となるならば，

$$x_t = \begin{cases} 1 & t = t' \\ 0 & t \neq t' \end{cases} \qquad t = 1, \cdots, T$$

とする．これにより，原問題に対する実行可能な近似解を生成することができた．

α-Point法は，線形計画緩和解の近傍に原問題の解が存在するという発想による解法である．

第 6 章

本研究で用いるアプリケーション・ソフト

本章では，本研究で用いるアプリケーション・ソフトの概要について述べる．本研究では，以下の3種類のアプリケーション・ソフトを使用する。

6.1 Project98

Project98は，Microsoft社が開発したスケジュール管理ソフトであり，様々な規模のスケジュールを効率良く管理できる（図6.1）．実際の現場の問題に適用できるように様々なオプション機能がつけられている．ただし，最適化を行う機能は含まれていない．

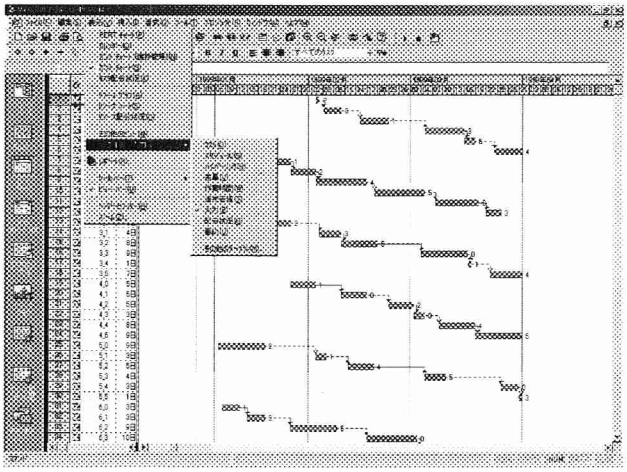

図 6.1: Project98の実行画面

6.1.1 オプション機能

Project98には，様々なオプション機能がついている．[8]

- 平準化機能
  6.1.2参照.
- カレンダー機能
  1日単位，1週間単位，1年単位など，様々な期間を設定でき，見やすい表示を行える．
- PERT機能
  PERT（Program Evaluation and Review Technique）とは，米海軍のSpecial Projects Officeによって開発されたスケジュール管理の方法論である．スケジュール全体をアローダイアグラムで表し様々な操作を行うことにより，スケジュールを管理する方法であり，Project98上では簡単に実現できる．
- Excelやインターネットとのリンク
  Excelやインターネットとのリンクが可能である．Excel上のデータをインポートすることが可能であり，インターネットを利用して企業内イントラネットと接続することもできる．

6.1.2 平準化機能

平準化とは，使用可能量の上限が破られている資源について，その資源に割り当てられている作業のうちのいくつかの処理を延期することにより資源の使用量の超過をなくし，すべての資源を均等に配分することをいう（図6.2）．

どの作業を優先して早い時刻に割り当てるかは，以下のような基準に従い決定する．ただし，各作業にはID番号(作業固有の正数)と優先度値(1〜9の整数，値の小さい方が優先度が高い)が与えられている．

ID順

作業ID番号の小さい作業を優先する（図6.3）．

標準方式

Project98の初期状態での平準化基準．以下に示す5つの要素により，優先する作業を決定する．要素を重要な順序にあげる（図6.4）．

1. **先行タスク** 依存関係のある作業を優先する．
2. **総余裕期間量** 総余裕期間が少ない作業を優先する．
3. **リリース時刻** 早く開始する作業を優先する．
4. **優先度値** 優先度の高い作業を優先する．
5. **制約条件** 制約条件のある作業を優先する．

優先度順＋標準方式

標準方式とほぼ同じだが，優先度値を一番重要とする（図6.5）.

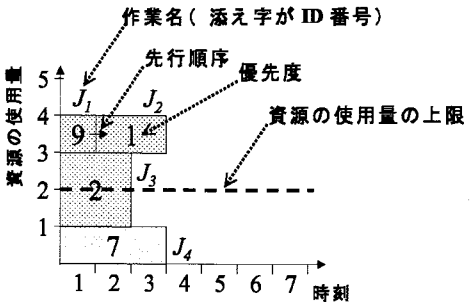

図 6.2: 資源の使用可能量が超過している例

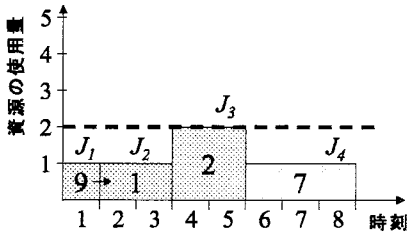

図 6.3: ID順による平準化

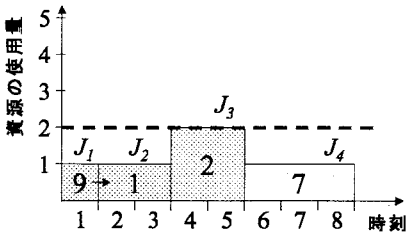

図 6.4: 標準方式による平準化

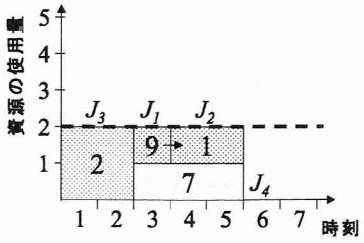

図 6.5: 優先度順＋標準方式による平準化

6.2 AMPL

AMPLは，数理計画モデル記述言語である（図6.6）．数理計画モデル記述言語とは，定式化された数理計画問題を計算機上で表現する言語である．この言語で記述することによって，(1)問題の構造に対する理解を深めることや，(2)計算機を用いて最適化を行うこと，の2つが可能になる．

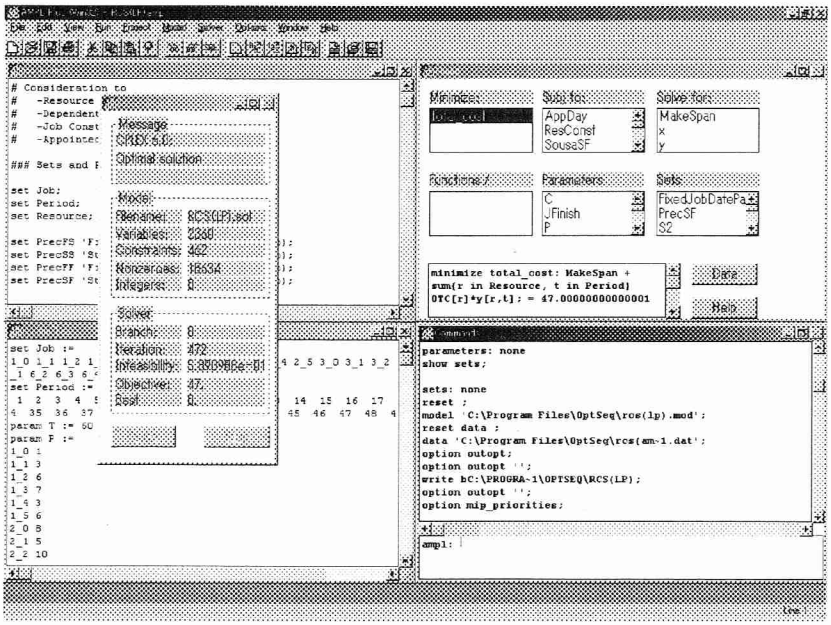

図 6.6: AMPLの実行画面

6.3 CPLEX

CPLEXは数理計画ソルバーである．数理計画ソルバーとは，数理計画モデル記述言語により記述された数理計画問題の解を求めるアプリケーション・ソフトである．

CPLEXは，現在最速の数理計画ソルバーと言われている．

第 7 章

拡張モデル

本章では，拡張モデルを用いるシステムを提案する．

7.1 拡張モデルを用いるシステムの概要

本研究では，資源制約付きスケジューリング問題の基本形に，目的関数の変更，または制約条件の変更・追加を行い拡張したものを拡張モデルと呼ぶ．

拡張モデルでは，以下のような様々な条件を考慮することができる．

- 時間依存の資源使用量の考慮
  作業を開始してからの時間により，処理のために必要な資源使用量が変化する場合．
- 段取り時間の考慮
  作業間に一定の空き時間が必要な場合．例えば，機械のセットアップに要する時間．
- 様々な先行順序関係の考慮
  作業間に様々な先行順序関係がある場合．ここでは先行順序関係を一般化した定式化を示す．
- 資源に対する超過費用の考慮
  資源の超過を許し，超過した量に応じたペナルティ費用を目的関数値に追加する場合．
- リリース時刻と納期の考慮
  作業を開始できる最早の時刻（リリース時刻）と，作業を終了しなければならない最遅の時刻（納期）を考慮する場合．

拡張モデルを用いるシステムでは，拡張モデルを数理計画ソルバーを用いて求解する．本研究では数理計画ソルバーとして，CPLEX（CPLEX社）を用いる．

拡張モデルに対して数理計画ソルバーを適用するためには，モデル記述言語への変換を行わなければならない．

しかし，基本形の定式をモデル記述言語に変換しておけば，拡張する部分のみモデル記述言語への変換を行えば良いので，拡張は簡単に行える．資源制約付きスケジューリング問題の基本形の定式をモデル記述言語に変換した例を，付録Aとして示す．

拡張モデルを用いるシステムの解の精度は，数理計画ソルバーに依存している．しかし，既存の数理計画ソルバーでは，規模の大きな整数計画問題の解を得ることは困難である．

そこで本研究では，拡張が容易な定式化と数理計画ソルバーを組み合わせることにより，原問題の近似解を利用し，効率の良いスケジュールを計画するシステム（手法）を提案する．

拡張モデルを用いたシステムとして，(1)線形計画緩和解を用いるシステムと，(2)実行可能な最良解を用いるシステムを示す．

7.2 拡張モデルを用いるシステムの構成

拡張モデルを用いるシステムは，基本部分，拡張部分，求解部分の3つに分けられる（図7.1）．基本部分は，資源制約付きスケジューリング問題の基本形の定式化から，その定式を数理計画モデル記述言語で記述する．

拡張部分は，おのおのの問題に対する固有の要素を抽出し，定式化，モデル記述言語化する．

求解部分は，基本部分，拡張部分で生成したモデルにデータを加え，数理計画ソルバーを使い，効率の良いスケジュールを計画する．このとき，数理計画ソルバーの性能上の都合により，実行可能な最良解と線形計画緩和解を利用する．

7.3 線形計画緩和解を用いるシステム

線形計画緩和解を用いるシステムとして，(i)α-Point法を用いるシステムと，(ii)最大値を用いるシステムの2つを示す．

7.3.1 α-Point法を用いるシステム

線形計画緩和による解を表す変数x'_{jt}と，関数z_{jt}を導入する．また，定数$\alpha(0 < \alpha \leq 1)$を設定する．関数$z_{jt}$は，

$$z_{jt} = \sum_{s=1}^{t} x'_{js} \qquad \forall j \in \mathcal{J}, t = 1, \cdots, T$$

と設定される．

関数z_tが，

$$z_{j,t'-1} < \alpha \text{かつ} z_{jt'} \geq \alpha$$

となるならば，

$$x_{jt} = \begin{cases} 1 & t = t' \\ 0 & t \neq t' \end{cases} \qquad \forall j \in \mathcal{J}, t = 1, \cdots, T$$

とする．これにより，原問題に対する近似解を生成することができる．

7.3.2 最大値を用いるシステム

線形計画緩和による解を表す変数をx'_{jt}としたとき，

$$x'_{jt'} = \max_t \{x'_{jt}\}$$

ならば，

$$x_{jt} = \begin{cases} 1 & t = t' \\ 0 & t \neq t' \end{cases}$$

とする．これにより，原問題に対する近似解を生成することができる．

7.4 実行可能な最良解を用いるシステム

実行可能な最良解を用いるシステムは，計算時間の上限や反復回数の上限を設定し，計算を中断する．計算が中断されるまでに実行可能解が見つかっていれば，そのうちで最も最適値に近い目的関数値を持つ解を近似解とする．

7.5 拡張モデルの定式化

ここでは，前節で紹介した拡張モデルの例に対する定式化を行う．

7.5.1 時間依存の資源使用量

ここでは，作業の処理を開始した後で，使用する資源の量が時間の経過とともに変化する場合への拡張を行う．

R_{jrt}: 処理開始後 t 時間経過したときに，作業 j を処理するときに要する資源 r の量．ここで，処理開始時刻は 0 時間経過とみなす．

資源制約（式（3.2））を以下のように変更する．

資源制約:

$$\sum_{j \in \mathcal{J}} \sum_{s=\max\{t-p_j+1,1\}}^{\min\{t,T-p_j+1\}} R_{jr,t-s} x_{js} \leq RUB_{rt} \qquad \forall r \in \mathcal{R}, t = 1, \cdots, T \tag{7.1}$$

時刻tにおいて，資源rの使用量はその使用可能量の上限RUB_{rt}を越えないことを表す．

7.5.2 段取り時間の考慮

作業jの処理が終了した後に作業kの処理を開始する際，一定時間s_{jk}の空きが必要な場合がある．このs_{jk}を段取り時間（setup time）と呼ぶ．段取り時間を考慮するためには，先行順序制約を以下のように変更する．

先行順序制約:

$$x_{jt} \leq \sum_{s=t+p_j+s_{jk}}^{T-p_k+1} x_{ks} \quad \forall (j,k) \in \mathrm{Prec}, \quad t = 1, \cdots, T-p_j+1 \tag{7.2}$$

作業jの処理が終了してから段取り時間s_{jk}が経過するまで，作業kの処理が開始できないことを表す．

7.5.3 先行順序関係の一般化

2つの作業の依存関係として次の4つを考える(図7.3)．

終了 - 開始 (FS): 先行作業jが終了してから，ほかの作業kを開始しなければならない．この依存関係がある作業の組の集合をPrec_{FS}と記す．これは，基本形で用いた先行順序関係 Prec に他ならない．

依存関係 開始 - 開始 (FS):

$$x_{jt} \leq \sum_{s=t+p_j}^{T-p_k+1} x_{ks} \quad \forall (j,k) \in \mathrm{Prec}_{FS}, \quad t = 1, \cdots, T-p_j+1 \tag{7.3}$$

作業jの処理が終了するまで，作業kの処理が開始できないことを表す．作業jが時刻tに作業を開始したとき（すなわち$x_{jt}=1$のとき），作業kの開始時刻sは，時刻$t+p_j$以降でなければならないことから，上式を得る．

開始 - 開始 (SS): 先行作業の開始と同時に，ほかの作業を開始しなければならない．この依存関係がある作業の組の集合をPrec_{SS}と記す．

依存関係 開始 - 開始 (SS):

$$x_{jt} = x_{kt} \quad \forall (j,k) \in \mathrm{Prec}_{SS}, \quad t = 1, \cdots, T-\max\{p_j, p_k\}+1 \tag{7.4}$$

作業jの処理の開始と同時に，作業kの処理を開始することを表す．

終了 - 終了 (FF): 先行作業の終了と同時に，ほかの作業が終了しなければならない．この依存関係がある作業の組の集合をPrec_{FF}と記す．

依存関係 終了 - 終了 (FF):

$$x_{j,t-p_j+1} = x_{k,t-p_k+1} \quad \forall (j,k) \in \mathrm{Prec}_{FF}, \quad t = \max\{p_j, p_k\}, \cdots, T \tag{7.5}$$

作業 j の処理の終了と同時に，作業 k の処理を終了することを表す.

開始 - 終了 (SF): 先行作業が開始してから，ほかの作業が終了しなければならない．この依存関係がある作業の組の集合を Prec_{SF} と記す.

依存関係 開始 - 終了 (SF):

$$x_{jt} \leq \sum_{s=\max\{1,t-p_k+1\}}^{T-p_k+1} x_{ks} \quad \forall (j,k) \in \mathrm{Prec}_{SF}, \quad t = p_k + 1, \cdots, T - p_j + 1 \tag{7.6}$$

作業 j の処理が開始するまで，作業 k の処理が終了できないことを表す.

7.5.4 資源に対する超過費用の考慮

資源の使用量が使用可能量の上限 RUB_{rt} を超過したときに，超過量1単位あたり PEN_r のペナルティ費用がかかるものとする.

以下の変数を導入する.

y_{rt}: 時刻 t に資源 r の使用可能量の上限を逸脱した量

資源に対する超過費用を考慮する場合は，目的関数，資源制約をそれぞれ以下のように変更し，さらに変数 y_{rt} の制約を追加する.

目的関数

$$\text{最小化} \sum_{j \in \mathcal{J}} \sum_{t=1}^{T-p_j+1} C_{jt} x_{jt} + \sum_{r \in \mathcal{R}} \sum_{t=1}^{T} PEN_r y_{rt} \tag{7.7}$$

資源制約

$$\sum_{j \in \mathcal{J}} R_{jr} \sum_{s=\max\{t-p_j+1,1\}}^{\min\{t,T-p_j+1\}} x_{js} \leq RUB_{rt} + y_{rt} \quad \forall r \in \mathcal{R}, t = 1, \cdots, T \tag{7.8}$$

(7.9)

整数条件

$$y_{rt} \geq 0 \quad \forall r \in \mathcal{R}, t = 1, \cdots, T \tag{7.10}$$

7.5.5 リリース時刻と納期の考慮

実際問題においては，作業の開始時刻や終了時刻に制限がつけられる場合がある．作業jが処理を開始することができる最早の時刻をリリース時刻と呼び，r_jと記す．作業jの処理が終了することが望ましい最遅の時刻を納期と呼び，d_jと記す．

リリース時刻と納期を遵守する場合には，定式化は以下のように変形される．

目的関数:

$$\text{最小化} \quad \sum_{j\in\mathcal{J}} \sum_{t=r_j}^{d_j-p_j+1} C_{jt}x_{jt} \tag{7.11}$$

条件:

$$\sum_{t=r_j}^{d_j-p_j+1} x_{jt} = 1 \quad \forall j \in \mathcal{J} \tag{7.12}$$

$$\sum_{j\in\mathcal{J}} R_{jr} \sum_{s=\max\{t-p_j+1,r_j\}}^{\min\{t,d_j-p_j+1\}} x_{js} \le RUB_{rt} \tag{7.13}$$
$$\forall r \in \mathcal{R}, t \in 1,\cdots,T$$

$$x_{jt} \le \sum_{s=\max\{t+p_j,r_k\}}^{d_k-p_k+1} x_{ks} \quad \forall (j,k) \in \mathrm{Prec}, \tag{7.14}$$
$$t = r_j,\cdots,d_j-p_j+1$$

$$x_{jt} \in \{0,1\} \quad \forall j \in \mathcal{J}, t = r_j,\cdots,d_j-p_j+1 \tag{7.15}$$

7.5.6 様々な目的関数の考慮

納期以降に作業が終了したときのペナルティを考慮する場合には，処理開始費用C_{jt}を適当に設定すればよい．たとえば，作業jに対して重み（重要度）w_jが与えられており，納期を経過した時間にw_jを乗じただけのペナルティを費用と考える場合には，

$$C_{jt} = \begin{cases} 0 & t \le d_j - p_j + 1 \\ w_j\,(t - d_j + p_j - 1) & \text{それ以外} \end{cases}$$

とすればよい．上で定義したC_{jt}をすべての作業に対して合計した目的関数は，重みつき総納期遅れ基準と呼ばれる．また，重みつき総納期遅れ作業数を最小化するためには，

$$C_{jt} = \begin{cases} 0 & t \le d_j - p_j + 1 \\ w_j & \text{それ以外} \end{cases}$$

とすればよい．最大納期遅れ最小化などのmin-max基準への変更は，最大値を表すダミー変数の導入によって容易に可能である．

以下に制約タイプとその処理方法を示す．

できるだけ遅く： C_{jt}を$T-t+1$と設定する．

できるだけ早く： C_{jt}をtと設定する．

指定日以後に終了： リリース時刻r_jを指定日から処理時間p_jを減じた値にする．

指定日までに終了： 納期d_jを指定日から1を減じた値にする．

指定日以後に開始： リリース時刻r_jを指定日にする．

指定日までに開始： 納期d_jを指定日に処理時間p_jを加え，さらに1を減じた値にする．

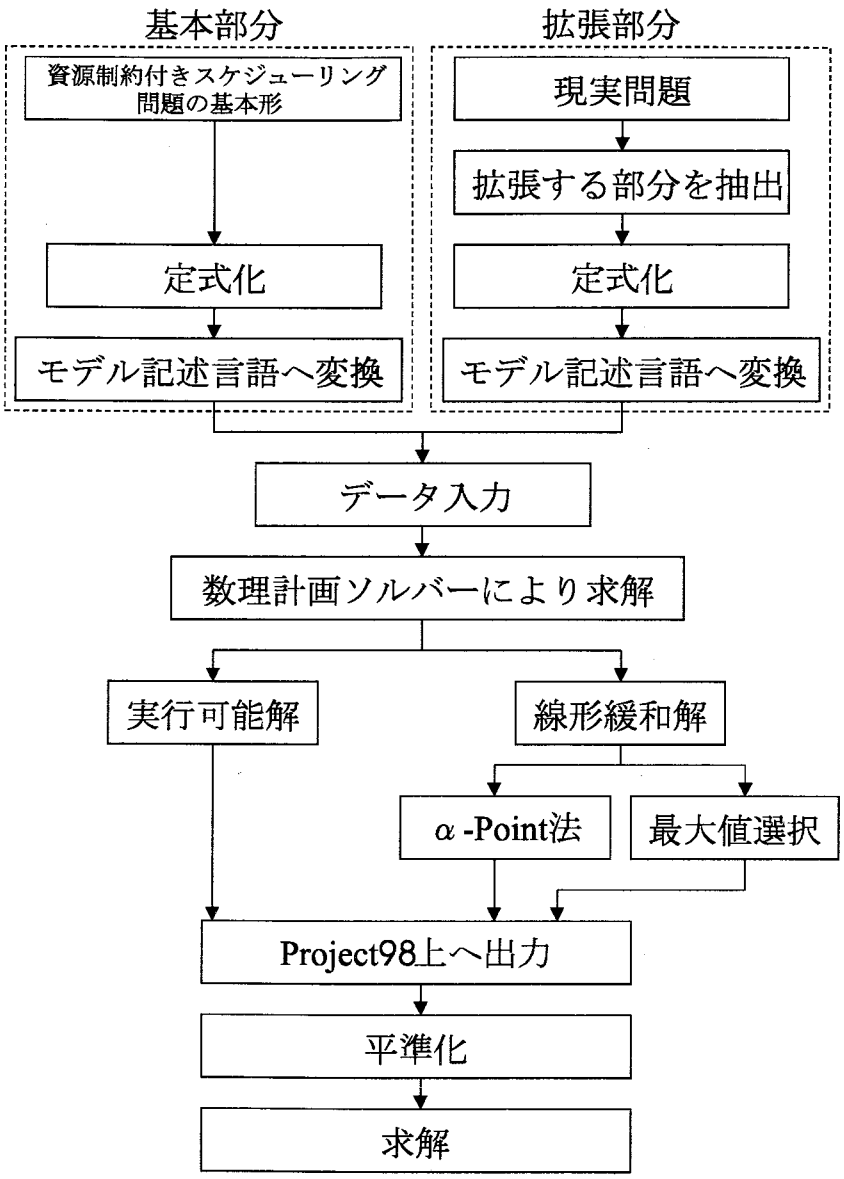

図 7.1: 拡張モデルを用いるシステムの構成

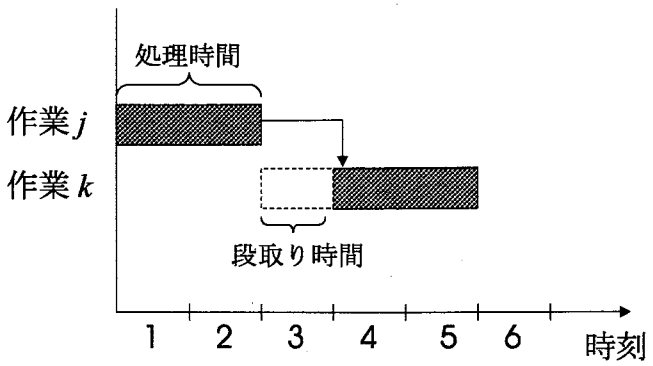

図 7.2: 段取り時間の考慮

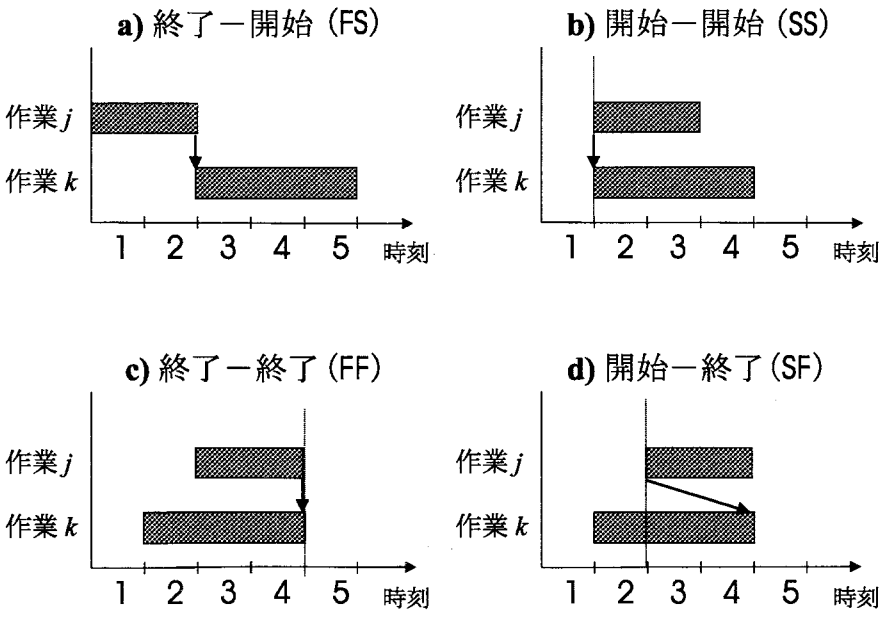

図 7.3: 先行順序の一般化

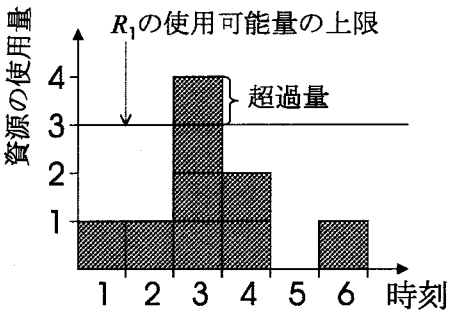

図 7.4: 資源に対する超過量の考慮

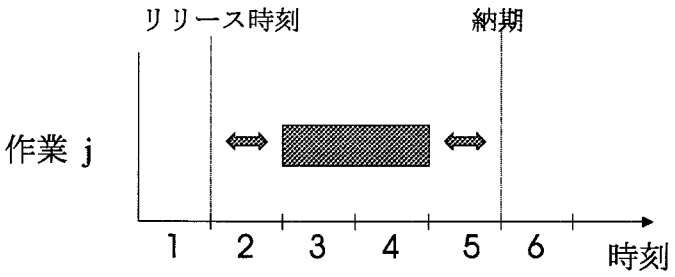

図 7.5: リリース時刻と納期の考慮

第 8 章

平準化機能を用いるメタ解法

本章では，市販のスケジュール管理ソフトとメタ解法を組み合わせるシステムを提案する．

8.1 市販のスケジュール管理ソフトとメタ解法を組み合わせるシステムの概要

本研究では，市販のスケジュール管理ソフトとしてProject98（Microsoft社）を，メタ解法として遺伝的アルゴリズムと局所探索法を用いる．

市販のスケジュール管理ソフトを利用するため，そのソフトが持つ機能をそのまま利用することができる．

8.2 市販のスケジュール管理ソフトとメタ解法を組み合わせるシステムの構成

市販のスケジュール管理ソフトとメタ解法を組み合わせるシステムでは，まず対象となる現実問題から必要なデータを抽出し，スケジュール管理ソフトに入力する．さらに，使用するメタ解法を決定し，スケジュール管理ソフトの機能（ここでは平準化機能）をその解法を用いて制御することにより，効率の良いスケジュールを計画する．

8.3 遺伝的アルゴリズムを用いるシステム

本システムでは，すべての作業に割り当てられた優先度をID番号の小さい順に並べた数列を1つの解，そのような優先度で平準化(優先度順)を行ったときの最大完了時刻（すべての作業が終了する時刻）を解の値とする（図8.3）．操作方法は，選択はルーレット選択，交叉は2点交叉，突然変異は1つの遺伝子(解の要素)をランダムに変化させる方法を用いた．[9]

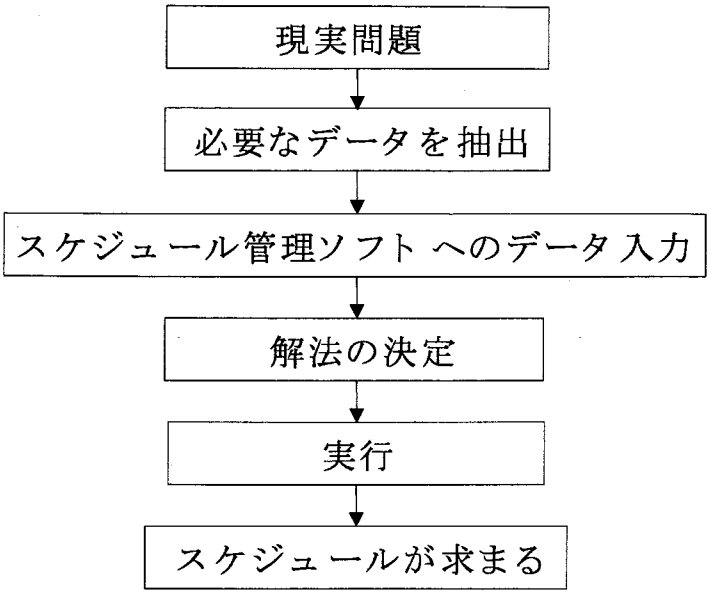

図 8.1: 市販のスケジュール管理ソフトとメタ解法を組み合わせるシステムの構成

以下に，Project98の平準化機能とGAを用いたアルゴリズムを示す．

PSは保持する実行可能解（すべての制約条件を満たす解）の個数を表し，$Iter$は反復回数を表す．

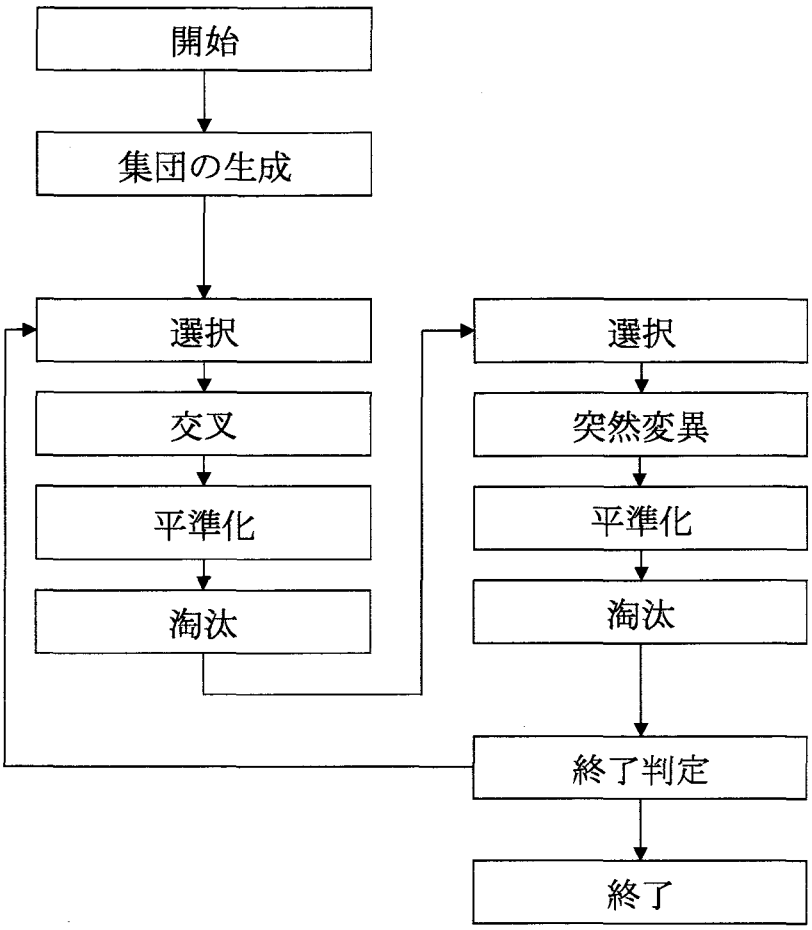

図 8.2: 遺伝的アルゴリズムの流れ

procedure GA

1 初期解をPS個生成する.
2 PS個の解の中から2つを選択し，交叉させ解を2つ生成する.
3 平準化し最大完了時刻を求める.
4 PS個の解と生成した2つの解のうち，最大完了時刻の小さい解をPS個残す.
5 PS個の解の中から1つを選択し，突然変異させ解を1つ生成する.
6 平準化し最大完了時刻を求める.
7 PS個の解と生成した1つの解のうち，最大完了時刻の小さい解をPS個残す.
8 手順2から手順7までを1世代とし，$Iter$世代まで繰り返す.

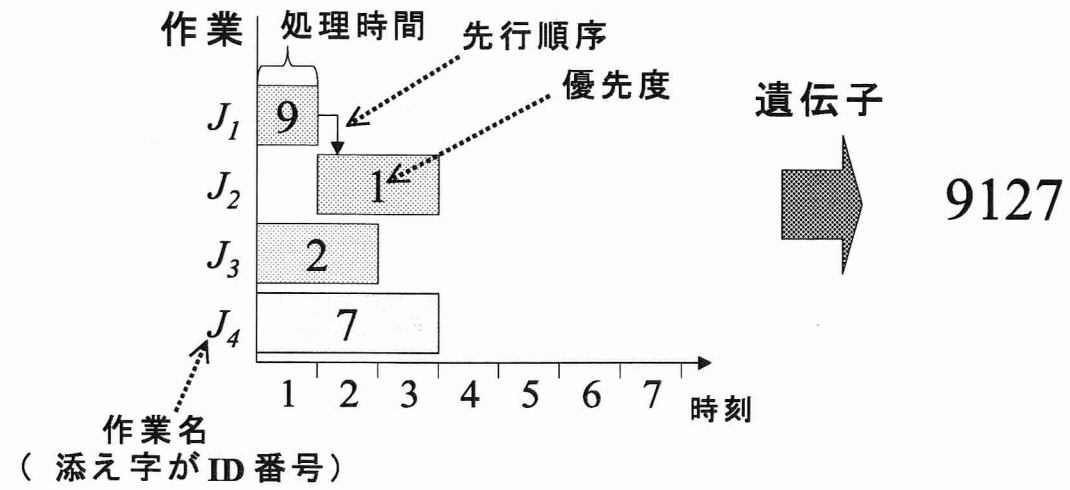

図 8.3: 遺伝子の表現

8.4 局所探索法を用いるシステム

各作業に割り当てられたID番号を入れ替えて，ID順の平準化を行うことにより，近似解を求める．

解の組み合わせを少し変えることにより目的関数値が小さくなれば，今度はその解を中心に組み合わせを変更する．目的関数値が小さくならなくなったとき終了する．

procedure 局所探索法

1 初期解を1個生成.
2 解の近傍を探索する.
3 平準化し，最大完了時刻を求める.
4 平準化を解除する.
5 現在の解より良い解が得られたら，得られた解に対し手順2へ.
6 近傍をすべて探索したならば終了.
7 手順2へ.

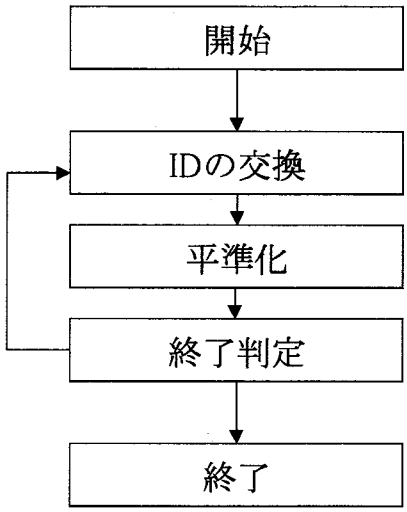

図 8.4: 局所探索法の流れ

第 9 章

システムの組合せ

第7章，第8章で提案したシステムを組み合わせることにより，新たなスケジューリング・システムを考案することができる．

例えば，拡張モデルを用いるシステムにより算出されたスケジュールを初期解に利用して，市販のスケジュール管理ソフトとメタ解法を組み合わせるシステムに導入し，スケジュールを計画するシステムである．

第 10 章

計算実験

本章では，提案したシステムに対する計算実験の結果を示す．

10.1 計算実験の手順

計算実験を行うため，拡張モデルを用いるシステムおよび市販のスケジュール管理ソフトとメタ解法を組み合わせるシステムをそれぞれ実装（プログラミング）する．

2つのシステムともに，スケジュール管理ソフト Project98（Microsoft 社）上の VBA（Visual Basic for Apprications）を用いて実装した．

システムの有用性を調べるためには様々な種類の問題に対して提案するシステムを適用してみる必要がある．そこで，複数のベンチマーク問題に対して提案するシステムを適用し，システムの有用性を探る．また，問題例の規模により，提案するシステムを用いて算出した最大完了時刻がどのように変化するかを明らかにする．

10.2 実験環境

実験は，VIP6300ST-AGP（TWOTOP，CPU:300MHz，メモリ:128MB）上で行った．

10.3 実験に使用した問題例

本研究では提案するシステムを汎用性を示すため，複数のベンチマーク問題に対し実験を行う．ベンチマーク問題はインターネットのホームページ上より入手した．

ここでは，ジョブショップ問題と資源制約付きスケジューリング問題に対する実験の結果を示す．

ジョブショップ問題の問題例は，OR-Library[1] より入手した．これらの問題例は，最大完了時刻（すべての作業が完了する時刻）を最小化することが目的であり，最適値がすでに分かっているものを用いる．

[1] http://mscmga.ms.ic.ac.uk/info.html

ジョブショップ問題の問題例のファイル名，規模および最適値を，表10.1に示す．ft06は，仕事数，資源数それぞれ少なく規模の小さい問題である．ft10，orb01は，仕事数，資源数それぞれ多く規模の大きな問題である．ft20,la01は，仕事数に対し資源数が少ない問題である．

資源制約付きスケジューリング問題（以下RCSPと呼ぶ）の問題例は，PSPLIB[2]より入手した．これらの問題例は，最大完了時刻を最小化することが目的であり，最適値（最小値）がわかっていないものを用いる．現時点までに分かっている最大完了時刻の中で，最小のものを最良値と呼ぶ．

RCSPの問題例のファイル名，規模および最良値を，表 10.2に示す．すべての問題例が資源数4を持ち，j301_1，j601_1，j901_1，j1201_1という順で，作業数が多くなる．

ジョブショップ問題とRCSPの問題例を比較すると，大きな違いが2つある．1つは最適値（最良値）の違いであり，もう1つは先行順序関係の違いである．

最適値（最良値）の違いをみると，ジョブショップ問題の問題例の最大完了時刻の方が，RCSPの問題例の最大完了時刻より，大きい値をとる．本研究で提案するシステムの1つである，拡張モデルを用いるシステムでは，時刻添え字付き定式化を利用している．そのため，対象とする計画期間が長いほど多くの変数を生成するため，求解することが困難になるという問題がある．

表 10.1: ジョブショップ問題の問題例の規模と最適値

	問題例の規模			
ファイル名	仕事数	資源数	作業数	最適値
ft06	6	6	36	55
ft10	10	10	100	930
ft20	20	5	100	1165
la01	10	5	50	666
orb01	10	10	100	1059

表 10.2: RCSPの問題例の規模と最良解

	問題例の規模		
ファイル名	作業数	資源数	最良値
j301_1	32	4	43
j601_1	62	4	77
j901_1	92	4	73
j1201_1	122	4	107

[2]http://www.bwl.uni-kiel.de/prod/psplib/index.html

それぞれの問題例に対し，(他の操作を行わずに) 平準化のみを行ったときの最大完了時刻を示す (表10.3, 表10.4)．表中の，標準，優先度順，ID順は，それぞれ平準化の基準である，標準方式，優先度順+標準方式，ID順，を表す．また，値は最大完了時刻を，誤差は最適値(最良値) と最大完了時刻の誤差を表す．

誤差は次の式により求める．

$$\text{誤差 (\%)} = \frac{\text{最適値(最良値)} - \text{最大完了時刻}}{\text{最適値(最良値)}} \times 100 \tag{10.1}$$

以下，誤差とはこの式により算出された値を表す．

ジョブショップ問題に対する実験では，ft06, ft10, la01といった比較的規模の小さい問題例に対しては，標準，優先度順の平準化は誤差が小さく，ID順は誤差が大きくなった (表10.3)．逆にft20, orb01といった規模の大きな問題例に対しては，標準，優先度順，ID順の誤差が同じくらい大きかった．特に規模の小さな問題例に対しては誤差が大きかったID順が，規模の大きな問題例に対しては誤差が小さくなり，他の基準よりも誤差が小さくなった．

RCSPに対する実験では，ジョブショップ問題に対する実験と同様に，問題例の規模が小さいとき，標準，優先度順の平準化の誤差が小さく，ID順は誤差が大きくなった (表10.4)．また，問題例の規模が大きいときは，3つの基準ともに同じくらいの誤差であり，ID順の誤差が他の基準に比べ小さかった．

以上のことから平準化のみで最大完了時刻を算出すると，問題例の規模が小さいときは，標準，優先度順の平準化が誤差が小さく，問題例の規模が大きいときは，ID順の平準化が誤差が小さいことがわかった．

表 10.3: ジョブショップ問題に対する平準化のみの実験結果

		標準		優先度順		ID順	
ファイル名	最適値	値	誤差	値	誤差	値	誤差
ft06	55	56	1.82	56	1.82	73	32.73
ft10	930	1099	18.17	1099	18.17	1283	37.96
ft20	1165	1518	30.30	1518	30.30	1461	25.41
la01	666	701	5.26	701	5.25	1040	56.16
orb01	1059	1337	26.25	1337	26.25	1326	25.21

10.4 拡張モデルを用いるシステムの計算実験

10.4.1 α-Point法を用いるシステムの計算実験

ここでは，7.3.1で提案したα-Point法を用いるシステムの計算実験の結果を示す．

まずはじめに，パラメータであるα値の適正化を行った．α値を0.1から0.9まで，0.1刻みに変化させたとき，最大完了時刻が最小となるようなα値を求めた．問題例としては，

表 10.4: RCSPに対する平準化のみの実験結果

		標準		優先度順		ID順	
ファイル名	最良値	値	誤差	値	誤差	値	誤差
j301_1	43	43	0.00	43	0.00	49	13.95
j601_1	77	77	0.00	77	0.00	82	6.49
j901_1	73	81	10.96	81	10.96	91	24.66
j1201_1	107	132	23.36	132	23.36	129	20.56

問題の構造が違う，規模の小さな問題例としてft06とj301_1の2つを，規模の大きな問題例としてj120_1を用いた（表10.5，表10.6，表10.7）．

実験の結果，3つの問題例すべて，α値が大きくなると，すべての平準化の基準において最大完了時刻が多くなることがわかった．

ゆえにα値は小さいとき，すなわち0.1のとき適正であるとする．

表 10.5: ジョブショップ問題の問題例ft06（最適値55）に対するα値の適正化

	標準		優先度順		ID順	
α	値	誤差	値	誤差	値	誤差
0.1	56	1.82	56	1.82	68	23.64
0.2	59	7.27	59	7.27	69	25.45
0.3	59	7.27	59	7.27	65	18.18
0.4	70	27.27	70	27.27	73	32.73
0.5	69	25.45	69	25.45	72	30.91
0.6	69	25.45	69	25.45	72	30.91
0.7	77	40.00	77	40.00	86	56.36
0.8	78	41.82	78	41.82	86	56.36
0.9	83	50.91	83	50.91	94	70.91

α値は，0.1と設定する．

ジョブショップ問題に対する実験では，ソルバーの性能上実験できなかった．

RCSPに対する実験では，問題の規模に関係なく，ほとんど同じ誤差を得られた（表10.8）．

α-Point法を用いるシステムでは，問題例の規模に関係なく，ほとんど同じ誤差を得られた．

10.4.2 最大値を用いるシステムの計算実験

ここでは，7.3.2で提案した最大値を用いるシステムの計算実験の結果を示す．

表 10.6: RCSPの問題例j301_1（最良値43）に対するα値の適正化

	標準		優先度順		ID順	
α	値	誤差	値	誤差	値	誤差
0.1	43	0.00	43	0.00	43	0.00
0.2	43	0.00	43	0.00	49	13.95
0.3	51	18.60	51	18.60	50	16.28
0.4	51	18.60	51	18.60	50	16.28
0.5	51	18.60	51	18.60	51	18.60
0.6	59	37.21	59	37.21	59	37.21
0.7	73	69.77	73	69.77	64	48.84
0.8	69	60.47	69	60.47	70	62.79
0.9	69	60.47	69	60.47	70	62.79

ジョブショップ問題に対する実験では，ソルバーの性能上実験できなかった．

RCSPに対する実験では，問題の規模に関係なく，ほとんど同じ誤差を得られた（表10.9）．

α-Point法を用いるシステムでは，問題例の規模に関係なく，ほとんど同じ誤差を得られた．

10.4.3 実行可能な最良解を用いるシステムの計算実験

7.4で提案した実行可能解を用いるシステムでは，整数変数を用いるため問題が難しくなり，既存のソルバーにて実行可能解を算出することができなかった．

10.4.4 本システムの評価

拡張モデルを用いるシステムでは，α-Point法を用いるシステム，最大値を用いるシステム，実行可能な最良解を用いるシステムの3つを提案した．

実験の結果，計画期間の長い問題例に対しては，計算を行うことができなかった．拡張モデルを用いるシステムでは数理計画ソルバーを利用するため，得られる解の精度はほとんど数理計画ソルバーの性能に依存してしまう．数理計画ソルバー以外の点で改善できる方法として，変数の数を減らす定式化を行うなどがある．

表 10.7: RCSPの問題例j1201_1（最良値107）に対するα値の適正化

	標準		優先度順		ID順	
α	値	誤差	値	誤差	値	誤差
0.1	127	18.69	127	18.69	128	19.63
0.2	135	26.17	135	26.17	128	19.63
0.3	130	21.50	130	21.50	137	28.03
0.4	132	23.36	132	23.36	137	28.03
0.5	148	38.32	148	38.32	149	39.25
0.6	155	44.86	155	44.86	163	52.34
0.7	156	45.79	156	45.79	163	52.34
0.8	172	60.75	172	60.75	172	60.75
0.9	179	67.29	179	67.29	186	73.83

表 10.8: RCSPに対するα-Point法を用いたシステムの実験結果

		標準		優先度順		ID順	
ファイル名	最良値	値	誤差	値	誤差	値	誤差
j301_1	43	43	0.00	43	0.00	43	0.00
j601_1	77	93	20.78	93	20.78	81	5.19
j901_1	73	90	9.59	90	9.59	100	30.14
j1201_1	107	127	18.69	127	18.69	128	19.63

10.5 市販のスケジュール管理システムとメタ解法を組み合わせるシステムの計算実験

10.5.1 遺伝的アルゴリズムを用いるシステムの計算実験

パラメータ

実験より，パラメータを以下のように設定する．

- 集団サイズは，個体数20.
- 交叉率，突然変異率ともに100パーセント.

ここでは，8.3で提案した遺伝的アルゴリズムを用いるシステムの計算実験の結果を示す．

ジョブショップ問題に対する実験では，問題例の規模に関係なく，誤差にばらつきがあった（表10.10）．また，優先度順の平準化が最も誤差を小さくしていた．これは遺伝的アルゴリズムが，優先度順の平準化により算出された最大完了時刻を最小とするのを目的としているためである．

表 10.9: RCSPに対する最大値を用いるシステムの実験結果

ファイル名	最良値	標準		優先度順		ID順	
		値	誤差	値	誤差	値	誤差
j301_1	43	53	23.26	53	23.26	53	23.26
j601_1	77	102	32.47	102	32.47	102	32.47
j901_1	73	99	35.62	99	35.62	107	46.58
j1201_1	107	151	41.12	151	41.12	150	40.19

RCSPに対する実験では，すべての問題例に対し誤差が少なかった（表10.11）．また，優先度順の平準化が最も誤差を小さくしていた．

ジョブショップ問題での誤差よりもRCSPでの誤差の方が小さかった．これは，遺伝的アルゴリズムを用いるシステムが計画期間の短い問題例の方に対し有効であることを示している．

表 10.10: ジョブショップ問題に対する遺伝的アルゴリズムを用いるシステムの実験結果

ファイル名	最適値	標準		優先度順		ID順	
		値	誤差	値	誤差	値	誤差
ft06	55	56	1.82	56	1.82	73	32.73
ft10	930	1099	18.17	1067	14.73	1283	37.96
ft20	1165	1559	33.82	1267	8.76	1461	25.41
la01	666	701	5.26	666	0.00	1040	56.16
orb01	1059	1337	26.25	1271	20.02	1326	25.21

表 10.11: RCSPに対する遺伝的アルゴリズムを用いるシステムの実験結果

ファイル名	最良値	標準		優先度順		ID順	
		値	誤差	値	誤差	値	誤差
j301_1	43	43	0.00	43	0.00	49	13.95
j601_1	77	77	0.00	77	0.00	82	6.49
j901_1	73	81	10.96	81	10.96	91	24.66
j1201_1	107	132	23.36	124	15.89	129	20.56

10.5.2 局所探索法を用いるシステムの計算実験

ここでは，8.4で提案した局所探索法を用いるシステムの計算実験の結果を示す．

ジョブショップ問題に対する実験では，ft06，ft10, la01といった比較的規模の小さい問題例に対しては，標準，優先度順の平準化は誤差が小さく，ID順は誤差が大きくなった（表10.12）．逆にft20，orb01といった規模の大きな問題例に対しては，標準，優先度順，ID順の誤差が同じくらいであり，大きかった．特に規模の小さな問題例に対しては誤差が大きかったID順が，規模の大きな問題例に対しては誤差が小さくなり，他の基準よりも誤差が小さくなった．これは，平準化のみによる実験結果と同様である．

RCSPに対する実験では，実装の都合上行えなかった．

以上のことから平準化のみで最大完了時刻を算出すると，問題例の規模が小さいときは，標準，優先度順の平準化が誤差が小さく，問題例の規模が大きいときは，ID順の平準化が誤差が小さかった．

表 10.12: ジョブショップ問題に対する局所探索法を用いるシステムの実験結果

		標準		優先度順		ID順	
ファイル名	最適値	値	誤差	値	誤差	値	誤差
ft06	55	56	1.82	56	1.82	60	9.09
ft10	930	1099	18.17	1099	18.17	1153	23.98
ft20	1165	1518	30.30	1518	30.30	1359	16.65
la01	666	701	5.26	701	5.26	726	9.01
orb01	1059	1337	26.25	1337	26.25	1309	23.61

10.6 計算実験の結果の分析

市販のスケジュール管理ソフトとメタ解法を組み合わせたシステムでは，遺伝的アルゴリズムを用いるシステムと局所探索法を用いるシステムを提案した．

2つのシステムともに，平準化のみのときの最大完了時刻を改善した．

計算実験の結果を表10.13，表10.14にまとめた．

表中の，拡張モデルは拡張モデルを用いるシステム，メタ解法は市販のスケジュール管理システムとメタ解法を組み合わせるシステムによる実験結果を表す．また，平準化のみは平準化のみを行ったとき，α-Pointはα-Point法を用いるシステム，最大値は最大値を用いるシステムによる実験結果を表す．

表中の数値は，各システムにおいて，標準，優先度順，ID順それぞれによる平準化をした際に，最大完了時間が最も小さかったときの誤差である．

ジョブショップ問題に対する実験結果をみてみると，問題例の規模が大きくなると誤差も大きくなることが分かる（表10.13）．遺伝的アルゴリズムは，すべての問題例に対し，平準化のみ行った場合より誤差を少なくした．局所探索法も同様に，すべての問題例に対し，平準化のみ行った場合より誤差を少なくした．最も誤差を小さくしたのは，すべての問題例に対し，遺伝的アルゴリズムであった．

RCSPに対する実験結果をみてみると，問題例の規模が大きくなると誤差も大きくなることが分かる（表10.14）．RCSPの問題例はすべて同じ構造のであるため，その変化は顕著に現れている．最大値を用いるシステムが最も誤差を大きくした．最も誤差を小さくしたのは，すべての問題例に対し，遺伝的アルゴリズムであった．

以上の結果から，本研究で提案したシステムの中で最も誤差を小さくするのは遺伝的アルゴリズムであることがわかった．

表 10.13: ジョブショップ問題に対する実験結果

		拡張モデル		メタ解法	
ファイル名	平準化のみ	α-Point	最大値	GA	LS
ft06	1.82	-	-	1.82	1.82
ft10	18.17	-	-	14.73	18.17
ft20	25.41	-	-	8.76	16.65
la01	5.25	-	-	0.00	5.26
orb01	25.21	-	-	20.02	23.61

表 10.14: RCSPに対する実験結果

		拡張モデル		メタ解法	
ファイル名	平準化のみ	α-Point	最大値	GA	LS
j301_1	0.00	0.00	23.26	0.00	-
j601_1	0.00	5.19	32.47	0.00	-
j901_1	10.96	9.59	35.62	10.96	-
j1201_1	20.56	18.69	40.19	15.89	-

第 11 章

結論と今後の課題

本章では，結論と今後の課題を述べる．

11.1 結論

本研究では，汎用性の高いスケジュール最適化システムとして，拡張モデルを用いるシステムと，市販のスケジュール管理システムとメタ解法を組み合わせたシステムを提案した．

拡張モデルを用いたシステムでは，様々な条件を考慮した場合の定式化を示し，市販のスケジュール管理システムとメタ解法を組み合わせたシステムでは，市販のスケジュール管理システムとしてProject98，メタ解法として遺伝的アルゴリズムとメタ解法を例としたシステムを示した．

提案したシステムに対し，ベンチマーク問題を用いて計算実験を行った結果，汎用のシステムとしては良い解を得ることができた．

11.2 今後の課題

提案したシステムは，現実問題へ適用可能であることを目指したものであるが，本研究ではインターネット上で配られているベンチマーク問題を用いた計算実験しか行っていない．今後の課題としては，現実問題に適用し，実用性を検討することである．

謝辞

今回，本論文を作成するにあたり，様々なご指導をいただきました東京商船大学流通管理工学講座の久保幹雄先生，苦瀬博仁先生に深くお礼申し上げます．また，京都大学大学院工学研究科の野々部宏司氏には，資源制約付きスケジューリング問題に関する貴重な資料をいただきました．流通システム研究室の学生の方々には，有益な示唆をいただきました．ここに厚くお礼申し上げます．

参考文献

[1] 園川 孝夫, 伊藤 謙治, "生産マネジメントの手法," 朝倉書店, (1996).

[2] D.E. Goldberg, "A Genetic Algorithm in Search, Optimization, and Machine Learning," Addison-Wesley, (1989).

[3] G. Syswerda, "Uniform Crossover in Genetic Algorithms," *Proc. of ICGA-89*, (1989).

[4] W. Spears and K. DeJong, "An Analysis of Multi-point Crossover," G. Rawlins, (Ed.), *Foundations of Genetic Algorithms*," Morgan Kaufmann, (1991).

[5] Z. Michalewicz, "Genetic Algorithm + Data Structure = Evolution Programs," Springer-Verlag, (1992).

[6] 野々部 宏司, 茨木 俊秀, "資源制約付きスケジューリング問題の定式化と近似解法," 生産スケジューリング・シンポジウム'98講演論文集, (1998).

[7] C.C.B. Cavalcante, C.Carvalho de Sousa, M.W.P. Savelsbergh, Y. Wang and L.A. Wolsey, "Scheduling Projects with Labour Constraints," Report LEC-98-08, Georgia Institute of Technology, (1998).

[8] T. Pyron, "Microsoft Project98徹底解説," ソフトバンク株式会社, (1998).

[9] 北野 宏明, "遺伝的アルゴリズム," 産業図書株式会社, (1993).

[10] R. Fourer, D.M. Gay, B.W. Kernighan, "AMPL," Wadsworth Publishing Company, (1997).

[11] M. Mori and C.C. Tseng, "A genetic algorithm for multi-mode resource constrained project scheduling problem," *European Journal of Operational Research*, 100 (1997).

[12] P. Brucker, A. Drexl, R. Mohring, K. Neumann, and E. Pesch, "Resource-constrained project scheduling: Notation, classification, model, and methods," *European Journal of Operational Research*, 112 (1999).

[13] H. Fisher and G.L. Thompson, "Probabilistic learning combinations of local job-shop scheduling rules," in: J.F. Muth and G.L. Thompson (eds.), *Industrial Scheduling*, Prentice-Hall, Englewood Cliffs, NJ.

[14] R. Kolisch and K. Hempel, "Finite Scheduling Capabilities of Commercial Project Management Systems," *Manuskripte aus den Instituten fuer Betriebswirtschaftslehre*, 397 (1996).

[15] S. Hartmann, "Project Scheduling with Multiple Modes: A Genetic Algorithm," *Manuskripte aus den Instituten fuer Betriebswirtschaftslehre*, 435 (1997).

付録A AMPLによる資源制約付きスケジューリング問題の基本形の記述

付録 A

AMPLによる資源制約付きスケジューリング問題の基本形の記述

```
# Resource Constrained Scheduling Problem (Sousa's Precedence Constraint)

# Consideration to
#   -Resource Excess Cost
#   -Dependent Relationship
#   -Job Constraint
#   -Appointed Day

### Sets and Parameters ###

set Job;
set Period;
set Resource;

set PrecFS 'Finish-Start' within {Job, Job};
set PrecSS 'Start-Start'  within {Job, Job};
set PrecFF 'Finish-Finish'within {Job, Job};
set PrecSF 'Start-Finish' within {Job, Job};

set RJ 'Resources and Jobs' within {Job, Resource};

set FixedJobDatePair within {Job, Period};

param T 'term';
```

```
param SetUpFS 'setup time FS' {PrecFS};
param SetUpSS 'setup time SS' {PrecSS};
param SetUpFF 'setup time FF' {PrecFF};
param SetUpSF 'setup time SF' {PrecSF};

param P 'processing time' {Job};
param C 'cost' {Job, Period};

param JBegin 'release time' {Job};
param JFinish 'time of delivery' {Job};

param R 'quantity of using resource' {RJ};
param RUB 'upper bound of resource' {Resource, Period};

param OTC 'resource excess penalty' {Resource};

set S1
  := setof {j in Job, t in Period:
  (t <= JFinish[j] - P[j] + 1) and (t >= JBegin[j])}
  (j, t);

set S2 {j in Job}
  := setof {t in Period:
    (t <= JFinish[j] - P[j] +1) and
    (t >= JBegin[j])}
    (t);

set S3 {t in Period}
  := setof {j in Job, s in Period:
    (s <= t) and
    (s <= JFinish[j] - P[j] + 1) and
    (s >= t - P[j] + 1) and (s >= JBegin[j])}
    (j,s);

set S4 {(j,k) in PrecFS, t in Period}
  := setof {s in Period:
    (s >= t + P[j] + SetUpFS[j,k]) and
    (s <= JFinish[k] - P[k] + 1) and
    (s >= JBegin[k])}
```

```
    (s);

set S5 {j in Job}
  := setof {t in Period:
    (t <= JFinish[j] - P[j] + 1) and
    (t >= 2) and
    (t >= JBegin[j])}
    (t);

set S6
  := setof {(j,k) in PrecSS, t in Period, s in Period:
    (t <= JFinish[j] - P[j] + 1) and
    (s <= JFinish[k] - P[k] - SetUpSS[j,k] + 1) and
    (t >= JBegin[j]) and
    (s >= JBegin[k]) and
    (s == t + SetUpSS[j,k])}
    (j,t,k,s);

set S7
  := setof {(j,k) in PrecFF, t in Period, s in Period:
    (t <= JFinish[j] - P[j] + 1) and
    (s <= JFinish[k] - P[k] + 1) and
    (t >= JBegin[j]) and
    (s >= JBegin[k]) and
    (t == s + P[k] - P[j] - SetUpFF[j,k])}
    (j,t,k,s);

set S8 {k in Job, t in Period}
  := setof {s in Period:
    (s >= t - P[k] + 1) and
    (s <= JFinish[k] - P[k] + 1) and
    (s >= JBegin[k])}
    (s);

set S9
  := setof {(j,k) in PrecSF, t in Period:
    (t <= JFinish[j] - P[j] + 1) and
    (t >= P[j] + 1) and
    (t >= JBegin[j])}
    (j,k,t);
```

```
### Variables ###

#var x {Job, Period} >= 0;
var x {Job, Period} binary;
var y {Resource, Period} >= 0;
var MakeSpan >=0;

### Objective ###

minimize total_cost:
MakeSpan + sum{r in Resource, t in Period} OTC[r] * y[r,t];

#  sum{(j,t) in S1} C[j,t] * x[j,t]
#   + sum{r in Resource, t in Period} OTC[r] * y[r,t];

### Constraints ###

subject to bin '0-1 constraint' {j in Job, t in Period}:
  x[j,t] <= 1;

subject to JobSat 'Job Satisfaction' {j in Job}:
  sum {t in S2[j]}x[j,t] == 1;

subject to ResConst 'Resource Constraint' {r in Resource, t in Period}:
  sum {(j,s) in S3[t]:(j,r) in RJ} R[j,r] * x[j,s]
   <= RUB[r,t] + y[r,t];

#subject to FSStd 'Standerd FS' {(j,k) in PrecFS, t in Period}:
#  x[j,t] - sum {s in S4[j,k,t]} x[k,s] <= 0;

subject to SousaFS {(j,k) in PrecFS}:
  sum {t in S5[j]} (t - 1) * x[j,t] + P[j] + SetUpFS[j,k]
   <= sum {s in S5[k]} (s -1) * x[k,s];

#subject to SS {(j,t,k,s) in S6}:
#  x[j,t] == x[k,s];

subject to SousaSS {(j,k) in PrecSS}:
  sum {t in S5[j]} (t - 1) * x[j,t] + SetUpSS[j,k]
```

```
   == sum{s in S5[k]} (s - 1) * x[k,s];

#subject to FF {(j,t,k,s) in S7}:
#  x[j,t] == x[k,s];

subject to SousaFF {(j,k) in PrecFF}:
  sum {t in S5[j]} (t - 1) * x[j,t] + P[j] + SetUpFF[j,k]
   == sum {s in S5[k]} (s - 1) * x[k,s] + P[k];

#subject to SF {(j,k,t) in S9}:
#  x[j,t] + SetUpSF[j,k] <= sum {s in S8[k,t]} x[k,s];

subject to SousaSF {(j,k) in PrecSF}:
  sum {t in S5[j]} (t - 1) * x[j,t] + SetUpSF[j,k]
   <= sum {s in S5[k]} (s - 1) * x[k,s] + P[k] - 1;

subject to AppDay 'Appointed Day' {(j,t) in FixedJobDatePair}:
  x[j,t] == 1;

subject to MakeSpanConstraint {j in Job}:
 MakeSpan >= sum {t in S5[j]} (t - 1) * x[j,t] + P[j];
```