遺伝的機械学習に基づくスケジューリング・ルール獲得方法に関する研究

榊原, 一紀

(Degree)
博士（工学）

(Date of Degree)
2004-03-31

(Date of Publication)
2009-11-27

(Resource Type)
doctoral thesis

(Report Number)
甲3076

(URL)
https://hdl.handle.net/20.500.14094/D1003076

博 士 論 文

遺伝的機械学習に基づく
スケジューリング・ルール獲得方法に関する研究

平成16年 1月

神戸大学大学院自然科学研究科

榊 原　一 紀

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，スケジューリングは生産システムのみならず，計算機タスク，鉄道ダイヤなどの計画・運用における重要な問題として注目されており，理論および実用的な観点から多くの報告がなされている[1, 2]．特に生産スケジューリングにおいて，理論的な研究の多くは，単純化された問題を取り扱ったものがほとんどであり，様々な制約が複雑に絡み合っているような実際の問題に対応することは難しい．さらには運用時において，事前に予測・考慮されていないことが起こる場合や，問題の構成要素があらかじめ確定的に与えられない場合も少なくない．これらの現実的な事象を考慮したスケジューリングに対しては，ヒューリスティック・ルールを基にした研究が多く[3]，「どのルールがどの問題に対して有効か」といったことに焦点が当てられ，数理的なアプローチに基づく研究はほとんどない．

このような点に留意して本研究では，対象とする問題のモデルを記述し，問題を明確とした上で，ダイナミックにスケジュールを作成する方法の一つとして，ルールを用いたスケジューリングを取り上げる．ルールを用いたスケジューリングは，一般に，仕事割り付けと同時に意思決定するリアルタイム方式との関連が深い．リアルタイム方式の構成に際しては，従来，ルール・ベースの解法アプローチを前提として，あらかじめ用意されたルール群から問題/状況に応じて適切なルールを選択するというアプローチが一般的である．例えば，動的なジョブショップ環境下における各種ルールの有効性をシミュレーションにより検討した研究や[4, 5]，if-then 形式で表現されたルールから構成されるエキスパート・システムを帰納学習を用いて構築する研究[6, 7] などが挙げられる．これらのアプローチにおいては，既存のルールを用いることを前提にしており，一般に，解法構成にあたり対象問題に対する高度な知識が必要とされる．

これらの研究を踏まえて本論文では，既存のルールを用いることを前提とはせずに，ルールそのものを獲得することを目的とする．スケジューリング問題では，通常，一定期間のスケジューリングの結果に対して事後的に評価が与えられるといった場合が多い．これを踏まえて，シミュレーションを通して事前 (オフライン) にルールを獲得することを考慮する．ルール獲得を静的な枠組みで捉えることにより，静的問題を前提としたような各種手法の適

用が可能となる．

一般に，スケジューリング問題は組合せ最適化問題として分類され，組合せ最適化問題に対しては大別して以下の 3 つのアプローチが取られる:

(a) 列挙的アプローチ,

(b) 発見的アプローチ,

(c) 探索的アプローチ.

(a) は，問題に関する厳密なモデルが必要であり，また問題の規模が大きくなると，一般に膨大な計算量を必要とする．また (b) は，解法設計にあたり問題に対する高度な知識を必要とされる．これらに対して (c) は，許容時間内に少ない計算量で良質な (準最適な) 解を得るアプローチであり，近年，スケジューリング問題をはじめとして様々な問題に対する効果的な解法が提案されている [8]．本論文では，(c) の中で代表的方法である遺伝アルゴリズム (Genetic Algorithms: GA) [9] に注目し，GA を用いてルールを獲得する方法である遺伝的機械学習 (Genetics-Based Machine Learning: GBML) [10] を用いた解法構成を試みる．GBML は大きく 2 つのアプローチ: オンライン学習型の Michigan アプローチとオフライン学習型の Pitt アプローチに分類されるが，本論文では上記の理由により，Pitt アプローチの適用を考慮する．

GBML を用いるにあたっては，ルールを適当な記号列として表現することが望ましい．そこで，GBML によるルール学習を前提として，スケジューリング状況に依存して意思決定を与えるようなルールの構成，および，このルール構成を用いた場合の GA の適用方法を提案する．GBML によるルール学習については，従来，生産システムの状態に基づいて割り付け仕事を選択するような if-then 形式のルールを考慮し，このルールを自動的に調整・獲得するために，後件部 (then 部) のみを GA を用いて決定する計算モデルが提案されている [11]．このモデルは，(離散) 状態の各々に対していずれか一つのルールを対応させておくものであり，規模の大きな問題あるいは本論文で対象とするような複雑な構造を持つ問題には適用が難しい．本論文では，ルール集合が状態空間の一部を保持するものとして，前件部と後件部の双方を GA を用いて決定するような計算モデルを提案する．これにより，大規模・複雑な問題に対しても適切なルールが獲得されることが期待される．

以上で示したアプローチについて，本論文では，次に示す 2 つの問題に対して具体的なルール構成および GBML の構成を提案していくことにより，GBML に基づくアプローチの有効性を確認する．まず， 完成した製品を逐次在庫設備に振り分ける問題 (在庫設備選択問題) を取り上げる．この問題では，在庫設備の入出庫に対してコストが発生し，適当な時間間隔で発生する製品に対して，製品の属性 (重量，保管時間など) や在庫設備の空き状況に応じて入庫する在庫設備を決定する．このとき，計画期間における総コストを最小とすることが目的となる．

次に，古典的なスケジューリング問題の中で最も広範な応用の可能性を有するフレキシブ

ルショップ問題を取り上げる．この問題は，ジョブショップ問題において，各ショップに複数の機械が並列に存在するような問題であり，本研究ではさらに現実的な応用の観点から，段取り替えに関する制約も考慮する．在庫設備選択問題と比較してフレキシブルショップ問題は，仕事を割り付ける機械の選択に加えて，仕事の処理順序という時間軸上に広がりを持つような，ルール・ベースの枠組みでの解法構成が困難であるような特徴を有する問題となっている．

1.2 論文の構成

以下，本論文の構成について述べる．

第 2 章では，本論文が対象とするルール・ベースのスケジューリングについて，従来研究を整理するとともに，その一般的枠組み・方法論を示す．その際，ルール・ベースのスケジューリングが，特に仕事割り付けと同時に (ダイナミックに) 意思決定するリアルタイム方式との関連が深い点に注目し，ダイナミックにスケジュールを作成する方式，すなわちダイナミック・スケジューリングの範疇でその位置付けを明確にする．

第 3 章では，第 2 章で整理したスケジューリング・ルールを遺伝的機械学習に基づいて生成・調整するための計算モデルを示す．その際，シミュレーションをベースとして，事前 (オフライン的) にルール集合を獲得する Pitt アプローチの構成を考慮する．以降の章で用いられる枠組みとして，ルールをスケジューリング状況に応じて意思決定するためのパラメータから構成し，パラメータ空間で最適あるいはロバストなルールを探索する方式を示す．

第 4 章では，在庫設備選択問題に GBML を用いたルール獲得法の適用方法を提案する．その際，ルール構成として，前件部に入庫する製品の属性および在庫設備の空き状況に関する条件を記述し，後件部に決定事項 (製品を割り付ける在庫設備番号) を直接記述したものを考慮する．最後に，計算例を通して，提案手法の有効性を確かめる．

第 5 章では，フレキシブルショップ問題に対するルール獲得法を構成することを踏まえて，まず問題の混合整数計画問題への定式化を示し，問題を明確に記述しておく．さらに，フレキシブルショップ問題に対するルール獲得法を構成する (第 6 章) ことを踏まえて，2 種類の決定 (仕事の機械割り付け (決定事項 A と呼ぶ) と機械上の作業の処理順序 (決定事項 B と呼ぶ)) を定める方法として，(i) 決定事項 A と B を独立に定める方法，(ii) 決定事項 A を定めた上で B を定める階層的方法，および (iii) (ii) とは逆に，決定事項 B を定めた上で A を定める方法，の 3 つについて比較・検討する．

第 6 章では，フレキシブルショップ問題に対する GBML を用いたルール獲得法を提案する．Pitt アプローチによるルール獲得時には，ルール集合の評価のためのシミュレーションが必要となるが，第 5 章での検討を踏まえて，シミュレータを構築することを考慮する．すなわち，機械割り付けをヒューリスティックに決定した上で，機械上の作業の処理順序を

ルールを用いて決定する．またルール構成として，決定の際に用いられる問題の属性値の重みを後件部とするものを提案する．これによって，様々な状況に対応したルールを実現することができると考えられる．最後に，計算例を通して，提案手法の有効性を確かめる．

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

ルールに基づくスケジューリング

2.1 緒言

本章では，本論文が対象とするルールに基づくスケジューリングについて，従来研究を整理するとともに，その一般的枠組み・方法論を示す．その際，ルールに基づくのスケジューリングが特に，仕事割り付けと同時に (ダイナミックに) 意思決定するリアルタイム方式との関連が深い点に注目し，ダイナミックにスケジュールを作成する方式 (ダイナミック・スケジューリング) の範疇で，その位置付け (どのように用いられるかといった点) を明確にする．

そこでまず，スケジューリング問題の概要およびアプローチについて述べ，さらに (本論文で対象とする) 生産システムにおけるスケジューリング，すなわち生産スケジューリングについて，生産計画との関わりの観点から説明する．次に，ダイナミック・スケジューリングの方式について，意思決定のタイミングの観点から 3 つのアプローチ: 事前準備方式，リアクティブ方式およびリアルタイム方式に分類する．その際，リアルタイム方式において代表的アプローチであるルールを用いたスケジューリングに着目し，既存研究を整理するとともに，その枠組みを明確に記述する．

2.2 スケジューリング問題

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 [12, 13, 14]．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．

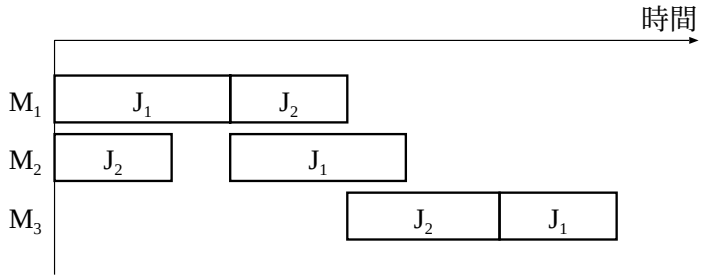


図 2.1: ガント・チャート
J は仕事，M は機械を表す

スケジューリングの目的は，上記の基本条件を含む制約条件を満たす実行可能なスケジュールを見いだすこと，あるいは，もし複数の実行可能スケジュールが存在するなら，それらの中から与えられた評価基準 (最大完了時間や納期遅れ和など) を最適にするスケジュールを見いだすことである．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，縦軸に機械，横軸に時間をとってガント・チャート (Gantt chart) で表されることが多い (図 2.1).

従来，スケジューリングに関する研究は，オペレーションズ・リサーチ，組合せ最適化および人工知能の分野において大きく発展した．それらの研究の多くは，解法を構成するにあたり，概ね下記の 4 つのアプローチに基づいている:

(a) 数理計画によるアプローチ,

(b) (ヒューリスティクス) ルールによるアプローチ,

(c) シミュレーションによるアプローチ,

(d) メタ・ヒューリスティクスによるアプローチ.

(a) は問題の厳密なモデルを構成することに主眼をおくアプローチであり，問題に関する知識 (構成要素，決定事項，制約条件，評価基準など) があらかじめ十全に与えられている必要がある．またスケジュール獲得にあたり，得られるスケジュールの良さ (最適性) を評価できる一方で，(問題の規模が大きくなるにつれて) 一般に膨大な計算量が必要となる．それに対して (b) および (c) では，どのようにすれば良いスケジュールが得られるかという (ある意味で高度な) 手続きが具体化されている一方で，問題が明確に記述されておらず (問題の記述と解法構成を明確に分離できない)，どのようなスケジュールが実行可能で，またどのようなスケジュールが良いのかという点がはっきりしていない場合も多い．このとき，得られるスケジュールの良さ (最適性など) を定量的に評価する，さらに解法を改善する，あるいは他の手法を試験的に用いてみるということが困難となってくる．(d) は主に，組合せ最適化問題において発展した探索に基づくアプローチであり，(a) ∼ (c) をベースとして，あるいは組み合わせて用いられる．得られる解 (スケジュール) の良さと求解に要する時間をバランスさせるという観点から，(d) のアプローチがとられることが多い.

生産活動におけるスケジューリング (生産スケジューリング) についてみてみると [15, 16]，生産活動の計画が長期計画，中期計画，短期計画などに階層化され，上層から下層に進むに従って計画対象期間と生産設備などの物的な計画対象を限定しながらより確度の高い情報に基づいていっそう具体的な計画が立てられる中で，生産スケジューリングは短期計画に対応する．つまり，通常，生産スケジューリングは製造活動の最も具体的な計画の作成を意味し，上位レベルにおいて決定された情報 (仕事の種類，数量，製造制約，納期，スケジューリングの評価尺度など) が仕事や行程に関するパラメータや制約条件および目的関数の形で表現され，これらに基づいてどの仕事をいつ，どの機械を使って処理するかを決定する問題と捉えられる．以降，生産スケジューリングを中心に議論を進めていく．

2.3 ダイナミック・スケジューリング

従来のスケジューリング問題に対する理論的な研究は，問題の属性 (仕事属性，機械属性，制約条件，評価基準など) を既知と仮定した静的なスケジューリング問題を対象としているものが多い．一方，実際のスケジューリングにおいては，特に生産スケジューリングについてみると，上位の生産計画の変更に伴う構成要素の変動や，生産活動における基本要素 (仕事属性，機械属性) の不確実性 (以下，不確定要因) が存在する．例えば，生産実施段階においては，

(上位の生産活動の変更に伴う)

(a) 新規仕事の追加，

(b) 既存の仕事のキャンセル，

(c) 仕事の開始時刻/納期の変更，

(生産活動に伴う)

(d) 機械故障，

(e) 処理時間の変動

などが発生し得る．

近年の生産システムの自動化，さらには変種変量生産に伴う生産システムの複雑化により，生産実施段階における不確定要因への対応を積極的に考慮したスケジューリングが重要視されている．このようなスケジューリングの一つとして，生産実施時にスケジュールを作成する，あるいは修正するといった (ダイナミックにスケジューリングする) アプローチ (ダイナミック・スケジューリングと呼ぶ) が考えられる [2, 17]．スケジュールの立案時期，すなわちスケジューリングにおける意思決定のタイミングに注目した場合，スケジューリングは，以下の 3 つの方式を取ることが可能である [15, 18]：

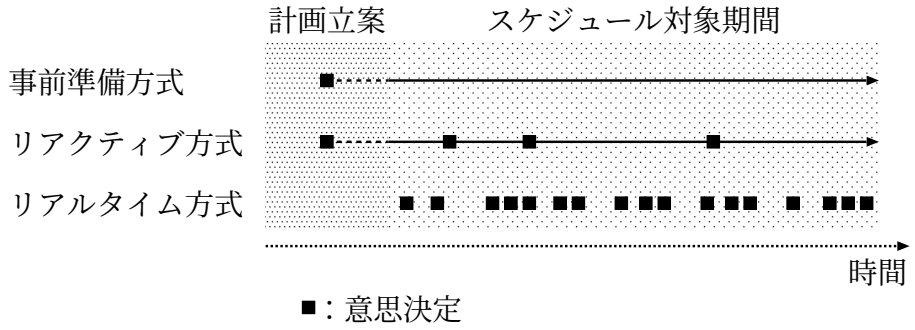


図 2.2: ダイナミック・スケジューリングの 3 つのアプローチ

(1) 事前準備方式:
事前に (計画立案段階において) スケジュールを作成する静的なアプローチ.

(2) リアクティブ方式:
スケジュール運用時 (生産実施時) において，事前に作成したスケジュールでは対応が困難な事象が発生した場合 (機械故障や新規仕事の追加など) や，自然発生的にスケジュールが遅れた場合などに対応するために，再スケジューリング/スケジュール修正を適宜繰り返すアプローチ.

(3) リアルタイム方式:
事前にスケジュールを作成することなく，到着した仕事を随時機械に割り付ける (ディスパッチングする) アプローチ.

図 2.2 にこれらのアプローチにおける意思決定のタイミングを比較・整理しておく.

(1) の事前準備方式では，計画立案段階においてスケジュールを作成し，スケジュール運用時には可能な限りスケジュールの修正を行わない，という立場を取る．従来の静的スケジューリングはこの方式に含まれる．不確実性を陽に考慮した研究においては，操業変動 (トラブル) 発生の前後のスケジュールのずれ (順序の変更の差異や完了時間のずれ) が最小となるようなスケジュールを獲得することに主眼が置かれている．仕事の処理順序を遵守することに重点を置き，スケジュールの頑強さ (ロバスト性) の指標を評価関数に組み込んだものを用いてスケジュールを求める方法 [19] や，意図的に遊休時間をスケジュールに挿入することによって，機械故障などによるスケジュールの遅れを吸収するための冗長性を積極的に取り得れるといったアプローチ [20] が代表的である.

(2) のリアクティブ方式では，計画段階で作成したスケジュールに対し，特急仕事やショップ・フロアの進捗状況に応じて修正・変更を加えるなどのアプローチを取る．リアクティブ方式に関する研究の多くは，スケジュールの反復的な改善操作に着目するものであり，ヒューリスティクスや知識ベースに基づくものがほとんどである．改善操作そのものは，時間的・手続き的な制約から，必然的に仕事の交換や仕事の移動といった比較的シンプルな操作に限定されることになる．Bierwirth ら [21] は，遺伝アルゴリズム (Genetic Algorithms: GA) [9]

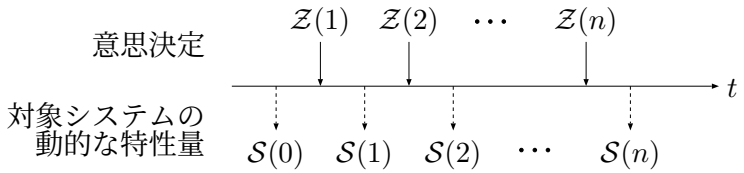


図 2.3: 意思決定のタイミング

を利用したスケジュール修正法を提案している．そこでは，仕事の到着時刻が確定的ではない場合を想定し，リアルタイムに仕事が到着した場合のスケジュールの改善案の生成に注目している．すなわち，ある状況におけるスナップショットに基づいて GA を適用することにより，スケジュールの修正を行うというものである．一方，知識ベース技術を利用したリアクティブ方式の研究として，スケジューリング専門家のスケジュール修正知識を事例ベースに蓄え，生産状況に柔軟に対応するスケジューリング・システム (CABINS) [22] などがある．

(3) のリアルタイム方式については，仕事が到着する，もしくは機械が利用可能になるときなどのタイミングで仕事を割付けるといったアプローチが一般的であり，その際，意思決定がシンプルな形式で記述されたルールを用いてスケジュールが作成されることが多い．次節では，ルールがリアルタイム方式においてどのように用いられるか，またどのようなルールが用いられるかという点について，既存研究を整理する．さらにそれを踏まえて，本論文で提案するルール構成について述べる．

2.4 ルールの構成・適用

2.4.1 ルールの適用方法

2.3 で述べたように，リアルタイム方式は，事前にスケジュールを用意することなく，生産実施時に到着した仕事を随時ディスパッチングするアプローチである．リアルタイム方式は，実際の生産スケジューリングにおいて幅広く活用されているアプローチであり，統一的な枠組みを与えることは困難であるが，概ね次に以下に示すように形式化される．

まず前提として，

- 計画期間 $[0, T]$,
- 時刻 t_k ($t_k < T$; $k = 1, \cdots, n$) に仕事をディスパッチング (例えば，機械が空く最早時刻など),

を考慮するものとする．このとき，時刻 t_k における決定事項を $\mathcal{Z}(k)$ とし，$\mathcal{Z}(k)$ に基づいて観測される対象システムの動的な特性量を $\mathcal{S}(k)$ (初期の特性量を $\mathcal{S}(0)$ とする;　例えば，仕事の完成した割合や最大完了時刻など) とすると，問題は，各時刻 t_k において $\mathcal{S}(\ell)$ ($\ell = 0, \cdots, k-1$) に基づき $\mathcal{Z}(k)$ を決定することであるといえる．図 2.3 に対象システムか

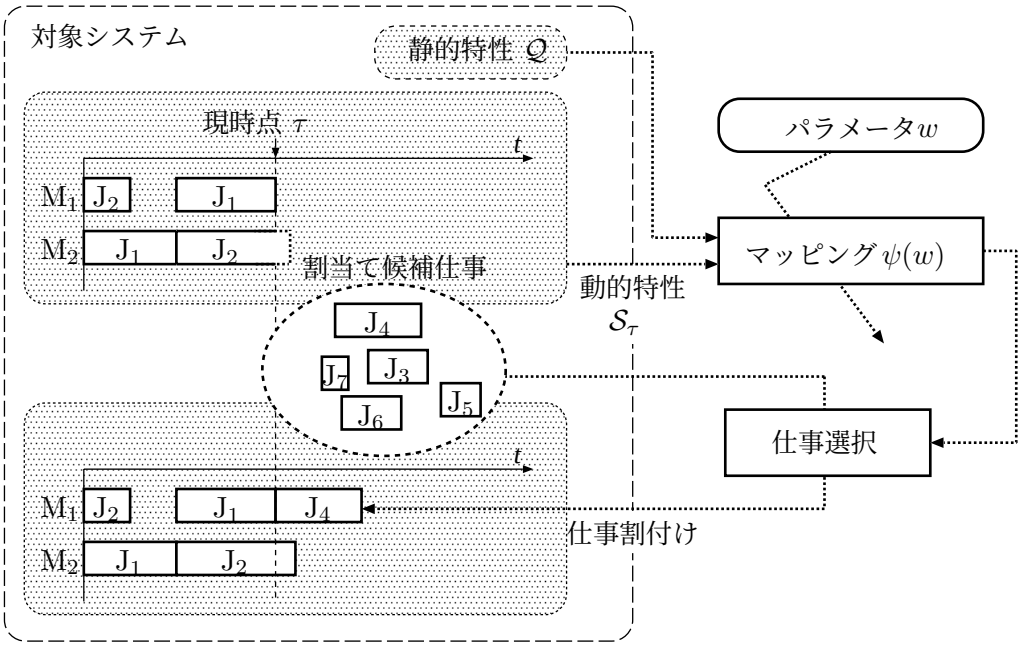


図 2.4: ルール適用の枠組み

ら $\mathcal{S}$ が観測されるタイミングと，$\mathcal{Z}$ を与えるタイミングを示す．このとき $\mathcal{S}$ に基づく $\mathcal{Z}$ の決定が，マッピング:

$$\psi : \hat{\mathcal{S}} \times \mathcal{Q} \to \mathcal{Z} \tag{2.1}$$

で表されるものであると捉えられる．ここで,

$$\hat{\mathcal{S}}(k) = \mathcal{S}(0) \times \mathcal{S}(1) \times \ \ldots \ \times \mathcal{S}(k-1)\,, \tag{2.2}$$

であり，$\mathcal{Q}$ は機械および仕事の静的な属性を表す．この ψ がパラメータ w によって定められるものとすると，リアルタイム方式の構成にあたっては,

(a) システムの特性量 $\mathcal{S}$ および $\mathcal{Q}$ をどのように与えるか？

(b) マッピング ψ をどのようなものにするか？

(c) パラメータ w をどのように設定/調整するか？

が問題となる．以上で示したリアルタイム方式の枠組みを図2.4 に示す．このとき，ルールを用いて決定する対象として次の 2 つが考えられる (図2.5):

(A) マッピング ψ そのもの,

(B) マッピング ψ を調節する重みパラメータ w.

(A)，(B) それぞれに対して，どのようなルール構成が考えられるかを，(**2.4.2** に) これまで提案されてきたルールの表現形式を整理・分類した上で (**2.4.3** に) 示す.

2.4.2　ルールの構成

2.3 で示したように，リアルタイム方式における意思決定プロセスでは，動的環境の時間的・手続き的制約により，スケジューリング手順に迅速性および単純性が強く要求される [23]．そこで，意思決定の際にはスケジューリング操作を記述したルールが用いられることが多い．このときルールは，対象システムの状況 ($\hat{\mathcal{S}}$) から意思決定 $\mathcal{Z}$ へのマッピング ψ を与えるための「仕掛け」であると考えられる．このようなルールの表現形式に注目すると，既存のスケジューリング・ルールは，マッピング ψ の構造に応じて以下の 3 つに大別される:

(a) 静的ディスパッチング・ルール:　主に静的な仕事属性 (仕事 J_j の処理時間 p_j，納期 d_j，到着時刻 r_j など; $\mathcal{Q}$) のみで表現可能なルールであり，時間経過に伴う仕事や生産設備などの属性情報 ($\hat{\mathcal{S}}$) は考慮されない．以下，代表的な静的ディスパッチング・ルールを挙げる [1, 24, 25]:

- SPT (Shortest Processing Time):
  p_j の非減少順に仕事を処理する．
- EDD (Earliest Due Date):
  d_j の非減少順に仕事を処理する．
- FCFS (First Come First Served):
  r_j の非減少順に仕事を処理する．

一機械最大納期遅れ最小化問題において，EDD ルールにって求められるスケジュールが最適となる [1] など，いくつかの静的ディスパッチング・ルールは，静的スケジューリング問題に対する最適化手法としての局面を持つことがわかっている．

(b) 動的ディスパッチング・ルール:　静的な仕事属性 ($\mathcal{Q}$) に加え，仕事 J_j の残り作業の処理時間和や納期余裕などの経過時間に依存する仕事属性・機械属性 ($\hat{\mathcal{S}}$) を考慮したルールである．各仕事の処理優先度が時間経過とともに変化する点が特徴である．静的ディスパッチング・ルールに比べ，記述の自由度は高い反面，ルール構造が複雑化しやすいといえる．代表的な動的ディスパッチング・ルールを以下に示す [26]．

- MST (Minimum Slack Time, SLACK):
  納期余裕 $(d_j - w_j(t) - t)$ の非減少順に仕事を処理する．
- MDD (Modified Due Date):
  修正納期 $\max\{d_j, t + w_j(t)\}$ の非減少順に仕事を処理する．
- CR (Critical Ratio):
  重要度 $(d_j - t)/w_j(t)$ (納期遅れの可能性) の非増加順に仕事を処理する．

ただし，t，d_j および $w_j(t)$ は，それぞれ仕事の割り付け時刻，仕事 J_j の納期および仕事 J_j の残り作業の処理時間和を表す．Rajendran ら [25] は，ここに列挙したものを含

む多くの動的ディスパッチング・ルールについて，フローショップおよびジョブショップ型の生産システムにおける動的環境下での有効性を比較・検証している．動的ディスパッチング・ルールは，進捗状況というリアルタイムの情報を利用するために記述の自由度が高く，現在でも研究が盛んに進められている [4, 5, 27]．

(c) 状況依存型ルール:　対象システムのある時点における (生産) 進捗状況 ($\hat{\mathcal{S}}$) とそこでの問題解決策 (スケジューリング操作) を対で表現するルールであり，ダイナミック・スケジューリングでは，

前件部 : (システムの状況)　→　後件部 : (スケジューリング操作)

の表現形式をとる場合が多い．前件部を構成するシステムの状況は，設備稼働率などを定量化した数値列で表される．また，ルールの後件部となるスケジューリング操作に相当するものとして，上記の静的/動的ディスパッチング・ルールや，所与のディスパッチング・ルールの優先度が挙げられる [6, 26]．すなわち，状況依存型ルールはあるシステムの状況における静的/動的ディスパッチング・ルールの (事前) 評価を内包した「メタ・ルール」としての側面をもつものであると考えられる．

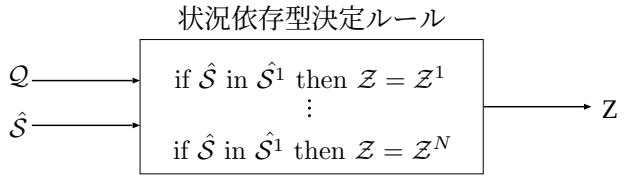
2.4.3　本論文で用いられるルール構成

2.4.2 では，既存のスケジューリング・ルールを整理・分類した．本研究では，次章で述べる遺伝的機械学習に基づくルール獲得を踏まえ，最も一般的な形式であり，かつ多様な意思決定が可能であると予想される if-then 形式のルール構成を考慮する．**2.4.1** で述べたように，対象システムから観測される特性量から決定事項へのマッピング ψ において，ルールを用いて決定する対象として (a) マッピング ψ そのもの，および (b) マッピング ψ を調節する重みパラメータ w，の二つが考えられる．

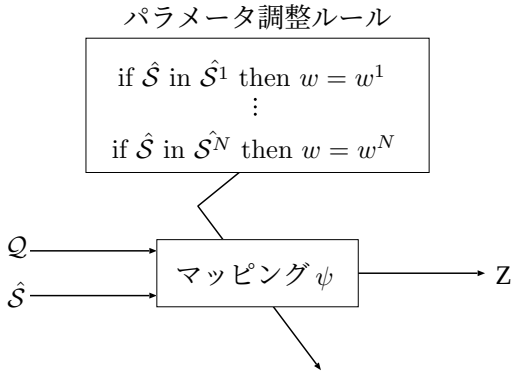
(a) の場合，ルールが

前件部 : (システムの状況)　→　後件部 : (スケジューリング操作)

といった表現形式が考えられ，既存のスケジューリング・ルール (状況依存型ルール) との関連性も大きい．なお前件部には (後述のルール獲得方法との親和性の観点から)，$\hat{\mathcal{S}}^{\mathrm{B}}$ をいくつかの部分空間 $\{\hat{\mathcal{S}}_1^{\mathrm{A}}, \ldots, \hat{\mathcal{S}}_{n^{\mathrm{S}}}^{\mathrm{A}}\}$ に分割したものを記述しておくことが好ましい．ただし，意思決定のよりどころ (ψ の入力) となる $\hat{\mathcal{S}}$ および $\mathcal{Q}$ の種類 (次元数) を多くすると，きめ細かいスケジューリングが可能となる一方で，ルールの前件部の取り得る状態数が増大し，ルールの獲得が困難となることも予想される．このときのルールの構成・適用方法を図 2.5-(1) に示す．一方 (b) の場合，(a) の場合と同様に，前件部にはシステムの状況に関する条件 ($\hat{\mathcal{S}}$) を記述するといった形式が考えられる．この際，マッピング ψ そのものとマッピング調整を分けて設計するため，意思決定の表現に関する自由度が高い．このときのルールの構成・



(1) ルール構成 (a)



(2) ルール構成 (b)

図 2.5: ルールの構成

適用方法を図2.5-(2) に示す．第 4 章では (a) のルール構成を用いた場合のルール獲得法を，第 6 章では (b) のルール構成を用いた場合のルール獲得法をそれぞれ示す．

2.5 結言

本章では，本論文が対象とするルール・ベースのスケジューリングについて，従来研究を整理するとともに，その一般的枠組み・方法論を示した．その際，ルール・ベースのスケジューリングが特に，仕事割り付けと同時に (ダイナミックに) 意思決定するリアルタイム方式との関連が深い点に注目し，ダイナミックにスケジュールを作成する方式 (ダイナミック・スケジューリング) の範疇で，その位置付け (どのように用いられるかといった点) を明確にした．それを踏まえて，スケジューリング状況に応じて意思決定 (ディスパッチング) するためのパラメータ (対象システムから観測される特性量から決定事項へのマッピングを定めるためのパラメータ) を決定するためのルール構成を示した．

第 3 章では，本章で示したルールを遺伝的機械学習を用いて獲得する方法について述べる．

第 3 章

遺伝的機械学習を用いたルール獲得法

3.1 緒言

本章では，第 2 章で整理したスケジューリング・ルールを遺伝的機械学習に基づいて生成・調整するための計算モデルを示す．その際，シミュレーションをベースとして，事前 (オフライン的) にルール集合を獲得する Pitt アプローチの構成を考慮する．第 4 章および第 6 章で用いられる枠組みとして，ルールをスケジューリング状況に応じて意思決定 (ディスパッチング) するためのパラメータ (対象システムから観測される特性量から決定事項へのマッピングを定めるためのパラメータ) から構成し，パラメータ空間で最適な (あるいはロバストな) ルールを探索する方式を示す．

そこでまず，ルールの生成・獲得に関する従来研究を示し，その上で本研究の遺伝的機械学習に基づくルール獲得法を提案し，その位置付けを明らかにする．さらに，遺伝アルゴリズムの方法を示した上で，遺伝的機械学習についてそのアプローチをオンライン学習型の Michigan アプローチとオフライン学習型の Pitt アプローチに分けて説明する．最後に両アプローチ比較し，Pitt アプローチを用いた理由について述べる．

3.2 ルールの獲得方法

第 2 章で取り上げたスケジューリング・ルールは，いずれも特定の問題 (あるいは状況) において優れたパフォーマンスを示すことが分かっているものの，広く様々な問題に対して良好なスケジュールを生成するようなルールは未だ報告されていない．ルール・ベースの解法を設計するにあたっては，問題/状況に応じて適切なルールを選択するというアプローチが一般的となっている．例えば，ある生産状況における所与のルールの適切さを事前にシミュレーションによって評価しておき，生産実施段階で状況に応じた適切なルールを選択するといった方策をとるものが多い．Shafaei ら [4, 5] は，動的なジョブショップ環境下における静的/動的ディスパッチング・ルールの有効性をシミュレーションにより検討しており，SPT と CR を組み合わせたルールが有効であることを示している．Jeong ら [26] は，16 種類のディスパッチング・ルールを組み込んだリアルタイム・スケジューリング・システムの枠組みを提

案している．このシステムでは，ディスパッチング処理の前に，比較的小規模なシミュレーションを実施し，シミュレーション結果に基づいてスケジューラが意思決定 (適用なルールの選択) を行う．

一方，知識ベースを利用したスケジューリング法として，Shaw[6] らは学習機能を付加したスケジューリング・システムを提案している．このシステムでは，人工知能分野における帰納的学習が用いられ，決定木で表現された状況依存型ルールの獲得が可能となっている．

これらの研究に対して本研究では，問題/状況に対して適切な意思決定を与えるルールそのものを獲得することを目的とし，**3.4** で述べる遺伝的機械学習を用いたルール獲得法を提案する．

(**2.4.3** で示したルール構成の) ルールを獲得するにあたっては，ルールを適当なパラメータによって記述することにより，ルール獲得問題を (パラメータ空間に対する) 組合せ最適化問題の枠組みで捉えることができる．スケジューリング問題では，通常，一定期間のスケジューリングの結果に対して事後的に評価が与えられるといった場合が多い．これを踏まえて，シミュレーションを通して事前 (オフライン) にルールを獲得することを考慮する．ルール獲得を静的な枠組みで捉えることにより，(静的問題を前提とする) 各種手法の適用が可能となる．一般に，組合せ最適化問題に対しては大別して以下の 3 つのアプローチが取られる:

(a) 列挙的アプローチ,

(b) 発見的アプローチ,

(c) 探索的アプローチ.

(a) は，問題に関する厳密なモデルが必要であり，また問題の規模が大きくなると，一般に膨大な計算量を必要とする．また (b) は，解法設計にあたり問題に対する高度な知識を必要とされる．これらに対して (c) は，許容時間内に少ない計算量で良質な (準最適な) 解を得るアプローチであり，近年，スケジューリング問題をはじめとして様々な問題に対する効果的な解法が提案されている[8]．本論文では，(c) の中で代表的方法である遺伝アルゴリズムに注目し，遺伝アルゴリズムを用いてルールを獲得する方法である遺伝的機械学習に基づく解法構成を試みる．

次章以降では，まず遺伝アルゴリズムの方法を示した上で，遺伝的機械学習についてそのアプローチをオンライン学習型の Michigan アプローチとオフライン学習型の Pitt アプローチに分けて説明する．

3.3　遺伝アルゴリズム

自然界において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．この染色体の変異と環境による選択淘汰によって種の進化が実現されている．Holland の研究[28] に始まる GA は，上述のような遺伝子情報に基づく生物の環境適応的な進化過程

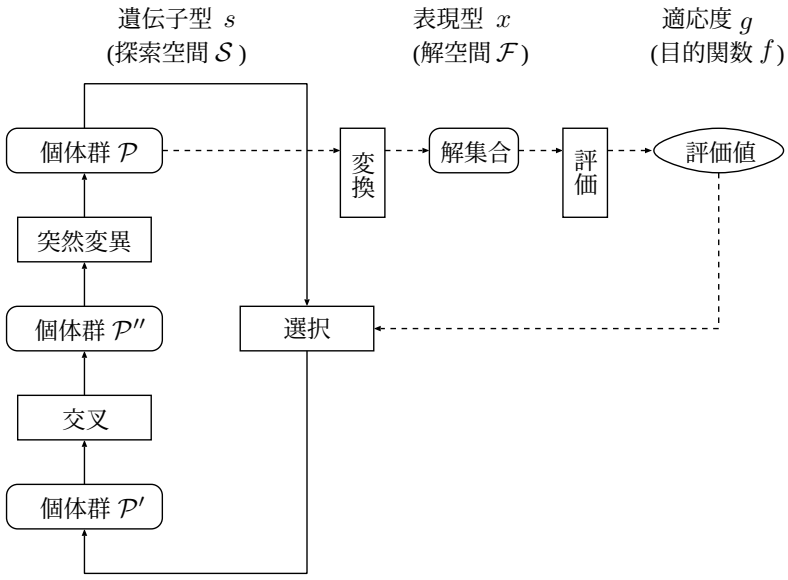


図 3.1: GA による探索の手続き

を模擬した最適化手法である．GA を最適化問題に適用する方法[29] について以下に述べる．

まず，最適化問題の決定変数ベクトル x を N 個の記号 X_i $(i = 1, \cdots, N)$ の列で表すこととし，これを N 個の遺伝子座からなる染色体とみなす．X_i は遺伝子であり，これのとり得る値が対立遺伝子である． 対立遺伝子としては，通常，0 と 1 の 2 値，ある整数の組，ある範囲の実数値，単なる記号などが用いられる．このような記号化された染色体を個体の遺伝子型と呼び，染色体に対応して定まる変数 X の値を表現型と呼ぶ．次に，進化の過程の実現方法について述べる．上述の記号化された染色体を用いて表現される N_{p} 個の個体からなる集合 (個体群) を考え，世代 t における個体群に対して，以下のような遺伝的操作 (遺伝演算子) を適用し，次の世代 $t+1$ の個体群を生成する．

(a) 選択

世代 t の個体群中の各個体 i について，その適応度 g_i に応じて，次世代に残すこの数を増減させる．代表的な選択方法として，適応度比例選択 (ルーレット選択)，期待値選択，ランキング選択などが挙げられる．

(b) 交叉

個体群中の個体をランダムにペアリングし，確率 p_{c} (以下，交叉確率) で二つの個体の遺伝子列を部分的に交換する．代表的な交叉方法として，単純交叉，複数交叉，一様交叉などが挙げられる．対立遺伝子が 0 および 1 である場合の二点交叉の例を以下に示す．

$$
\begin{array}{ccc}
1\,0\,|\,0\,1\,0\,|\,1 & & 1\,0\,|\,1\,0\,0\,|\,1 \\
\quad\updownarrow \text{交換} & \longrightarrow & \quad\updownarrow \\
1\,1\,|\,1\,0\,0\,|\,0 & & 1\,1\,|\,0\,1\,0\,|\,0
\end{array}
$$

(c) 突然変異

各個体について，確率 p_{m} (以下，突然変異率) で遺伝子座の遺伝子を他の対立遺伝子と入れ換える．対立遺伝子が 0 および 1 である場合の例を以下に示す．

$$1\,0\,0\,\boxed{1}\,0\,1 \qquad \longrightarrow \qquad 1\,0\,0\,\boxed{0}\,0\,1$$

このような演算子の適用による世代の更新を繰り返し，世代が進むにつれてより良い個体の数が増えるようにすれば，やがて最適解 x^* が得られるであろうというのが GA の基本的な考え方である．

個体群のサイズを N_{p} ，計算終了までの世代数を N_{g} とすると，GA の全体的手順は以下のようになる (図 3.1)．

1° ランダムに N_{p} 個の個体を作り，初期個体群 $\mathcal{P}(1)$ を生成する．世代を $t = 1$ とする．

2° 個体群 $\mathcal{P}(t)$ 内の各個体について，表現型への変換を行ない，目的関数値を求め，適応度を計算する．

3° $t < N_{\mathrm{g}}$ であれば 4° へ．そうでなければ 5° へ．

4° 選択・交叉・突然変異の各遺伝演算子を適用することにより，次の世代の個体群 $\mathcal{P}(t+1)$ を生成する．世代を $t = t+1$ とし 2° へ．

5° 計算終了．これまでに得られた最大適応度の個体を解とする．

さらに，GA を実現するにあたっては以下の点について具体的に定める必要がある．

(a) 個体の遺伝子表現

対象とする問題の構造・性質を十分把握した上で決定する必要がある．

(b) 適応度の計算

一般に GA では適応度の最大化を行うので，最適化問題での目的関数値が良い解ほど，それに対応する個体の適応度の値が大きくなるようにしておく必要がある．また，適応度の分布を適当に調節するスケーリングと呼ばれる操作を行う必要がある場合がある．

(c) 遺伝演算子の実現

対象とする問題の構造・性質を十分に把握した上で決定する必要がある．

(d) パラメータの設定

あらかじめ以下の四つのパラメータの値を設定しておく必要がある：

N_{p} ： 個体群のサイズ．

N_{g} ： 計算終了までの世代数．

p_{c} ： 交叉確率．

p_{m} ： 突然変異確率．

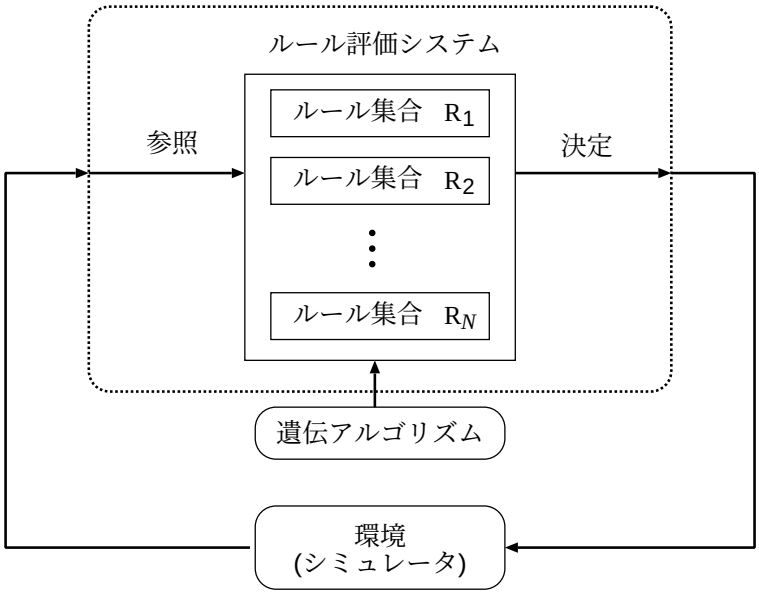


図 3.2: Pitt アプローチの概要

3.4 遺伝的機械学習

遺伝的機械学習 (Genetics-Based Machine Learning: GBML) [10, 30] は，記号列 (染色体) にコーディングされたルール集合あるいはルールを，GA を用いて自動的に選択・生成する機械学習の枠組みの一つである．GBML は二つの手法に大別される．一つは Pitt アプローチと呼ばれるもので，1980 年，Smith によって開発された LS-1 (Learning System One) [31] が起源となっている．もう一つは Michigan アプローチと呼ばれるもので，1971 年から 1978 年にかけて Holland らによって開発された CS-1(Cognitive System level One) [32] が起源となっている．本節ではまず GBML の二つのアプローチについて説明を行い，これらを比較した上で，本研究で Pitt アプローチを用いる理由について述べる．

3.4.1 Pitt アプローチ

Pitt アプローチでは，ルールを記号に，ルール集合を記号列 (個体) に対応させる．各々のルール集合を用いて得られる結果に対する評価を適応度とみなし，GA を適用して新しい記号列すなわちルール集合を生成する．

Pitt アプローチでは二つのサブシステム: (a) ルール集合，(b) GA, から成り立っている (図 3.2)．以下，それぞれについて簡単に説明する．

(a) ルール集合

ルール集合は次のように機能する．環境からディテクタを通じてメーッセージを取り込み，それに対して一つのルール集合が働いて外部に対するアクションを決定し，そ

れがイフェクタを通じて環境に出される．この一連の動作がある回数繰り返された後に，そのルール集合に対して環境から何らかの評価が与えられる．ルールは，

if 〈 condition 〉 then 〈 action 〉

という形式で記述される．ここで，〈 condition 〉を前件部，〈 action 〉を後件部と呼ぶ．

(b) GA

各ルール集合を個体，その評価を適応度と見なして GA を適用する．

ここで計算終了までの世代数を N_{rmg} とすると，Pitt アプローチによるルールの学習アルゴリズムは以下のようになる．

1° ルール集合を個体にコーディングし，初期個体群 $\mathcal{P}(1)$ を生成する．世代を $t=1$ とする．

2° 各個体 (ルール集合) をそれぞれ対象とする問題に対して適用し，その結果より適応度を計算する．

3° $t < N_{\mathrm{g}}$ であれば 4° へ．そうでなければ 5° へ．

4° 個体群に対して GA を適用し，次の世代の個体群 $\mathcal{P}(t+1)$ を生成する．世代を $t=t+1$ とし 2° へ．

5° 計算終了．これまでに得られた最大適応度の個体を目的とするルール集合とする．

3.4.2　Michigan アプローチ

Michigan アプローチによるシステムでは，ルールが記号列，ルール集合が個体群，ルールにつけられた信頼度が適応度に対応する．ルール集合を用いて得られる結果に基づいて各ルールの信頼度を変更しつつ，適当なタイミングで GA を適用して新しいルールを生成する [30]．このシステムはクラシファイア・システム (Classifier System) とも呼ばれる．Michigan アプローチは三つのサブシステム: (a) ルール/メッセージ・システム，(b) クレジット分配システム，(c) GA，から成り立っている (図 3.3)．以下それぞれについて簡単に説明する．

(a) ルール/メッセージ・システム

ルール/メッセージ・システムにおいて，環境からの情報は，まずディテクタを通して一定長のメッセージに変換され，メッセージ・リストに蓄えられる．これらのメッセージによってクラシファイア (ルール) が発火し，発火したクラシファイアはメッセージを出す．あるメッセージはクラシファイアを発火させるのでなく，イフェクタを通して環境への出力 (動作) を引き起こす．このようにルール/メッセージ・システムでは，環境からの入力とシステムの内部状態に基づいて次の出力が決定されることになる．メッセージは一般に，

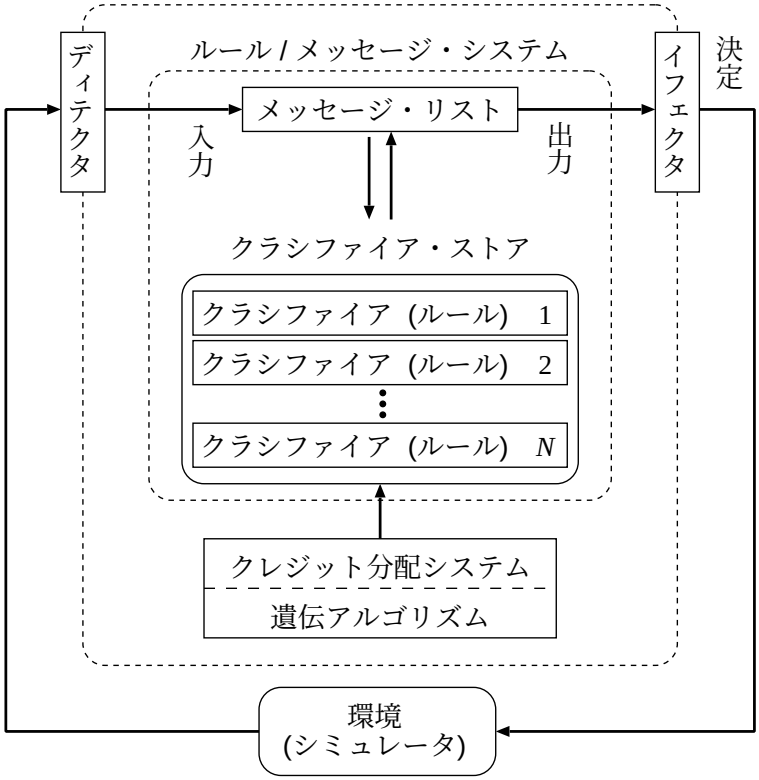


図 3.3: Michigan アプローチの概要

$$\langle \text{ message } \rangle \quad = \quad \{ 0, 1 \}^{l}$$

と定義される．ここで $\{ 0, 1 \}^{l}$ は 0 あるいは 1 を l 個並べた記号列を表わす．クラシファイアは極めて単純な形をしたプロダクション・ルールであり，

$$\langle \text{ classifier } \rangle \quad = \quad \langle \text{ condition } \rangle \quad : \quad \langle \text{ message } \rangle$$

となる．さらに，条件 $\langle$ condition $\rangle$ は，

$$\langle \text{ condition } \rangle \quad = \quad \{ 0, 1, \# \}^{l}$$

という形をとる．ここで，‘#’ は 0, 1 のどちらでも良いことを表わす．

クラシファイアの条件部があるメッセージに適合すると，そのクラシファイアは発火してメッセージを出すが，メッセージ・リストの容量が発火したクラシファイアの数より小さい場合，クラシファイアの強さ (信頼度) に基づいてメッセージを出すことのできるクラシファイアを選択するものとする．そこで，このクラシファイアの強さを合理的に調節することが必要となり，それを行うのがクレジット分配システムである．

(b) クレジット分配システム

環境への望ましい出力に対するクラシファイアの貢献度に応じて，クラシファイアの強さを調節するのがクレジット分配システムである．これの代表的なものとして，経済システムを手本とするバケツリレー・アルゴリズム (Bucket Brigade Algorithm: BBA)

およびプロフィット・シェアリング (Profit Sharing Pattern: PSP) がある [33]．

BBA では，まずメッセージに適合したクラシファイアが，その強さに比例したビッドを提示してオークションに参加する．その結果，ビッド値の大きいものほど高い発火優先権が与えられ，優先度の高いクラシファイアから順にビッド値を支払ってメッセージを出していく．この際，発火したクラシファイアによって支払われたビッドは，それを発火させるメッセージを出したクラシファイア間で分配される．さらに，環境への出力を引き起こすメッセージを出してクラシファイアは，環境から報酬を受けることになる．

PSP では，発火したクラシファイアの系列が記憶される．そして，報酬が得られた場合には，あらかじめ定められた配分率に従ってそれまでに用いられた一連のクラシファイアに報酬が分配される．

BBA あるいは PSP では，各クラシファイアの強さを正当に評価するだけであり，新しいクラシファイアを生成しない．そこで，新しいクラシファイアを生成するために GA が用いられる．

(c) GA

クレジット分配システムによって評価されたクラシフィアの強さをその環境への適応度と見なして GA を適用する．

ここで，計算終了までの (GA の) 世代数を N_{g} , GA を適用するまでの繰り返し回数を N_{b} とすると，Michigan アプローチによるルール学習アルゴリズムは以下のようになる．

1° ルールを個体にコーディングし，初期個体群 $\mathcal{P}(1)$ を生成する．世代を $t=1$, 繰り返しカウンタを $c=1$ とする．

2° 環境からの入力に対して適合するルールを選択し，対象とする問題に対して適用する．適合するルールが複数ある場合は，それらの信頼度に応じた確率で一つのルールを選択し適用する．

3° ルールを適用して報酬が得られた場合，報酬が得られるまでに用いられた一連のルールに対して報酬を分配し，各ルールの信頼度を変更する．$c < N_{\mathrm{b}}$ であれば，カウンタを $c=c+1$ として 2° へ．そうでなければ 4° へ．

4° $t < N_{\mathrm{g}}$ であれば 5° へ．そうでなければ 6° へ．

5° 各ルールの信頼度を適応度とみなして GA を適用し，次の世代の個体群 $\mathcal{P}(t+1)$ を生成する．世代を $t=t+1$, カウンタを $c=1$ とし 2° へ．

6° 計算終了．最終的に得られている個体群を目的とするルール集合とする．

3.4.3 二つのアプローチの比較

一般にいくつかのルールを適用して問題を解くとき，個々のルールではなく一連のルールを連続して適用した結果に対して評価 (報酬) が与えられる場合が多い．本研究で対象とする在庫設備選択問題の場合においても， 複数のルールで構成されるルール集合を適用することによって一つのスケジュールが得られ，目的関数の値から評価値を求めることになる．

ミシガン・アプローチでは，ルールが個体に，ルール集合が個体群に対応しているので，得られた評価を各ルールに分配し， 各ルールの信頼度 (適応度) を調節する仕組み (クレジット分配システム) が必要となる．Pitt アプローチが前もってルールを生成・獲得するといった枠組みにならざるを得ないのに対し，ミシガン・アプローチは，オンライン学習が可能であり，リアルタイム性という点に関してミシガンアプローチが優れている．

しかし，各ルールの良さを正当に評価できるクレジット分配システムを構築することは困難である場合が多い．これに対して，Pitt アプローチではルール集合が個体に対応している．そのため一つのルール集合を適用した結果，得られた評価をそのまま個体の適応度とみなすことができるので，ミシガン・アプローチのようなクレジット分配システムは必要とされない．よって本論文では，ミシガン・アプローチよりも Pitt アプローチのほうが評価を適応度により正確に反映することができると考え，以下バッファ選択ルールの学習に対し Pitt アプローチを用いた解法構成を行う．

3.5 結言

本章では，第 2 章で整理したスケジューリング・ルールを遺伝的機械学習に基づいて生成・調整するための計算モデルを示した．その際，シミュレーションに基づいて，事前 (オフライン的) にルール集合を獲得する Pitt アプローチの構成を考慮した．第 4 章および第 6 章で用いられる枠組みとして，ルールをスケジューリング状況に応じて意思決定 (ディスパッチング) するためのパラメータ (対象システムから観測される特性量から決定事項へのマッピングを定めるためのパラメータ) から構成し，パラメータ空間で最適な (あるいはロバストな) ルールを探索する方式を示した．

第 4 章および第 6 章では，在庫設備選択問題 (第 4 章) およびフレキシブル・ショップ問題 (第 6 章) への具体的なルール獲得法の構成を示し，GBML に基づくアプローチの有効性を確かめる．

第 4 章

在庫設備選択問題に対するルール獲得法

4.1 緒言

本章では，第 3 章で示した遺伝的機械学習を用いたルール獲得方式について，在庫設備選択問題への適用方法を提案する [34]．在庫設備選択問題は，ある機械で処理が完了した (半) 製品を逐次在庫設備に振り分ける問題である．在庫設備の入出庫に対してコストが発生し，適当な時間間隔で発生する製品に対して，製品の属性 (重量，保管時間など) や在庫設備の空き状況に応じて入庫する在庫設備を決定する．このとき，計画期間における総コストを最小とすることが目的となる．本章ではまず，在庫設備選択問題について，問題の属性，決定，制約条件および評価関数を整理することによって問題を明確にする．

次にこの問題に対して，ルールを用いたスケジューリングを考慮する．ルール構成として，前件部に入庫する製品の属性および在庫設備の空き状況に関する条件を記述し，後件部に決定事項 (製品を割り付ける在庫設備番号) を直接記述したものを提案し，このルールを遺伝的機械学習を用いて獲得する．ルールベースのスケジューリング・システムに対して遺伝アルゴリズムを適用する試みとしては，例えば計画型エキスパートシステムを前提として，スケジュール生成における状態選択ルールを GA で探索する研究 [35] が挙げられる．これに対して本研究では，ルールそのものを GA で獲得することを目的としている．一方，遺伝的機械学習によるスケジューリング・ルール獲得については，従来，入木型の先行関係制約を持つスケジューリング問題を対象として遺伝的機械学習を構成した研究 [11] があるが，ここでのモデルでは，(離散) 状態の全てにいずれかのルールが対応するように状態および前件部を設計している．これに対し提案手法では，ルール集合が状態空間の一部のみを保持するものとした上で，前件部にドント・ケア記号を導入することにより，多様な状態に対して少数のルールで対応可能になり，大規模な問題への適用も可能となる．

最後に，例題を用いた計算機実験を通して，獲得されたルール集合を適用して得られる解の良さ，ルール集合の内容，ならびに汎用性について検討し，提案手法の有効性を確認する．

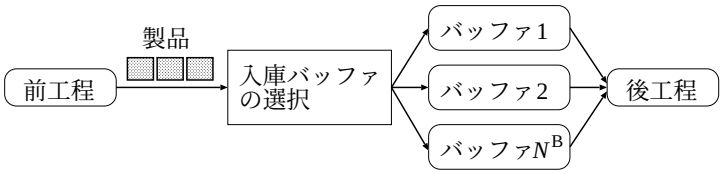


図 4.1: 在庫設備選択問題

4.2　問題の記述

生産工場や倉庫物流システムなどにおいては，処理待ちの製品 (半製品) を一時的に保管する在庫設備 (バッファ) がよく利用されている．大規模なシステムにおいてバッファは一種類とは限らず，製品は複数個のバッファの中から選択されたある一つのバッファに保管される．本章では，対象として前工程と後工程の間に在庫設備 (バッファ) があるというようなモデルを考える．バッファへの入出庫においては，

1) 製品の到着,
2) 入庫バッファの決定,
3) バッファへ入庫・保管,
4) バッファから出庫

といった作業が行われる (図 4.1)．これらの作業にはコストが発生し，バッファの特徴 (容量，保管コストなど) が異なる場合，適当な時間間隔で発生する製品に対して製品の属性 (重量，保管時間など) やバッファの空き状況に応じてバッファを選択し，保管に要する総コストを少なくすることが望ましい．そこで，計画期間における運用総コストが最小となるように各製品を保管するバッファを選択する問題 (在庫設備選択問題と呼ぶ) を考える．

本問題においてスケジュールを立案するタイミングについて考えると，事前には製品に関する属性が与えられず，計画段階にスケジュールを立案することができないものとする．従って実行時期において，到着した製品に対して随時バッファの選択に関する意思決定を行うアプローチが取られることになる．

在庫設備選択問題の構造について見ると，図 4.2 に示すように,

(a) 上位レベルでの決定に関する情報や過去の保管製品の時系列データに基づいて，将来の保管製品の時系列を予測する問題，および
(b) 上記 (a) の予測データならびに入庫対象となる製品の属性情報や各時点でのバッファの在庫状況に基づいて，各製品が保管されるバッファ(ならびに横持ち作業) を決定する問題,

のような部分問題からなっていると捉えることができる．以下では，(a) の予測データが利用できない状況での最適化問題 (図 4.1 中の (∗) の部分) を考慮する．つまり，決定を行う意

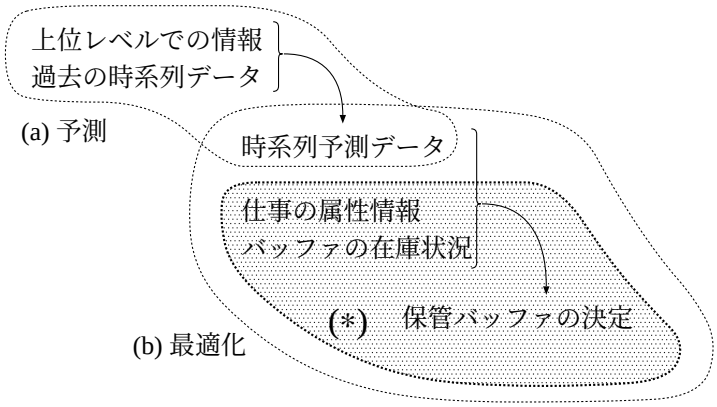


図 4.2: 在庫設備選択問題における予測と最適化

思決定者は，到着した製品がいつ出庫されるかに関する情報を利用できない．よって，事前にスケジュールを準備せずに，生産実施時期において到着した製品に対して随時バッファの選択に関する意思決定を行う．

以上で示した問題の基本要素，制約項目，決定事項および評価項目を以下のように整理する:

(A) 基本要素と属性:

- 製品 P_j $(j = 1, \ldots, N^{\mathrm{P}})$,
  - 到着時刻 t_j^{I},
  - 出庫時刻 t_j^{O},
  - 重量 w_j *,
  - 属性情報 (番号) a_j,
  - 指定バッファ(番号) s_j †,
- バッファ B_i $(i = 1, \ldots, N^{\mathrm{B}})$,
  - 在庫製品総重量 w_{it}^{Z},
  - 入庫作業コスト c_i^{I} (単位容量あたり),
  - 保管作業コスト c_i^{K} (単位容量・時間あたり),
  - 出庫作業コスト c_i^{O} (単位容量あたり),
- 計画期間 T.

(B) 制約項目:

- 初期在庫量 w_{i0}^{Z} に関する制約 (制約 (a)),

* 本論文ではバッファの “大きさ” を重量で表す.

† この属性を持つ製品は完成段階で入庫バッファが指定されており，バッファ選択 (決定) に関する自由度はない.

- 許容量 $\overline{w}_i$ に関する制約 (制約 (b)),
- 指定バッファ s_j に関する制約 (制約 (c)).

(C) 決定事項 (決定変数) :

- 製品を入庫するバッファ(番号).

(D) 評価項目 (目的関数) :

- 製品の入出庫・保管に際する総コスト (最小化),
- 横持ち[‡]の発生回数 (最小化).

4.3　ルールを用いたスケジューリング方式

4.3.1　ルールの構成

4.2. に記述した在庫設備選択問題では，製品を入庫するバッファの決定は，その製品が到着した時点で行われる．このような問題に対し，本論文ではバッファ選択のための優先ルールを用い，このルールを学習によって自動的に獲得・調整する枠組みを考える．

入出庫計画の作成においては，在庫状況を表す三種類の属性:

(a) 製品 P_j の重量 w_j,

(b) バッファB_i の在庫製品総重量 w_{it}^{Z}，および

(c) 製品 P_j の属性情報 a_j

をそれぞれ状態変数とする三次元の状態空間 $\mathcal{S}$ を用意し，あらかじめ各状態変数に適当な閾値を設けて $\mathcal{S}$ を互いに独立な部分空間 $\mathcal{S}_k\,(k = 1, \cdots, n^{\mathrm{S}})$ に分割しておく．このとき，各意思決定のタイミングにおける状態がどの部分空間に属しているかということを前件部⟨condition⟩とし，そのときの意思決定の内容を後件部⟨action⟩ とするルール:

$$\text{if } \langle\text{condition}\rangle \text{ then } \langle\text{action}\rangle \tag{4.1}$$

を導入する．ルール適用時には，在庫設備システムの状態と前件部が比較され，一致したルールの後件部に記述された内容に基づいてバッファが選択される．

図 4.3 にルール構成を示す．状態変数 w_j には一つの記号を対応させ (前件部 (1))，状態変数 w_{it}^{Z} に対しては，バッファの数 N^{B} だけ記号を用意する (前件部 (2))．前件部 (1) および (2) はバッファ容量に関する状態を特定するために用いられるものであり，それぞれ状態を数段階に離散化した値もしくはドント・ケア記号 “#” が記述される．“#” はどのような入力に対しても一致する記号で，解空間を縮退させることが可能である．一方で Holland は，

[‡] 制約 (b) と制約 (c) により，あるバッファに指定された製品が到着した時点で そのバッファに十分な空きがなければ，そのバッファに保管されている製品の中から適当な製品を選んで代替バッファに移さなければならない．この作業を「横持ち」と呼ぶ.

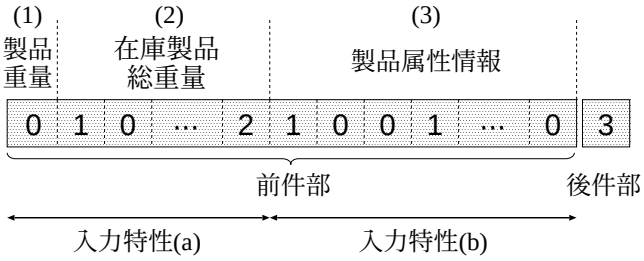


図 4.3: ルールの構成

「入力情報の特定部位のみに注目し，残りを無視するのに役立つ」と述べており [36]，動的な環境における入力情報の取捨選択機構として機能することも期待される．さらに，状態変数 a_j に対しては，製品の属性情報の種類数 N_{s} に応じたビットを用意し，入庫製品 P_j の属性と一致するルールの a_j 番目のビットには “1” が立つようにする (前件部 (3))．また，複数個の遺伝子座が同時に “1” となることを許すことによって，学習が進むにつれて入庫製品の属性に基づくクラスタリングが行われることが期待される．ルールの後件部については，製品を入庫するバッファ番号 B_i を記述するものとする．

本章では前件部に含まれる三種類の情報それぞれに応じた取扱いを考える．前件部 (1) および (2) はバッファ容量の制約項目に対する状態の記述であり，これらをひとまとまりとして扱う (「入力特性 (a)」と呼ぶ)．一方，前件部 (3) を「入力特性 (b)」と呼ぶ．入力情報と完全に一致するルールが見つからない場合には，あらかじめ定められる入力特性のランキングに基づき，ランクの高い入力特性から順に，その特性のみに注目したルールとのマッチングが行われ，入力情報と最初に一致したルールが適用される．これにより，特定の入力特性に注視したヒューリスティクスを設計者が導入することも可能となる．

4.3.2 入出庫の手続き

4.3.1 で示したルールを用いた入出庫計画の作成手続きは以下の通り:

1° 入庫対象となる製品に指定バッファについての属性がない場合，2° へ．そうでない場合，横持ち作業が必要であれば行い，入庫作業をして 4° へ．

2° 発火したルールが複数個の場合，前件部 (3) において “1” であるビットの数が最も少ないルールを選択し，適用する．発火したルールがない場合には，あらかじめ定められたバッファを選択する．

3° 2° で選ばれたルールの後件部で指定されたバッファに入庫可能であれば (入庫後も容量制約を満たす場合)，入庫作業を行う．そうでない場合，あらかじめ定められたヒューリスティックに従って (代替の) 入庫バッファを決定し，入庫作業を行う (この作業を「横振り」と呼ぶ)．

4° 次の入庫対象製品 (イベント) が発生するまで時刻を進める．1° へ戻る．

ここで，手順 1° において，横持ち対象製品 P_{ℓ^*}，ならびに代替倉庫 $\mathrm{B}_{i_{\ell^*}}$ は，横持ちによるコストを最小限に抑える観点から次のヒューリスティックに従い決定される．

1° 横持ち対象製品 P_{ℓ^*}:　指定バッファに入庫済みの製品 P_ℓ ($\ell \in \mathcal{B}$; $\mathcal{B}$ は横持ちが発生した時点でバッファ内に保管されている製品の集合) の中から，重量 w_ℓ が入庫対象製品以上であるものを選び，その中で最も重量が軽い製品を P_{ℓ^*} とする，

2° 代替倉庫 $\mathrm{B}_{i_{\ell^*}}$:　P_{ℓ^*} を入庫可能であるようなバッファの中から，ランダムに一つを選び，$\mathrm{B}_{i_{\ell^*}}$ とする．

4.4　遺伝的機械学習によるルール学習方式

4.4.1　概要

4.3.1 で示したルールを自動的に獲得・調整するために，第 3 章で述べた遺伝アルゴリズム (Genetic Algorithm: GA)[9] を用いたルール学習システムである遺伝的機械学習 (Genetics-Based Machine Learning: GBML) を適用する．GBML は大きく二つのアプローチ，すなわち Michigan アプローチと Pitt アプローチに分類されるが，Michigan アプローチでは個体 (ルール) の適応度の算出にクレジット分配機構を用いており，計算機上に実現することは容易ではない．これに対して，Pitt アプローチでは予めルール集合を生成・獲得しておくといった枠組みになるものの，Michigan アプローチのようなクレジット分配システムを必要としないために実装が容易である．そこで，本研究では Pitt アプローチを用いた解法構成を考える．図 4.4 に，在庫設備選択問題に対して Pitt アプローチを適用する場合の枠組みを示す．

4.4.2　遺伝アルゴリズムの構成

GBML の適用に際して，GA における個体の遺伝子表現，適応度の計算，そして遺伝演算子を具体化する必要がある．

(1) 個体の遺伝子表現

Pitt アプローチでは，ルール集合を GA における個体として取り扱う．そこで，図 4.3 のように表現される一つのルールを $r_k\,(k = 1, \cdots, N_\mathrm{r})$ として，これを複数個順番に並べて表わされるルール集合:

$$\{r_1, r_2, \cdots, r_{N_\mathrm{r}}\}$$

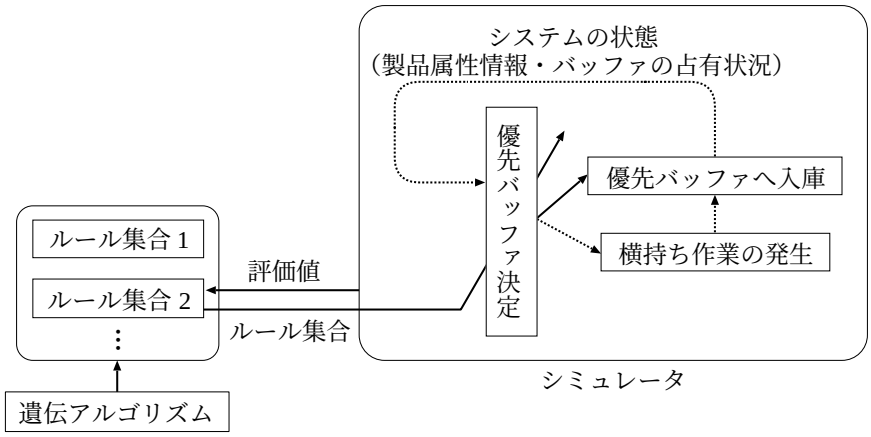


図 4.4: Pitt アプローチによるルール学習の枠組み

を, 個体の遺伝子表現とする. ただし N_{r} は一つのルール集合に含まれるルールの数であり, 予め定めておく.

(2) 適応度計算

個体の適応度はルール集合の評価値に対応している．ルール集合の評価は，(1) 入庫・保管・出庫作業コスト，(2) 横持ち回数 に加えて，ルールが有効にはたらいているかの指標として，(3) 横振り回数 に注目して行う．これより適応度 g_k を (1)，(2)，(3) の各評価値の重み付き線形和として求める．

(3) 遺伝演算子

GA を適用するにあたって，以下に示す三種類の遺伝演算子 [29] を用いる．

(a) 選択・複製: 期待値選択にエリート戦略を組合せたものを適用する．期待値選択の手順を以下に記す:

1° 各個体 I_k について次世代に残る数の期待値 $e_k = g_k/\overline{g}$ を求める. ただし g_k および $\overline{g}$ は, それぞれ個体 k の適応度および個体群での適応度の平均値である.

2° 各個体について $\lfloor e_k \rfloor$ 個の個体を次世代に残す. ただし, $\lfloor x \rfloor$ は x 以下の最大の整数である.

3° 次世代の個体数が N_{p} に等しいならば選択終了.

4° $e_k - \lfloor e_k \rfloor$ に応じた確率に基づくルーレット選択を, 次世代の個体数が N_{p} となるまで行う.

(b) 交叉: 個体群中の個体をランダムにペアリングし，交叉確率 p_{c} で一点交叉を行う (交

表 4.1: 倉庫運用の作業コスト

倉庫	c_i^{B}	c_i^{I}	c_i^{O}	c_i^{K}
B_1	200	200	200	0
B_2	400	400	400	20

又は，ルール集合間でのルールの交換に対応する).

(c) 突然変異 : 各個体の遺伝子座をランダムに一ヶ所選び，この遺伝子座に対応するルールの前件部 (1)，(2) あるいは (3) のいずれか一つをランダムに選択し，突然変異確率 p_{m} で，許容範囲内の任意の数値に変更する (突然変異は，ルール集合内でのルールの変更に対応する).

4.5　計算例

4.5.1　例題

出荷する製品を二つの倉庫 B_1 および B_2 に振り分けて保管を行うような倉庫物流システムを考える．なお，B_1 は容量が小さいが輸送・保管コストが少なくて済むのに対して，B_2 は容量が大きい分だけ輸送・保管コストも多くかかるという特徴をもつ．具体的には，

- 計画期間: 36 日

であり，製品について，

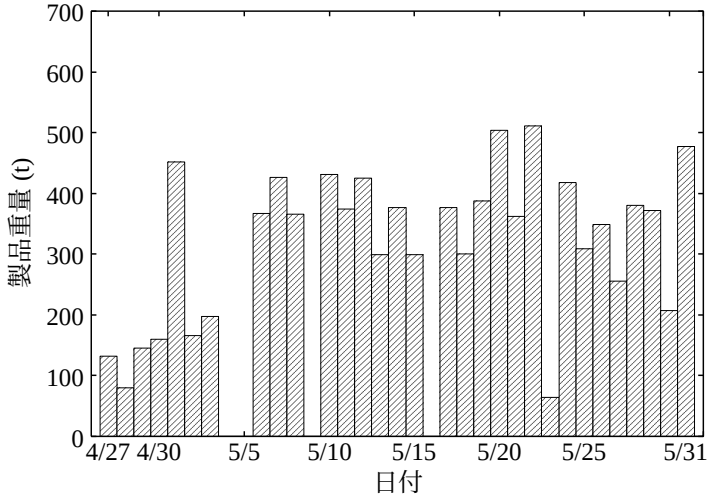
- 入庫製品総数 (総重量): 7,412 (9,969,921 kg),
- 出庫製品総数 (総重量): 6,639 (8,514,600 kg),
- 製品種類: 24,
- 納入先: 16,
- 指定倉庫の属性を持つ製品総数: 1,050

という属性を持つ．また，計画期間中の入出庫および指定倉庫製品の重量を図 4.5 に示す．

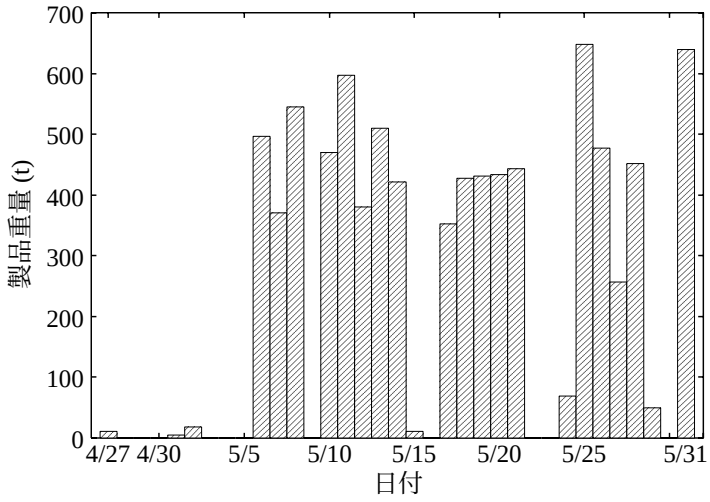
倉庫について，

- $\mathrm{B}_1(\mathrm{B}_2)$ の容量: 2,500 t (15,000 t),
- 計画期間開始時には，B_1, B_2 ともに容量の 75% を使用しているものとする

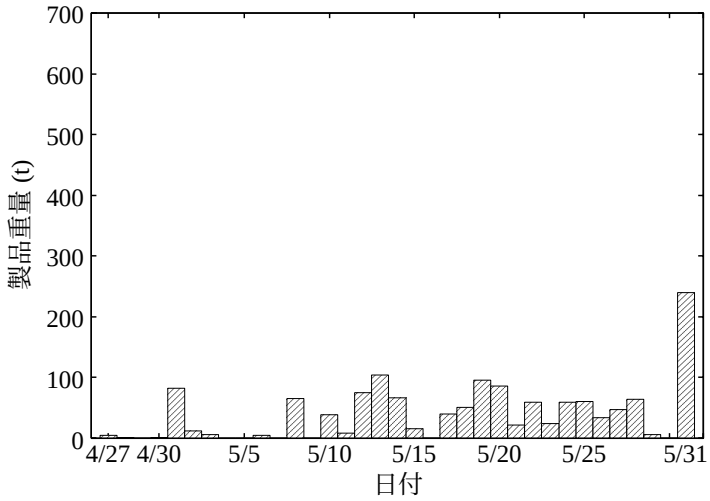
という属性を持つ．また，倉庫運用におけるコストを表 4.1 に示す．なお，横持ち作業および横振り作業に関しては，それぞれ 400 (円/t)，200 (円/t) としてコスト換算するものとする．



(a) 入庫製品



(b) 出庫製品



(c) 指定倉庫製品

図 4.5: 入出庫および指定倉庫製品の重量

4.5.2　アルゴリズムの構成

(1) ルールの構成

まず，GBML におけるルールの構成を以下のように定めた．

- 前件部：
  (a) 製品 P_j の重量 w_j
  入庫する製品の重量を次に示す三段階に離散化する．
  - 2,000kg 未満,
  - 2,000kg 以上, 6,000kg 未満,
  - 6,000kg 以上.

  (b) 倉庫 B_i の在庫製品総重量 w_{it}^Z
  倉庫 B_2 の容量は十分大きく，仮に計画期間中に全ての製品を B_2 に入庫しても満杯にはならない．よって，B_1 の在庫製品総重量のみを考慮し，B_1 の倉庫占有率 o_{1t} を

$$o_{1t} = w_{1t}^Z \,/\, \overline{w}_1 \tag{4.2}$$

  と定義し，$o_{1t} = 0.90$ を閾値として二段階に離散化する．

  (c) 製品属性情報 a_j
  本例題において製品属性情報は，品種と納入先の二種類存在する．よって，品種と納入先を組として取り扱い，$N_\mathrm{s} = 384$ だけのビットを用意する．

- 後件部：
  倉庫番号 1 あるいは 2 を記述する．

このルールの構成法を以下では「ルール構成 (a)」と呼ぶものとする．計算機実験においては，比較用としてルールの前件部を (1) および (2) のみで構成した場合 (ルール構成 (b)) と，前件部を (3) のみとして構成した場合 (ルール構成 (c)) についてルール学習を行う．さらに，ルール構成 (a) において，**4.3.1** で述べた入力特性に関するランキングについては，入力特性 (2) を入力特性 (1) に優先させるものとする．

入出庫の手続きの横振りについては，代替バッファとして，十分容量の大きい B_2 に入庫するものとする．

(2) 遺伝アルゴリズムの構成

GA の適用にあたっては，そこでのパラメータを 表 4.2 のように設定した．なお，表中の N_p, N_g, p_c, p_m および N_e は，それぞれ個体数，世代数，交叉確率，突然変異確率およびエリート保存個体数を表す．また，初期個体の作成にあたっては，個体中に含まれるルール数

表 4.2: GA のパラメータ

N_{p}	N_{g}	p_{c}	p_{m}	N_{e}
100	10000	0.5	0.1	1

$N_{\mathrm{r}} = 100$ としてランダムに作成する．なお，前件部 (3) に関しては，各ビットについて確率 p_{d0} で “1” となるようにする．

適応度の算出にあたっては，表 4.1 で示される入庫・保管・出庫コストに加えて，横持ち・横振りコストとして，それぞれ 400(円/t)，200(円/t) としてコスト換算したものをルール集合 r_k による総コスト c_k^{L} とする．ここで，提案手法の性能を検証するために，比較対象として実際の倉庫運用者の運用結果における総コスト c_{r} を導入する．このとき個体 k の適応度 g_k は，次のようにして求める．なお，M_1, M_2 はスケーリング定数であり，本例題では，$M_1 = 0.8$, $M_2 = 5$ とする．

$$g_k = \left(\frac{2 \times c_{\mathrm{r}} - c_k^{\mathrm{L}}}{c_{\mathrm{r}}} - M_1 \right) - M_2 \tag{4.3}$$

4.5.3　実験結果

計算機実験を，(1) 入力特性 (a) および (b) の提案手法における有効性，および (2) 入力特性 (b) に関して，初期解のルール学習結果への影響，を調べるために，次の二つの場合について行った．

(1) ルール構成 (a)，(b)，(c) を用いたルール学習．

(2) ルール構成 (a) で，p_{d0} を $\{0.005, 0.010, 0.050, 0.100, 0.200\}$ と設定したルール学習．

まず，計算機実験 (1) について，ルール構成 (b) に関しては，前件部 (1) および (2) に対応する状態をそれぞれ十段階に分割し，さらに，ルール集合が全ての状態に対応するように前件部を設計した．この際，GA の操作 (突然変異) が前件部には加えられないものとした．また，ルール構成 (c) に関しても同様にルール集合が全ての状態に対応するようにし，GA の操作が前件部に加えられないものとした．なお，ルール構成 (a) については，$p_{\mathrm{d0}} = 0.010$ とした．

図 4.6 に，ルール構成 (a)，(b) および (c) を用いた場合について，学習過程での個体群内で適応度が最大となる個体 (ルール) の総コストの推移を示す．なお，図には初期ルール集合を変化させて行った 5 回の試行の平均値を示している．これより，

- ルール構成 (a)，(b)，(c) について，取り扱える状態数は (a) の場合に最も多く，(c) が最も少ない．状態数がもっとも多くなるルール構成 (a) の場合に，適応度 (総コスト) の収束が遅くなるものの，より良い (低コストの) 解を生成するようなルールが求められている

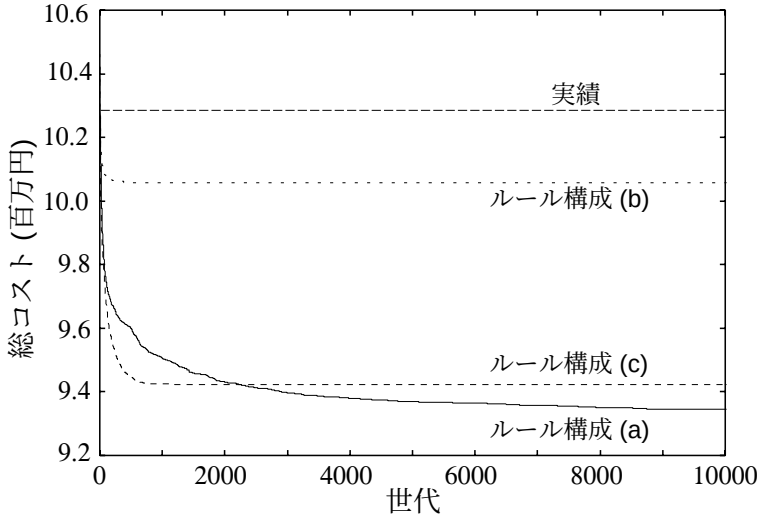


図 4.6: 学習過程における総コストの推移 (計算機実験 (1))

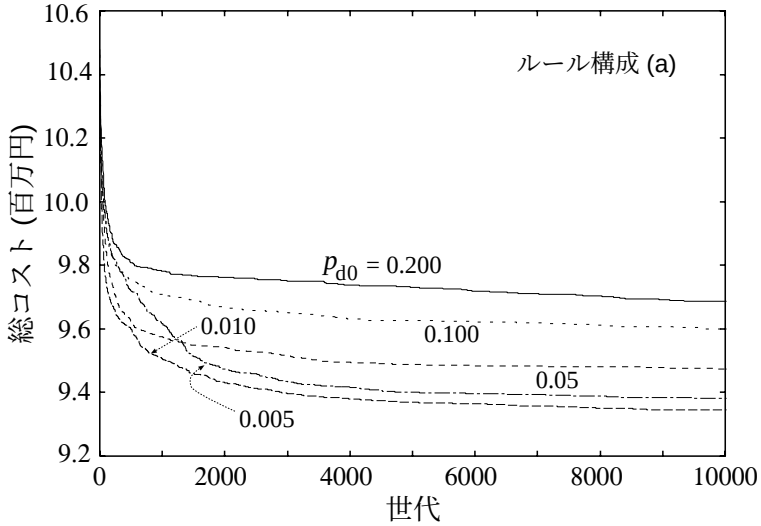


図 4.7: 学習過程における総コストの推移 (計算機実験 (2))

表 4.3: 前件部 (3) に “1” が含まれる割合

$p_{\mathrm{d}0}$	0.200	0.100	0.050	0.010	0.005
$p_{\mathrm{d}N_{\mathrm{g}}}$	0.2041	0.1052	0.0526	0.0168	0.0122

表 4.4: 計算時間

(a) 計算機実験 1

ルール構成	(a)	(b)	(c)
時間 (分)	387	279	88

(b) 計算機実験 2

p_{d0}	0.200	0.100	0.050	0.010	0.005
時間 (分)	279	293	299	387	467

ことがわかる．

次に，計算機実験 (2) について，図 4.7 に，$p_{\mathrm{d}0}$ を 0.005, 0.010, 0.050, 0.100 および 0.200 のそれぞれに設定した場合の学習過程における最大適応度個体 (ルール) の総コストの推移を示す．なお，計算機実験 (1) と同様，図には初期ルール集合を変化させて行った 5 回の試行の平均値を示している．また，獲得されたルール集合の前件部 (3) において “1” が含まれる割合を表 4.3 に示す．図 4.7 より，

- $p_{\mathrm{d}0} = 0.010$ のときに最も良い解を生成するようなルールが求められている，

ことがわかる．

なお，$p_{\mathrm{d}0}$ は初期ルールにおける前件部 (3) の “1” の割合を指定するものであるが，$p_{\mathrm{d}0}$ の値によって異なる解 (適応度が異なる値) に収束していることより，GA の操作が前件部 (3) に関して有効に機能していないと考えられる．これは，前件部 (3) に対する GA の操作が突然変異に限定されており，最終的に得られるルール中の前件部 (3) に “1” が含まれる割合 $p_{\mathrm{d}N_{\mathrm{g}}}$ が初期設定値 $p_{\mathrm{d}0}$ とほぼ変化がなかった (表 4.3) ことからも分かるように，Pitt アプローチがルール集合を個体として取り扱っているためにルール集合中の各ルールの評価が効果的に行われていないことが原因であると予想される．さらに，モデル獲得 (前件部 (3) におけるクラスタリング) および最適値探索が単一のルール探索過程に埋め込まれており，モデル獲得が陽に考慮されていないことも原因の一つであると考えられる．このような問題点に対処する試みとして，例えば階層型分類子システムを用いた注視点制御過程を導入する方法 [37] などが考えられる．

最後に，計算機実験 (1) および (2) について，ルール学習に要した時間を表 4.4-(a) および表 4.4-(b) にそれぞれ示す．なお，ここでの計算時間は，Pentium III 933MHz を用いた場

合のものである．

表 4.4-(a) より，ルール構成 (a) は他の構成に比べて前件部で定義される状態空間が大きく，また適用するルールの決定の方法が複雑であるために，学習により時間がかかることが分かる．また表 4.4-(b) より，p_{d0} の値が小さくなるにつれて，**4.5.4** で考察するように，入力情報と完全一致して適用されるルールが少なくなり，入力情報とのマッチングに時間がかかることが分かる．

4.5.4　得られたルール集合について

計算機実験により得られた最良ルール集合による倉庫運用について考察する．なお，ルール集合は計算機実験 (1) のルール構成 (a) で得られたものを選んだ．図 4.8 に，最良ルール集合による倉庫運用結果を示す．また，比較のため，エキスパートによる倉庫振り分けに基づいてシミュレーションを行い，評価値を計算した結果を図 4.9 に示しておく．さらに，両者の運用総コスト，最良ルール集合による横持ちおよび横振りの発生頻度および倉庫別の保管日数を，それぞれ 表 4.5，図 4.10，図 4.11 および 図 4.12 に示す．

最後に，提案方法により獲得された最良ルール集合 (ルール構成 (a)) における各ルールの適用状況を図 4.13 に示す．なお図中において (a)∼(d) は，それぞれ

(a) 入力情報と完全一致したとき，

(b) 入力情報と完全一致するがドント・ケアのビットを含むとき，

(c) 入力情報と製品属性情報の項目のみ一致したとき，

(d) 入力情報と，製品重量，倉庫占有率の項目のみが一致したとき，

を表す．さらに，ルール集合中において，一度も倉庫運用に用いられなかったルールの数を表 4.6 に示す．

(1) 倉庫占有率について

図 4.8 および 図 4.9 より，提案手法と実績ではほぼ同じ結果になっており，

- エキスパートによる倉庫運用で用いられているルールに準ずるルールが提案手法によって獲得できている

と考えられる．なお，表 4.5 を見ると，提案手法により得られた最良ルール集合による運用総コストが倉庫運用者によるものよりも小さくなっており，より効率的な運用が行われていることが確認できる．

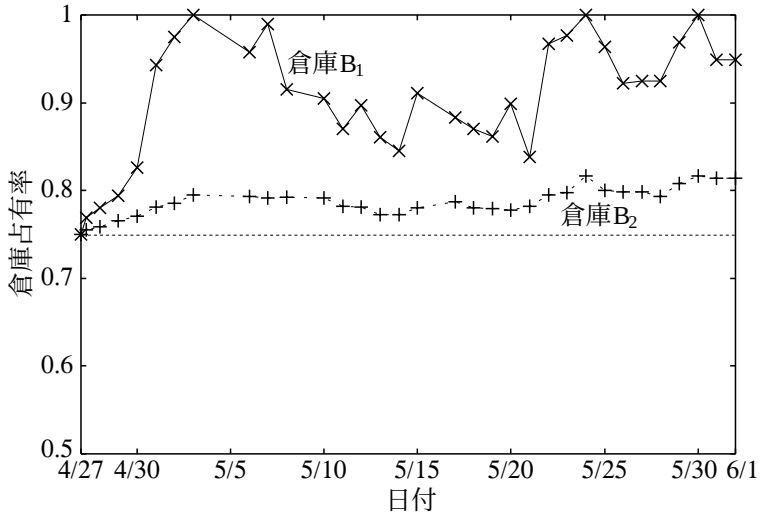


図 4.8: 倉庫占有率の推移 (提案手法)

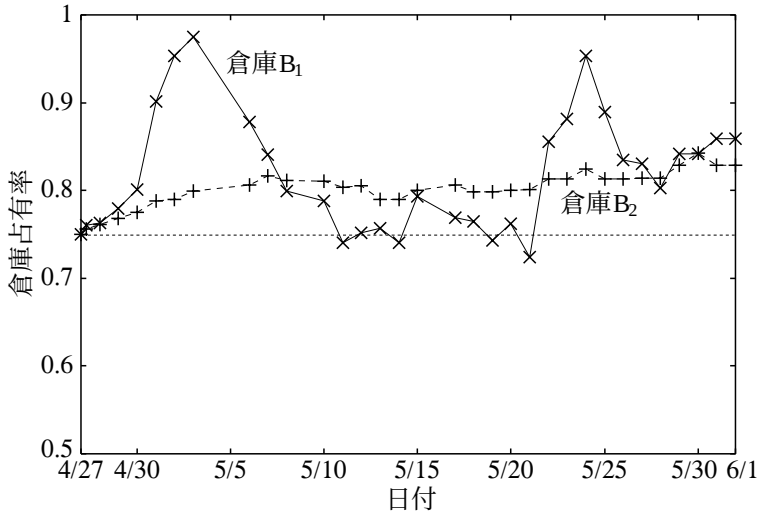


図 4.9: 倉庫占有率の推移 (実績)

表 4.5: 運用経費の比較

	総コスト	差
実績	10,286,850	—
提案方法	9,296,533	990,317

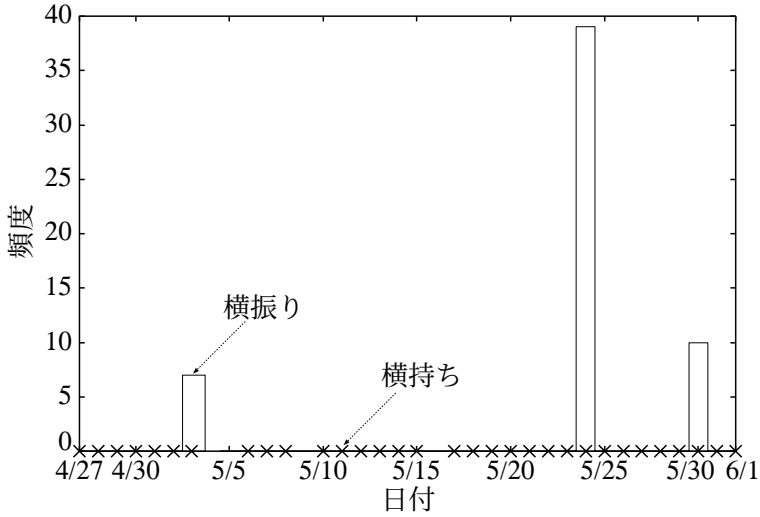


図 4.10: 横持ち・横振りの頻度

(2) 横持ちおよび横振りの発生頻度について

5/5 前後および 5/24 前後に指定倉庫の属性を持つ製品が多く入庫されるなどの理由により，エキスパートによる運用では横持ち・横振りが多く発生しているが，最良ルール集合による倉庫運用では，横持ち作業は全く生じなかった．また，横振りもほとんど発生しておらず，これは「ルールを適用したものの，実際には決定どおりに入庫できなかった」場合が少ないことを意味し，ルールが有効にはたらいていることを確認することができる．

(3) 倉庫別の保管日数の分布について

図 4.11 および 図 4.12 より，提案手法により獲得されたルール集合では保管日数が短い製品を倉庫 B_1 に多く入庫する傾向が見られ，倉庫 B_1 が有効に活用されていることがわかる．すなわち，

- 提案手法により，製品が保管日数に関して分類され，保管日数の短い製品を倉庫 B_1 に入庫するルールが獲得されている

ことが分かる．

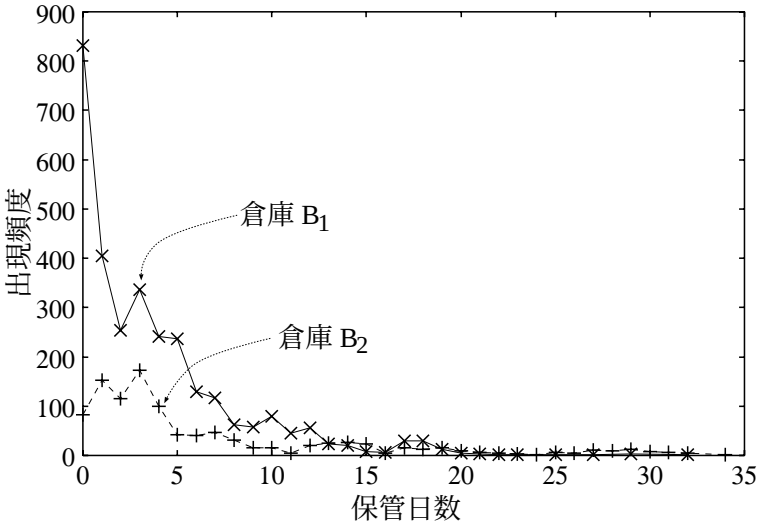


図 4.11: 倉庫別保管日数の集計 (提案手法)

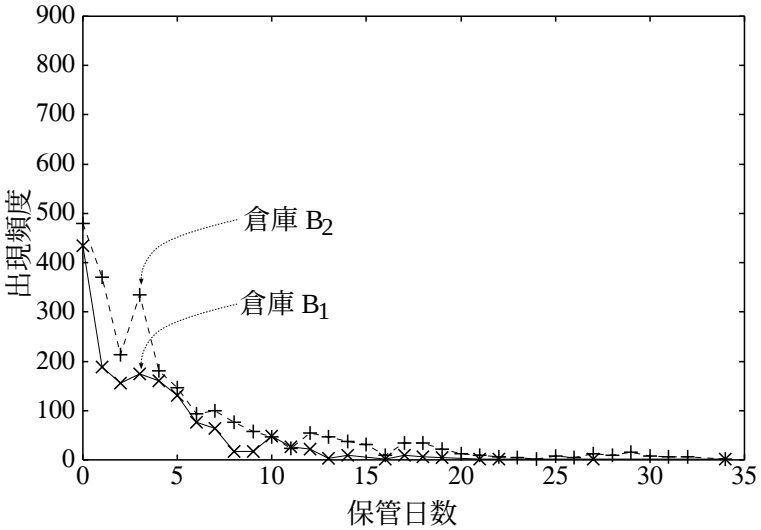
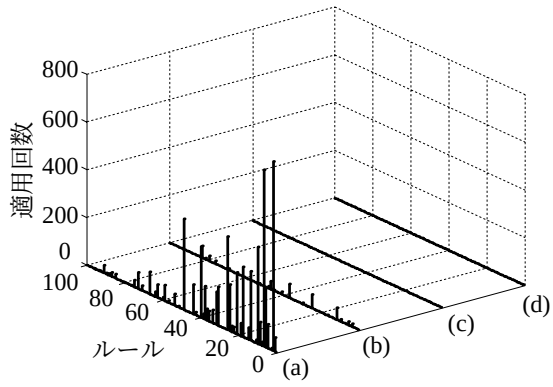
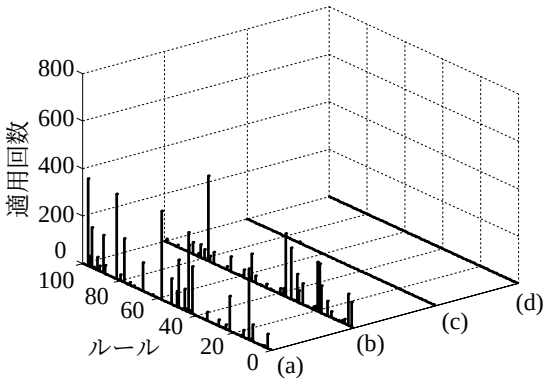


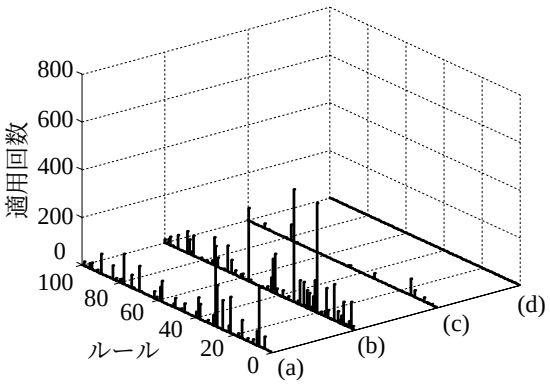
図 4.12: 倉庫別保管日数の集計 (実績)



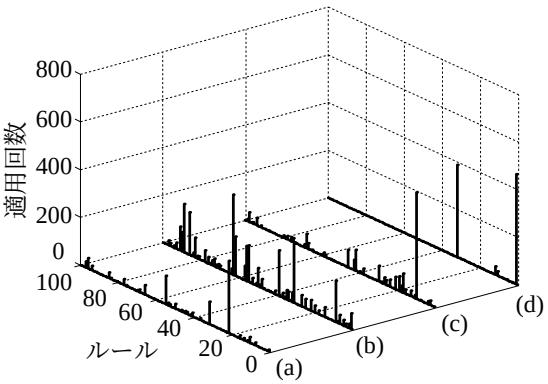
(a) $p_{d0} = 0.200$



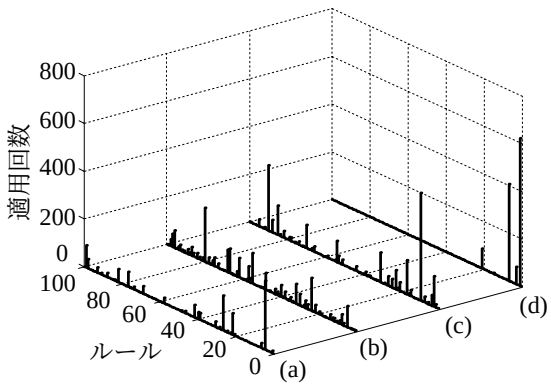
(b) $p_{d0} = 0.100$



(c) $p_{d0} = 0.050$



(d) $p_{d0} = 0.010$



(e) $p_{d0} = 0.005$

図 4.13: ルールの適用状況

表 4.6: 倉庫運用に用いられなかったルールの数

$p_{\mathrm{d}0}$	0.200	0.100	0.050	0.010	0.005
ルール数	75	84	95	97	97

表 4.7: ルール集合の汎用性

ルール構成	(a)	(b)	(c)
r_{c}	1.096	1.043	1.089

(4) ルール集合中のルールの適用状況について

最後に，ルール集合中のルールの適用状況について調べた結果，計画期間中に適用できるルールが見つからないというような状況は発生しなかっが，図 4.13 より $p_{\mathrm{d}0}$ の値によって状況が異なることがわかった．すなわち，$p_{\mathrm{d}0}$ の値が大きい場合には，**4.5.3** でも考察したように $p_{\mathrm{d}0} \simeq p_{\mathrm{d}N_{\mathrm{p}}}$ であり，入力情報と前件部 (3) が一致しやすい．また，入力情報とルールとのマッチングにおいて，先に一致したルールを優先して発火させているために，特定のルールが何度も適用される傾向にあり，一度も適用されないルール数も多い (表 4.6)．一方，$p_{\mathrm{d}0}$ の値が小さい場合には入力情報と完全に一致するルールが少なく，入力特性の取捨選択により発火したルールが多くなっているという状況が確認された．また，入力情報と完全一致したルールにおいても，ドント・ケアを含むルールが多く発火していることが見て取れる．さらに，表 4.6 よりこのときにはほぼ全てのルールが適用されている．

4.5.5 ルール集合の汎用性について

計算機実験により得られた最良ルール集合の汎用性について考察する．まず，計画期間を二つに分割し，期間の前半においてルール学習を行い，それによって得られた最良ルールを後半に適用する．そのときの総コスト c_{e} を用いて，

$$r_{\mathrm{c}} = c_{\mathrm{e}}/c_{\mathrm{d}} \tag{4.4}$$

から定義される総コスト比 r_{c} を計算することにより汎用性を確かめる．なお，コスト c_{d} は計画期間の後半において，学習を行って獲得された最良ルールによる総コストを示す．

ルール構成 (a), (b) および (c) を用いた場合の総コスト比を表 4.7 に示す．なお，表には初期ルール集合を変化させて行った 5 回の試行の平均値を示している．

表 4.7 より，いずれのルール構成を用いた場合においても，結果に大きな際は認められなかったものの，ルール構成 (b) は，対象システムの動的な特性量 (バッファ容量の制約項目に対する状態) のみで前件部が定義されていることから，他のルール構成の場合と比べて，多少汎用性が優れていることが分かる．一方，入力特性 (b) は，時間的な状態変化に対して静

的であることから，ルール構成 (a) および (c) を用いた場合には，比較的汎用性が優れないことが分かる．

4.6　結言

本章では，リアルタイム・スケジューリング問題の一例として，在庫設備選択問題を取り上げ，ルールによるスケジューリング方法ならびに，遺伝的機械学習の Pitt アプローチによるルール学習法を構成した．この際，

- ディスパッチング・ルールを逐次問題に適用，
- 意思決定に用いるルール集合を事前に準備するオフライン学習アプローチ，
- バッファ容量に対する制約項目と入力製品属性を前件部に，優先バッファを後件部とするルール構成

という考えに基づく手法を提案するとともに，計算機実験を通して優先度に基づくルール集合の獲得を行い，提案手法の有効性を確認した．その結果，提案手法により，効率的な運用を行うようなルール集合が獲得されていること，すなわち製品の保管日数に関する分類に基づいて保管日数の短い製品を作業コストの低いバッファに入庫する傾向を有するルール集合が獲得されていることが確認された．

今後の課題として，獲得されたルール集合の理論的な解析や，ルール集合の適用により得られたスケジュールの最適性の検証などが挙げられる．

第 5 章

フレキシブルショップ問題の数理計画モデルとハイブリッド解法

5.1 緒言

近年，スケジューリング問題に対して，理論および現実的な観点から多くの報告がなされている．特に理論的な研究の多くは，単純化された古典的な型の問題に関するものがほとんどであり[1, 2]，実際の様々な型の問題に対応することは難しい．一方，現実的なスケジューリングに関する研究においては，ディスパッチング・ルールを基にしたものが多い[3]．すなわち「どのルールがどの問題に対して有効か」といったことに焦点が当てられ，多くの報告がなされている．本章および次章ではこの実際と理論との乖離に留意し，古典的なスケジューリング問題の中で最も広範な応用の可能性を有し，ジョブショップ問題の拡張として位置づけられるフレキシブルショップ問題を取り上げる．このフレキシブルショップ問題は，ジョブショップ問題において，各ショップに複数の機械が並列に存在するような問題であり，本研究では，さらに現実的な応用の観点から，段取り替えなどに関する制約も考慮する．

解法の構成にあたっては，それに先立って，必ずしも数理的でないにしても，決定事項，制約条件そして評価基準 (目的関数) を，手続きを含まない形で明確にしておくこと (最適化モデルの記述) が望ましい．このようなモデル化によって，種々の解法を構成・比較しやすくなる．そこで本研究では，フレキシブルショップ問題を数理計画モデルに定式化するとともに，そのモデル基づいて種々のアプローチを構成・比較する．

これまでに，フレキシブルショップ問題に対していくつかの研究が報告されているが，それらの多くがフレキシブルショップ問題のサブセットに相当するフレキシブル・フローショップ問題 (全ての仕事の処理順序が同じ場合) を対象として，離接グラフ・モデルに基づく種々の効果的な解法が構成されている．例えば，分枝限定法[2]やヒューリスティクス[38, 39]を独自に構成しているものがある．一方，より一般的なフレキシブルショップ問題に対しては，非線形整数計画問題への定式化に基づく解法[40]や，離接グラフあるいはシミュレーションに基づくメタ戦略 (遺伝アルゴリズム，タブー探索など) の適用法に関する研究[41, 42, 43]などがあるものの，特別にカスタマイズされた解法 (アルゴリズム) 以外の方法を構成・適用

することが困難であったり [40, 41]，(シミュレーション) モデルが解法と明確に分離して記述されておらず，他の解法の適用・比較が困難である [42, 43] といった問題点がある．

そこで本章では，まずフレキシブルショップ問題を混合整数計画問題 (Mixed-Integer Programming: MIP) として定式化し，数理計画法の適用を可能とする．しかし，一般に計算量の観点から，問題の規模が大きくなるにつれて (最適) 解を得ることが困難になる．そこで，準最適解を効率的に獲得するために，メタ戦略の一つである進化型計算を適用する枠組みを示す．さらに，フレキシブルショップ問題に対するルール獲得法を構成する (第 6 章) ことを踏まえて，定式化に基づき，2 種類の決定事項を定める方法に関して 3 つのアプローチを提案して，その有効性を確かめる [44, 45]．すなわち，解の良さと計算時間をより高いレベルでバランスさせるため，数理計画法と進化型計算法との階層型ハイブリッド解法を構成することを考える．具体的には，定式化された MIP の一部の決定変数を進化型計算法での探索対象とし，残りの変数を数理計画法によって決定する．このとき，進化型計算法が探索する決定変数の種類に応じて二種類の方法を提案する．最後に，計算例を通して提案手法の有効性・可能性について考察する．

5.2　問題の記述

5.2.1　フレキシブルショップ問題

本研究で対象とするフレキシブルショップ問題を以下のように定義する．N^{J} 個の仕事 J_j $(j = 1, \cdots, N^{\mathrm{J}})$ を，N^{M} 台の機械 M_i $(i = 1, \cdots, N^{\mathrm{M}})$ で処理するものとする．仕事 J_j は，n_j 個の作業 O_k $(k = N_{j-1} + 1, \cdots, N_j;\ N_j = \sum_{\ell=1}^{j} n_\ell;\ N_0 = 0)$ で構成され，各仕事の作業は，$\mathrm{O}_k \rightarrow \mathrm{O}_{k+1}$ $(k = N_{j-1} + 1, \cdots, N_j - 1)$ の順に処理される．各作業 O_k $(k = 1, \cdots, N;\ N = N_{N^{\mathrm{J}}} = \sum_{j=1}^{N^{\mathrm{J}}} n_j)$ に対して処理可能機械集合 $\mathcal{M}_k$ と処理タイプ δ_k といった属性があり，この処理タイプ δ_k と機械 M_i との組み合わせによって処理時間が決定する．

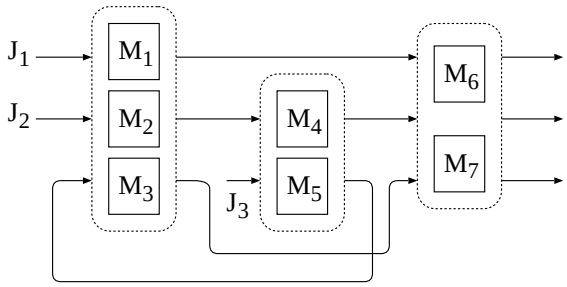
スケジューリングにおいては，次のような制約が課せられる:

(a) 処理機械: 各作業は処理可能機械のいずれか一つにおいて処理される．

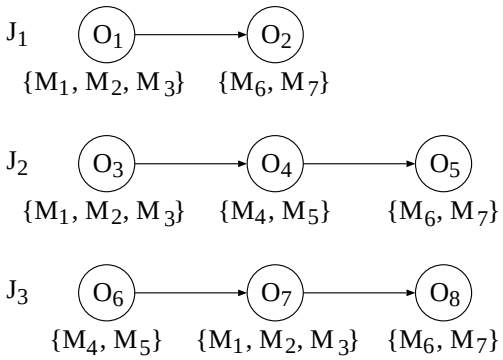
(b) 段取り替え時間: ある機械上で連続して処理される作業の処理タイプが異なる場合，二つの作業の間に段取り替え時間が必要となる．段取り替え時間は前段取りと後段取りの二つの部分からなる．

(c) 次工程時間: ある仕事の二つの連続する作業間において，前の作業の処理が終了してから次の作業の処理を開始するまでの時間 (次工程時間) を，あらかじめ与えられる最大値と最小値の範囲内にしなければならない．

このフレキシブルショップ問題を定式化するにあたって，以下のような記号を導入する．



(a) ショップ・フロア (仕事の流れ)



(b) 作業の先行関係

図 5.1: フレキシブルショップ問題例 (7 機械 3 仕事 8 作業)

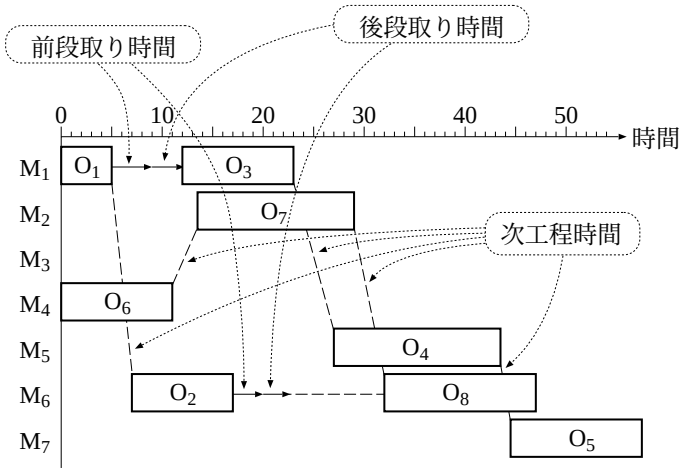


図 5.2: フレキシブルショップ問題 — スケジュール例 (7 機械 3 仕事 8 作業)

まず,

N^{J} : 仕事数,

N^{M}: 機械数,

N : 作業数 $(= N_{N^{\mathrm{J}}})$.

とし，機械 M_i，仕事 J_j および作業 O_k に対する属性を以下のように定義する.

機械属性:

$\mu_{i\delta}$: 処理タイプ δ に対して，機械 M_i 上におけ単位量あたりの処理時間.

仕事属性:

d_j : 納期,

n_j : 作業数.

作業属性:

δ_k: 処理タイプ,

r_k^{O}: 処理量,

s_k^1, s_k^2: 前段取り時間および後段取り時間,

s_k^3, s_k^4: 次工程時間の最小値および最大値,

$\mathcal{M}_k$: 作業 O_k の処理可能機械集合.

スケジュールの評価については様々な指標が存在するが，ここでは最大完了時間 $C_{\max}$ と納期遅れ和 T_{sum} を考え,

$$z_1 = C_{\max} = \max_j t_{N_j}^{\mathrm{F}} = \max_k t_k^{\mathrm{F}}, \tag{5.1}$$

$$z_2 = T_{\mathrm{sum}} = \sum_j \left\{ \max\left(0,\, t_{N_j}^{\mathrm{F}} - d_j\right) \right\}, \tag{5.2}$$

とする．ここで,

t_k^{S} : 作業 O_k の処理開始時刻,

t_k^{F} : 作業 O_k の処理終了時刻,

である．このとき目的関数をこれらの重みつき和で与えられるものとし,

$$z = w_1\, z_1 + w_2\, z_2 \tag{5.3}$$

とする．ここで w_1，w_2 は正の定数である.

以上で定義された問題をフレキシブルショップ問題と呼ぶ．問題例 $(N^{\mathrm{M}} = 7,\ N^{\mathrm{J}} = 3,\ N = 8)$ を図 5.1 に，そのスケジュール例を図 5.2 にそれぞれ示す.

5.2.2　問題の定式化

5.2.1 で記述されたフレキシブルショップ問題を定式化する上で，スケジュールを表わすための 0-1 決定変数 x, y および u を以下のように導入・定義する．

x_{ik} $(i = 1, \ldots, N^{\mathrm{M}};\ k = 1, \ldots, N)$:

O_k が M_i 上で処理されるときに $x_{ik} = 1$，それ以外のとき $x_{ik} = 0$.

$y_{kk'}$ $(k, k' = 1, \ldots, N)$:

O_k が $\mathrm{O}_{k'}$ に先行して処理されるときに$y_{kk'} = 1$，それ以外のとき $y_{kk'} = 0$.

$u_{kk'}$ $(k, k' = 1, \ldots, N)$:

$x_{ik} = x_{ik'}$ $(\forall i)$ のときに $u_{kk'} = 1$，それ以外のとき $u_{kk'} = 0$.

ここで，u は，x と y の決定に依存する従属変数である．また，機械割り付け (x に対応) と開始時刻 (t^{S} に対応) が決定すると完了時刻 (t^{F} に対応) が決定する．よって，独立変数は x, y および t^{S} となる．

なお，式 (5.3) を目的関数とする最適スケジュールは，セミアクティブ・スケジュール*になることがわかっている [1)]．したがって，スケジュールのクラスをセミアクティブに限定した場合，(x, y) はセミアクティブ・スケジュールに 1 対 1 で対応することになる．これより，式 (5.3) の z は x と y の関数となり，フレキシブルショップ問題を以下のように定式化できる．

Minimize

$$z = w_1 z_1 + w_2 z_2 \tag{5.4}$$

subject to

$$\sum_{i=1}^{N^{\mathrm{M}}} x_{ik} = 1 \qquad (k = 1, \ldots, N) \tag{5.5}$$

$$x_{ik} = 0 \qquad (i \notin \mathcal{M}_k;\ k = 1, \ldots, N) \tag{5.6}$$

$$x_{ik} \in \{0, 1\} \qquad (i = 1, \ldots, N^{\mathrm{M}};\ k = 1, \ldots, N) \tag{5.7}$$

— · — · —

$$u_{kk'} \geq x_{ik} + x_{ik'} - 1 \qquad (i = 1, \ldots, N^{\mathrm{M}};\ k, k' = 1, \ldots, N;\ k > k') \tag{5.8}$$

$$u_{kk'} \leq x_{ik} - x_{ik'} + 1 \qquad (i = 1, \ldots, N^{\mathrm{M}};\ k, k' = 1, \ldots, N;\ k > k') \tag{5.9}$$

$$u_{kk'} \leq x_{ik'} - x_{ik} + 1 \qquad (i = 1, \ldots, N^{\mathrm{M}};\ k, k' = 1, \ldots, N;\ k > k') \tag{5.10}$$

$$u_{kk'} \in \{0, 1\} \qquad (k, k' = 1, \ldots, N;\ k > k') \tag{5.11}$$

*作業処理順序を変更することなしに，いずれの作業もそれ以上早く始めることができないようなスケジュールを「セミアクティブ・スケジュール」と呼ぶ．

— · — · —

$$y_{kk'} \in \{\,0,\,1\,\} \qquad (k,\,k' = 1,\ldots,\,N\,;\;k > k') \quad (5.12)$$

— · — · —

$$t_k^{\mathrm{S}} \geq 0 \qquad (k = 1,\ldots,\,N) \quad (5.13)$$

$$t_k^{\mathrm{S}} \geq t_{k-1}^{\mathrm{F}} + s_{k-1}^{3}$$

$$(k = N_{j-1}+2,\ldots,\,N_j\,;\;j = 1,\ldots,\,N^{\mathrm{J}}) \quad (5.14)$$

$$t_k^{\mathrm{S}} \leq t_{k-1}^{\mathrm{F}} + s_{k-1}^{4}$$

$$(k = N_{j-1}+2,\ldots,\,N_j\,;\;j = 1,\ldots,\,N^{\mathrm{J}}) \quad (5.15)$$

$$t_k^{\mathrm{S}} \geq t_{k'}^{\mathrm{F}} + \left(s_{k'}^{2} + s_k^{1}\right)\Delta_{k'k} - M(1 + y_{kk'} - u_{kk'})$$

$$(k,k' = 1,\ldots,\,N\,;\;k > k') \quad (5.16)$$

$$t_{k'}^{\mathrm{S}} \geq t_k^{\mathrm{F}} + \left(s_k^{2} + s_{k'}^{1}\right)\Delta_{kk'} - M(2 - y_{kk'} - u_{kk'})$$

$$(k,k' = 1,\ldots,\,N\,;\;k > k') \quad (5.17)$$

$$t_k^{\mathrm{F}} = t_k^{\mathrm{S}} + \sum_{i=1}^{N^{\mathrm{M}}}\left(r_k^{\mathrm{O}}\,\mu_{i\delta_k}\,x_{ik}\right) \qquad (k = 1,\ldots,\,N) \quad (5.18)$$

— · — · —

$$z_1 \geq t_{N_j}^{\mathrm{F}} \qquad (j = 1,\ldots,\,N^{\mathrm{J}}) \quad (5.19)$$

$$z_2 \geq \sum_{j=1}^{N^{\mathrm{J}}} \max\left(\,0\,,\;t_{N_j}^{\mathrm{F}} - d_j\,\right) \quad (5.20)$$

ここで式 (5.16) および式 (5.17) の $\Delta_{kk'}$ は,

$$\Delta_{kk'} = \begin{cases} 1\,, & \delta_k \neq \delta_{k'} \text{ のとき,} \\ 0\,, & \text{それ以外.} \end{cases} \quad (5.21)$$

で定義される定数であり，M は十分大きな正の定数† である．なお作業の先行関係を表す $y_{kk'}$ それ自身は，割り付け機械とは無関係に (同一機械で処理されるか否かには無関係に) 二つの作業 O_k と $\mathrm{O}_{k'}$ の先行関係を表す変数である．実際の作業の処理順序は，同一機械に割

†式 (5.16) および式 (5.17) は,

$$t_k^{\mathrm{S}} \geq t_{k'}^{\mathrm{F}} + (s_{k'}^{2} + s_k^{1})\Delta_{kk'} \quad \textbf{または} \quad t_{k'}^{\mathrm{S}} \geq t_k^{\mathrm{F}} + (s_k^{2} + s_{k'}^{1})\Delta_{kk'}$$

という,「または」で結ばれた 2 式を「および」で表現した式である．すなわち，式 (5.16) と式 (5.17) は，$y_{kk'}$ と $u_{kk'}$ の値に応じて排他的に意味を持ち，M が十分大きな正の定数であるならば，一方の制約式の M の係数項が正の値となるとき，t_k^{S}，t_k^{F}，s_k^1，s_k^2 の値に関わらずその制約式は満たされ，同時に他方の制約式は (M の係数が 0 となり)M を含む項が 0 となる．

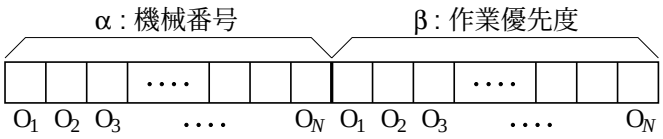


図 5.3: 個体の表現

り付けられた作業について式 (5.16) および式 (5.17) によって定まる．また，式 (5.20) の右辺の “max” は不等式を含むいくつかの線形式に変換可能である．以上より，フレキシブルショップ問題が MIP として定式化されたことになる．

5.3 遺伝アルゴリズムによる解法

5.2.2 で示した MIP への定式化を通して，フレキシブルショップ問題への数理計画法の適用が可能となる．しかしこの方法では膨大な計算量が必要となり，大規模な問題への適用は非現実的である．よって本研究では，進化型計算法の一つである遺伝アルゴリズム (Genetic Algorithm: GA) [9] を適用し，少ない計算量で良質な (準最適な) 解を得ることを考える．

GA やシミュレーテッド・アニーリングのようなメタ戦略を適用する際には，スケジュールを適当な記号列で表現することが有用である．このとき，記号列で表わされる探索空間と解空間との対応の取り方によって，探索効率が大きく影響される．このことに留意して，フレキシブルショップ問題に対する解法を以下のように構成する．

5.3.1 個体の表現

個体を二つの記号列 (α, β) から成るものとする (図 5.3)．すなわち，

α_k: 作業 O_k を割り付ける機械番号，

β_k: 作業 O_k の優先度，

とする．このとき, α_k によって作業 O_k を処理する機械 M_{α_k} (**5.2.2** で示した定式化における x) が決まり，また各機械上で，作業を β_k の非減少順に並べることにより作業順序 (y) が決定する．ここで，α および β の取り得る範囲を，

$$\alpha_k \in \{1, 2, \cdots, N^{\mathrm{M}}\}, \tag{5.22}$$

$$\beta_k \in \{1, 2, \cdots, B\}, \tag{5.23}$$

とする．なお，B は正の定数である．

5.3.2　表現型への変換と適応度の計算方法

5.2.2 で示したように，フレキシブルショップ問題においては，作業の機械割り付け (x) と機械上の作業の順序付け (y) および作業の開始時刻 (t^{S}) によってスケジュールが一意に決定される．各個体をそれぞれ表現型に変換する際，**5.3.1** に述べたように x と y を α および β によって決定する．すなわち，機械割り付け x を α を用いて定めた上で，まず，仕事内の作業間の先行順序制約から作業の開始時刻を前詰めで決定する．次に，コンフリクト (同一機械上での作業の処理時間区間が重なること) が生じる時刻の早い順に，コンフリクトを起こす作業の中から β の値が最大となる作業 O^* 以外の作業の開始時刻を O^* と処理時間が重ならないように後ろへずらす，ということをコンフリクトがなくなるまで繰り返すことにより実行可能スケジュールを得る．

この手続きにおいては，式 (5.15) の制約が考慮されていない．式 (5.15) を考慮して開始時刻を定める場合には，仕事内の処理順において先行する作業の開始時刻を (考慮しない場合に比べて) 遅くすることも必要である．しかし，単純な手続きでは，目的関数を不必要に悪化させることになるので，以下では式 (5.15) の制約を満たさないような決定 (x, y) にペナルティを課すといった方法を考える．すなわち，式 (5.15) の違反量を目的関数に組み込んだ (式 (5.15) の制約を緩和した) 形の目的関数:

$$z' = w_1\, z_1 + w_2\, z_2 + w_3\, z_3 \tag{5.24}$$

ただし，

$$z_3 = \sum_{j=1}^{N^{\mathrm{J}}} \sum_{k=N_{j-1}+2}^{N_j} \max\left\{0,\, t_k^{\mathrm{S}} - \tau_{k-1}\right\} \tag{5.25}$$

$$\tau_k = t_k^{\mathrm{F}} + s_k^4 \tag{5.26}$$

を考慮する．適応度については，式 (5.24) で定義される z' の値をそのまま用いるものとする．

5.3.3　世代交代モデルおよび遺伝演算子

GA の構成にあたっては，個体の評価回数を減らすことを目的として，世代交代モデルに Steady-state GA [46] を用いる．Steady-state GA の適用においては，個体群の中からランダムに二個体を抽出し，遺伝的操作を適用して，新たに一個体を生成する．新たに生成された子個体の適応度を計算し，その値が親個体群の最悪個体の適応度値よりも優れている場合には，最悪個体と子個体を入れ替えた個体群を次世代に残す．この際交叉には多点交叉 (交叉点数: n_{c}) を，突然変異としては，各個体について遺伝子座をランダムに一ヶ所選択し，突然変異確率 p_{m} で他の値に変更するものとする．

5.4 ハイブリッド解法

5.3 で述べた方法では，三つの決定事項:

(a) 作業の機械割り付け (x に対応),

(b) 作業の順序付け (y に対応),

(c) 作業の開始時刻 (t^{S} に対応),

を全て GA による探索の対象としている．しかしながら，解法を設計する際には，解空間の構造を考慮して，GA による探索の対象を見極めることが望ましい．このような観点から，図 5.4 に示すような，定式化を基にした数理的解法と，進化型計算法による解法を組み合わせたハイブリッド解法の構成を考える．具体的には，MIP の一部の決定変数を GA での探索対象とし，残りの変数は分枝限定法によって決定し，スケジュールを作成する．すなわち，GA によって一部の決定変数が固定された部分問題を分枝限定法で評価することにより，GA の探索空間を小さくすることが可能となる (図 5.5)．一方で，問題の規模，決定に対する制約の複雑さによって分枝限定法で解くことのできる (部分) 問題をあらかじめ見積もることは一般に困難である．そこで，ハイブリッド解法として以下では,

方法 H_1：決定事項 (a) を GA で探索する場合,

方法 H_2：決定事項 (b) を GA で探索する場合,

の二つを考える．方法 H_1 では，GA の個体情報によって機械割り付けを決定し，分枝限定法を用いて各機械上での処理順序を定めることにより，スケジュールが作成される．一方，方法 H_2 では，個体情報によって作業の先行関係をあらかじめ固定した上で，分枝限定法を用いて機械割り付けを定めることにより，スケジュールが作成される．このとき，同種類の決定をひとまとまりにして部分問題を構成することにより，部分問題と既存のスケジューリング問題との対応を取ることが可能となり，部分問題に対して，(GA や分枝限定法に限らず) これまでに開発されている解法アプローチの適用が容易である，という利点が考えられる．

5.4.1 個体の表現および適応度の計算方法

方法 H_1 および H_2 において，個体をそれぞれ次のように表現する．

H_1：**5.3.1** で示した記号列 α から構成する．α_k を参照して O_k を M_{α_k} に割り付ける．

H_2：記号列 β から構成する．全ての作業を β_k の非減少順に並べることにより先行関係が決定される．

ハイブリッド解法においては，作業の開始時刻が数理計画法により決定されるため，**5.3.2** に示したような制約式 (5.15) の緩和を考慮する必要はない．よって適応度は，式 (5.4) で定義される z の値を用いるものとする．なお，GA における探索点に対して実行可能解が存在し

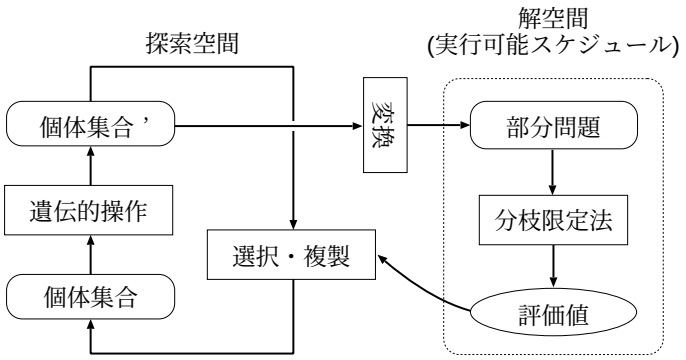


図 5.4: ハイブリッド解法の概要

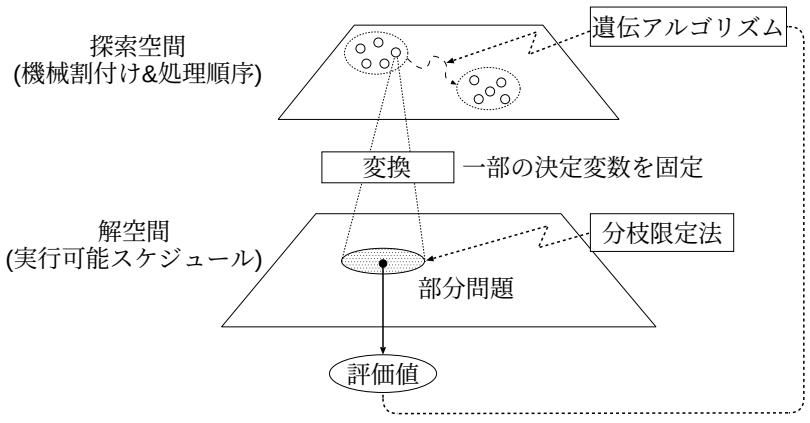


図 5.5: 探索空間と解空間

ない場合には，適応度にペナルティとして十分大きな整数値を与えるものとする．

5.4.2　世代交代モデルおよび遺伝演算子

GA の枠組みにおいて，分枝限定法によるスケジュールの生成は個体を評価する部分に相当する．問題の規模によっては，分枝限定法による計算に多くの計算量が必要となる．そこで，個体の評価回数を減らすことを目的として，世代交代モデルとして Steady-state GA を用いる．この際交叉には多点交叉を，突然変異の実現においては **5.3.3** に示した同様の方法を行う．

各世代における (分枝限定法による) 評価値の算出に際しては，親個体群の評価値に関する情報を分枝限定法の適用に反映させるために，親個体群の最悪個体の評価値を分枝限定法における初期上界値として設定する‡．これにより，分枝限定法の計算量の削減が期待される．

5.5　計算例

フレキシブルショップ問題の例題として，

E_1：12 機械 6 仕事 18 作業問題，

E_2：6 機械 6 仕事 18 作業問題，

E_3：12 機械 9 仕事 45 作業問題，

を用意し，**5.3** および **5.4** で提案した方法を適用する．表 5.1 に例題のパラメータを，表 5.2 に進化型計算およびハイブリッド解法の適用時のパラメータ設定値をそれぞれ示す．各例題に対して，進化型計算およびハイブリッド解法について，初期解をかえて 20 回の試行を行った．ハイブリッド解法において，分枝限定法には商用の数理計画パッケージ (CPLEX) [47] を用いた．

最良スケジュールの評価値の最小，最大，平均および標準偏差を表 5.3 に示す．なお，表には分枝限定法 (CPLEX) を用いて得られた結果も合わせて示しておく．さらに，表 5.4 に平均計算時間 (UltraSPARC III 750MHz を使用時) を示す．なお例題 E_3 については，個体の評価に膨大な計算時間が必要となり方法 H_2 の適用による結果は得られていない．また，分枝限定法に関しては，(最適) 解を得るには同様に，膨大な計算時間が必要となるため，$1,000,000$ ノードを生成した時点で計算を打ち切った場合の結果を示してある．

表 5.3 および表 5.4 より，以下のことが確認される．まず，進化型計算による解法について，

(a) 例題 E_1，E_2 に対して，最適に近い (実行可能) スケジュールが短時間で得られている．一方で，全試行を通して安定して良い解が得られているものは少ない．これは，n_p(個

‡部分問題の最適値が上界の設定値を上回る場合には，実行可能解は存在しない．そのような部分問題に対応する個体にはペナルティを与えるものとする．

表 5.1: 例題のパラメータ

重み (w_1, w_2, w_3)	(0.25, 0.25, 0.50)
定数 M	289 (for E_1), 649 (for E_2),
	1621 (for E_3)
優先度最大値 B	18 (for E_1, E_2),
	45 (for E_3)

表 5.2: 解法のパラメータ設定

個体数 n_{p}	50
世代数 n_{g}	1000
交叉点の数 n_{c}	2
突然変異率 p_{m}	0.1

体数) あるいは n_{g}(世代数) の設定値に起因しているものと思われる．

(b) 例題 E_3 に対しては実行可能解が得られていない．重み (w_1, w_2, w_3) の設定にも起因するものであると考えられるが，GA 単体による探索が本質的に難しいことを示唆するものである，と考えられる．

次に，ハイブリッド解法について，

(c) 全ての例題に対して，方法 H_1 および H_2 ともに進化型計算よりも良い解を安定して得ている．特に H_2 は例題 E_1 と E_2 に対して最適解を獲得している．H_2 の場合，本論文で示した定式化において GA により固定される制約式が式 (5.12)，式 (5.16) および式 (5.17) の一部にすぎず，分枝限定法により評価する部分問題の規模が比較的大きいためであると考えられる．

(d) 計算時間については，進化型計算による解法よりも長くなっているものの，問題の規模が大きくなるにつれて，数理的解法よりもより短くなっていることが分かる．特に，方法 H_2 には長い計算時間を要しているが，これは分枝限定法が評価する部分問題の規模が小さくないことや，GA により決定された順序付けが，実行可能解とはならない場合が少なくないこと，が原因として考えられる．

考察 (c) および (d) より方法 H_2 は，例題 E_1，E_2 および E_3 に対して，最適性の高い解を安定して得ることができるものの長い計算時間を要する，厳密解法に近い手法であることが分かる．一方，方法 H_1 は H_2 よりも解の質が低くなる場合があるものの，GA により選ばれたいずれの探索点も実行可能スケジュールと対応していることから，探索効率が良く，また，GA により式 (5.6)∼(5.11) と式 (5.16)，(5.17) の一部が固定されることから，BAB の求解に要する時間も比較的短い．これより方法 H_1 は，解の良さと計算時間のバランスがとれた

表 5.3: 実験結果 (目的関数値)

(a) E_1

方法	目的関数値			
	最小値*	最大値*	平均値	標準偏差
GA	17.75 (63, 8, 0)	21.50 (70, 12, 2)	20.31	0.84
ハイブリッド (H_1)	17.75 (63, 8, −)	20.00 (64, 16, −)	18.85	0.63
ハイブリッド (H_2)	17.75 (63, 8, −)	17.75 (63, 8, −)	17.75	0.00
CPLEX	17.75 (63, 8, −)			

(b) E_2

方法	目的関数値			
	最小値*	最大値*	平均値	標準偏差
GA	32.75 (78, 53, 0)	37.25 (84, 63, 1)	35.59	1.24
ハイブリッド (H_1)	29.75 (74, 45, −)	32.00 (75, 53, −)	31.04	0.58
ハイブリッド (H_2)	29.50 (74, 44, −)	31.00 (78, 46, −)	29.93	0.42
CPLEX	29.50 (74, 44, −)			

(c) E_3

方法	目的関数値			
	最小値*	最大値*	平均値	標準偏差
GA	68.75 (189, 42, 22)	91.75 (204, 101, 31)	82.08	6.18
ハイブリッド (H_1)	45.75 (170, 13, −)	48.50 (181, 13, −)	47.38	0.87
CPLEX**	67.75 (184, 87, −)			

* 括弧内に 3 種類の評価値 (z_1, z_2, z_3) を示す.

** 分枝限定法の適用において，1,000,000 ノードを生成した時点で
計算を打ち切った際の結果を示す.

表 5.4: 実験結果 (計算時間)

例題	方法	計算時間 [秒]*
E_1	GA (平均値)	0.11
	ハイブリッド (H_1, 平均値)	41.18
	ハイブリッド (H_2, 平均値)	2206.28
	CPLEX	41.72
E_2	GA (平均値)	0.12
	ハイブリッド (H_1, 平均値)	155.46
	ハイブリッド (H_2, 平均値)	19659.18
	CPLEX	8411.13
E_3	GA (平均値)	0.55
	ハイブリッド (H_1, 平均値)	18769.59
	CPLEX**	23626.08

* UltraSPARC III (750MHz) 使用時

** 分枝限定法の適用において，1,000,000 ノードを生成した時点で計算を打ち切った際の結果を示す.

表 5.5: 各方法の特徴

方法	解の質	安定性	計算時間
BAB	◎	◎	×
H_2	○	◎	×
H_1	△	○	△
GA	×	×	◎

方法であることが分かる．

最後に，以上の考察をもとに，各手法の特徴，すなわち解の良さ，試行毎の安定性および計算時間について比較した結果を表5.5 にまとめる．

なお，本研究で提案した方法は，ハイブリッド・アプローチの一つの枠組みに過ぎない．より高い次元で解の良さと計算時間をバランスさせるためには，どのような形でハイブリッド化を行うかについての検討，さらにはハイブリッド化を考慮した定式化の見直しも必要であると思われる．例えば，作業の機械割り付けと順序付けのそれぞれの一部の決定を GA により固定するような，方法 H_1 と H_2 の中間的なハイブリッドの枠組みについての検討が今後の課題として挙げられる．

5.6　結言

本章では，まずフレキシブルショップ問題を対象として，数理計画の枠組みで定式化を行った．さらに，定式化に基づいて，進化型計算および数理計画と進化型計算のハイブリッド解法を示し，特にハイブリッド解法においては，進化型計算が探索する決定の種類に応じて二種類の方法を提案し，計算機実験を通してそれらの有効性・可能性を検討した．その結果，進化型計算が探索する決定変数および分枝限定法が定める決定変数の種類に応じて，得られる解の良さと計算時間にトレード・オフを有するような性質が確認された．

今後の課題として，さらなる計算機実験による評価や，定式化の見直しを含めたハイブリッド化の枠組みの検討などが挙げられる．

第 6 章

フレキシブルショップ問題に対するルール獲得法

6.1 緒言

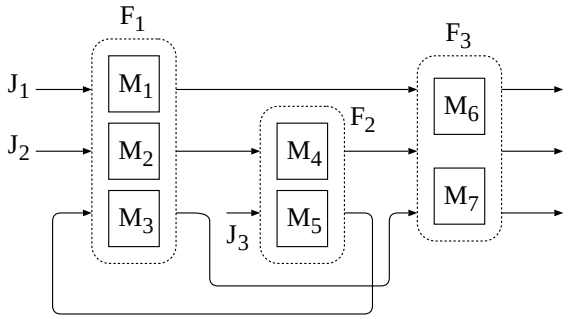
本章では，第 3 章で示したスケジューリング・ルール獲得方式について，現実の様々な問題に対して広範な応用の可能性を有するフレキシブルショップ問題への適用方法を提案する [48, 49]．このフレキシブルショップ問題は，ジョブショップ問題において，各ショップに複数の機械が並列に存在するような問題 [2] であり，本研究では，さらに段取り替えに関する制約も考慮する．この問題においては，ディスパッチング時に割り付け設備の選択のみを繰り返せばよいといった在庫設備選択問題 (第 4 章) とは異なり，仕事を割り付ける機械の選択に加えて，仕事を処理する順序という時間軸上の広がりを持つような決定を有する．

Pitt アプローチによるルール獲得時には，第 5 章での検討を踏まえて，シミュレータを構築することを考慮する．すなわち，機械割り付けをヒューリスティックに決定した上で，機械上の作業の処理順序をルールを用いて決定する．またルール構成として，決定の際に用いられる問題の属性値の重みを後件部とするものを提案する．第 4 章では，在庫設備選択問題に対して，後件部に決定事項を直接記述するようなルール構成を考慮していたが，本研究で取り扱う問題において，このような単純なルール構成では効果的なルールの獲得が困難であると予想される．本章で示すルール構成はこの点に留意したものであり，これによって，様々な状況に対応したルールを実現することができると考えられる．

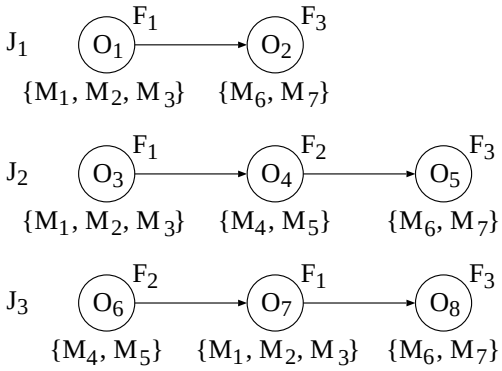
最後に，例題を用いた計算機実験を通して，獲得されたルール集合を適用して得られる解の良さ，ルール集合の内容，ならびに汎用性について検討し，提案手法の有効性を確認する．

6.2 問題の記述

本章で対象とするフレキシブルショップ問題を以下のように定義する．N^{J} 個の仕事 J_j $(j = 1, \cdots, N^{\mathrm{J}})$ を N^{M} 台の機械 M_i $(i = 1, \cdots, N^{\mathrm{M}})$ で処理するものとする．仕事は N^{F} 個の工程 F_h $(h = 1, \cdots, N^{\mathrm{F}})$ における処理 (作業) を経て完了し，工程 F_h は n_h 台の並列機械 M_i $(i = N_{h-1} + 1, \cdots, N_h\ ;\ N_h = \sum_{t=1}^{h} n_t;\ N_0 = 0)$ により構成される．工程を経る順序は仕事毎に異なり，仕事 J_j は n_j 個の作業 O_k $(k = N_{j-1}+1, \cdots, N_j;\ N_j = \sum_{\ell=1}^{j} n_\ell;\ N_0 = 0)$



(a) ショップ・フロア (仕事の流れ)



(b) 作業の先行関係

図 6.1: フレキシブルショップ問題例 (7 機械 3 仕事 8 作業)

図 5.1 のモデルに対して，このモデルでは工程 F_h を陽に取り扱っている．

で構成される．作業は $\mathrm{O}_k \to \mathrm{O}_{k+1}$ $(k = N_{j-1}+1, \cdots, N_j - 1)$ の順に処理されるものとする．各作業 O_k $(k = 1, \cdots, N^{\mathrm{O}})$ に対して，納期 d_k，処理可能機械集合 $\mathcal{M}_k$，および作業タイプ δ_k が存在する．機械 M_i における作業 O_k の処理時間 p_{ik} は，機械の処理能力 μ_i と作業タイプ δ_k の組み合せによって定まるものとする．

スケジューリングにおいては，次のような制約が存在する．

(a) 処理機械: 各作業は処理可能機械のいずれか一つにおいて処理される．

(b) 段取り替え時間: ある機械上で連続して処理される作業の処理タイプが異なる場合，二つの作業の間に段取り替え時間が必要となる．段取り替え時間は前段取りと後段取りの二つの部分からなる．

問題をモデル化するにあたり，まず，

N^{F} : 工程数,

N^{M}: 機械数,

N^{J} : 仕事数,

N ：作業数 $(= N_{N^{\mathrm{J}}})$.

とし，工程 F_h，機械 M_i，仕事 J_j および作業 O_k に対する属性を以下のように定義する．

(A) 工程属性

n_h：機械数,

(B) 機械属性

$\mu_{i\delta}$：処理タイプ δ に対して，機械 M_i 上における単位量あたりの処理時間.

(C) 仕事属性

d_j：納期,

n_j：作業数.

(D) 作業属性

δ_k: タイプ,

r_k: 処理量,

s_k^1, s_k^2: 前段取り時間および後段取り時間,

$\mathcal{M}_k$: 作業 O_k の処理可能機械集合.

p_{ik}: 処理時間 $(p_{ik} = \mu_{i\delta_k} r_k)$.

スケジュールの評価については様々な指標が存在するが，ここでは最大完了時間 $C_{\max}$ と納期遅れ和 T_{sum} を考え，

$$z_1 = C_{\max} = \max_j t^{\mathrm{F}}_{N_j} = \max_k t^{\mathrm{F}}_k \,, \tag{6.1}$$

$$z_2 = T_{\mathrm{sum}} = \sum_j \left\{ \max\left(0,\, t^{\mathrm{F}}_{N_j} - d_j\right) \right\} \,, \tag{6.2}$$

とする．ここで,

t^{S}_k：作業 O_k の処理開始時刻,

t^{F}_k：作業 O_k の処理終了時刻,

である．このとき，目的関数はこれらの重みつき和で与えられるものとし,

$$z = w_1\, z_1 + w_2\, z_2 \tag{6.3}$$

とする．ここで w_1 および w_2 は正の定数である．

このとき問題は，上述の制約を満たしながら評価値が最小となるようなスケジュール，すなわち

(1) 作業の機械割り付け,

(2) 機械上の作業の処理順序,

(3) 作業の開始時刻,

(4) 作業の終了時刻,

を決定することであると定義される．このとき，(1) の作業の機械割り付けを決めることにより作業の処理時間属性が決定する．問題例 ($N^{\mathrm{M}} = 7$, $N^{\mathrm{J}} = 3$, $N = 8$) を図 6.1 に，それぞれ示す.

ここで取り上げた目的関数の下での最適スケジュールは，一般にアクティブ・スケジュールであることが知られている [1)]．スケジュールのクラスをアクティブに限定した場合には，決定事項 (1) および (2) によってスケジュールが一意に定められる.

スケジューリングの過程において，意思決定のタイミングに注目すると，本研究では事前に仕事に関する情報が与えられない状況下におけるスケジューリングを考慮している．よって，事前にスケジュールを準備することができず，仕事が到着した時点でその仕事のスケジュールを決定することになる.

6.3　優先度に基づくスケジューリング

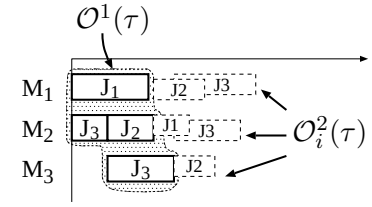
フレキシブルショップ問題に対し，ディスパッチング・ルールを用いたスケジューリングの方法について考える．**6.2** で示したように，この問題に対してはアクティブ・スケジュールのクラスを考えればよく，以下では，上工程より順に作業を割り付けるフォワードスケジューリングを前提とする [50)]．具体的には，いずれかの機械で仕事が完了する時刻を τ とし，処理開始可能作業を割り付けたとして最早に完了する機械を割り付け対象の機械 $\mathrm{M}_{i^*}(\tau)$ に選ぶ．割り付け作業 $\mathrm{O}_{\hat{k}}$ については，$\mathrm{M}_{i^*}(\tau)$ 上で時刻 τ において，(仕事内の) 直前の作業が処理中であるかあるいは処理済みであるような作業の集合 $\mathcal{O}^3_{i^*}(\tau)$ の中から優先度に基づいて決定する．具体的なスケジューリングの手続きを **6.3.1** に示す．また，図 6.2 にその一例を示す.

6.3.1　スケジューリングの手順

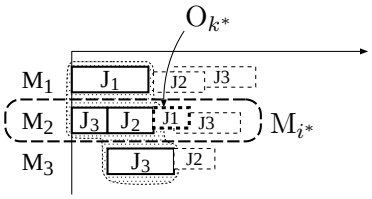
1° (初期化)
ディスパッチングのタイミングを $\tau = 0$ ，割り付け済み作業集合を $\mathcal{O}^1 = \emptyset$ とする.

2° (割り付け機械の決定)

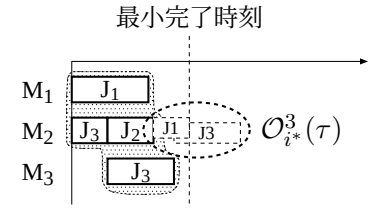
(a) 機械 M_i に対して τ における開始可能作業集合 $\mathcal{O}^2_i(\tau) = \emptyset$ とする．作業 O_k が仕事内の処理順において先頭 (ある j について $k-1 = N_j$) であるか，あるいは直前の作業が割り付け済み ($\mathrm{O}_{k-1} \in \mathcal{O}^1$ かつ，すべての j について $k-1 \neq N_j$) であれば，O_k を割り付け可能な全ての機



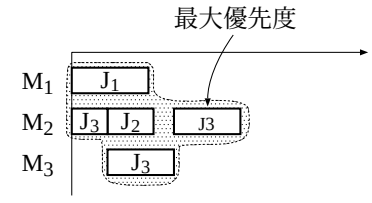
2° (a), (b)



2° (c)



3°



4°

図 6.2: スケジューリング手続き

械 ($\mathrm{M}_i \in \mathcal{M}_k$) に対し $\mathcal{O}_i^2(\tau) = \mathcal{O}_i^2(\tau) \cup \{\mathrm{O}_k\}$ とする．いずれの i に対して $\mathcal{O}_i^2(\tau) = \emptyset$ ならば，$\tau = \tau + 1$ として 2° へ．

(b) $\mathrm{O}_k \in \mathcal{O}_i^2(\tau)$ に対し，最早の開始時刻:

$$t_k^{\mathrm{S}'} = \begin{cases} \max(t_{k-1}^{\mathrm{S}} + p_{i,k-1}, \\ \quad t_{\ell_i}^{\mathrm{S}'} + p_{i\ell_i} + s_{\ell_i k}\Delta_{\ell_i k}), \\ \quad \text{if} \quad k \neq N_{j-1},\ j = 1, \cdots, N^{\mathrm{J}}, \\ \max(0,\ t_{\ell_i}^{\mathrm{S}'} + p_{i\ell_i} + s_{\ell_i k}\Delta_{\ell_i k}), \\ \quad \text{otherwise}, \end{cases}$$

を求める (O_{ℓ_i} は機械 M_i の最後尾に割り付けられた作業; $\Delta_{kk'}$ は $\delta_k \neq \delta_{k'}$ のとき 1，そうでないとき 0 であるような定数).

(c) $\bigcup_i \mathcal{O}_i^2(\tau)$ の中で完了時間 ($t_k^{\mathrm{S}'} + p_{ik}$) が最小となる作業ならびにその処理機械を求め，それぞれ O_{k^*} および M_{i^*} とする．

3° (割り付け作業候補の決定)
$\mathcal{O}_{i^*}^2(\tau)$ の中で $t_k^{\mathrm{S}'} < t_{k^*}^{\mathrm{S}'} + p_{i^*k^*}$ を満たす作業の集合を $\mathcal{O}_{i^*}^3(\tau)$ ($\subseteq \mathcal{O}_{i^*}^2(\tau)$) とする．

4° (作業の機械割り付け)
全ての $\mathrm{O}_k \in \mathcal{O}_{i^*}^3(\tau)$ に対して優先度を求め，優先度の最も高い作業を $\mathrm{O}_{\hat{k}}$ とし，$\mathrm{O}_{\hat{k}}$ の開始時刻を $t_{\hat{k}}^{\mathrm{S}} = t_{\hat{k}}^{\mathrm{S}'}$ とする．$\mathcal{O}^1 = \mathcal{O}^1 \cap \{\mathrm{O}_{\hat{k}}\}$ とする．

5° (終了判定)
$|\mathcal{O}^1| = N$ なら終了．そうでなければ 2° へ．

6.3.2　優先度の計算方法

6.3.1 の 4° における割り付け作業の決定のための優先度 ϕ_{uk} を，次式を用いて算出する:

$$\phi_{uk} = \sum_{\ell=1}^{n_{\mathrm{a}}} w_{u\ell} a_{k\ell}. \tag{6.4}$$

ただし $a_{k\ell}$ ($k = 1, \cdots, N;\ \ell = 1, \cdots, n_{\mathrm{a}}$) は，作業 O_k に付随する属性 (優先度属性) 値を，$w_{u\ell}$ は状態 (生産システムがとり得る状態 (空間) $\mathcal{S}$ をあらかじめ分割したもの) が $\mathcal{S}_u$ である場合の優先度属性値の重み係数をそれぞれ表す．

優先度属性としては，作業に付随する属性として次の 4 種類 ($n_{\mathrm{a}} = 4$) を考える:

(A) 処理時間 (段取り替え時間を含む) $\hat{p}_{ik}$,

(B) 処理可能機械台数 $|\mathcal{M}_k|$,

(C) 納期余裕時間 u_k *,

* $u_k = d_j - t_k^{\mathrm{F}*}$, $(k = N_{j-1} + 1, \cdots, N_j)$．ただし $t_k^{\mathrm{F}*}$ は (O_k に後続する) 未割り付けの作業を，作業間の先行制約に基づいて (処理可能機械集合の中で処理時間が最大となる機械に) 割り付けたときの仕事 J_j の完了時刻を表す．

図 6.3: ルールの構成

(D) 残りの工程数 n_k^{L}.

また，生産システム全体の状態を表す指標 (状態指標) としては，

(1) 仕事の完成率 C^{J},

(2) 最大完了時間 $C_{\max}$,

(3) 納期余裕時間平均 D_{avg},

を考え，この三つの指標値の直積空間 $\mathcal{S}$ により状態空間を定義する．

部分状態 $\mathcal{S}_u$ に対する重み $w_{u\ell}$ の決定にルールを用いる方法を考える．ルールは前件部 ($\langle$condition$\rangle$) と後件部 ($\langle$action$\rangle$) からなる一般的な構成:

$$\text{if } \langle\text{condition}\rangle \text{ then } \langle\text{action}\rangle \tag{6.5}$$

とし，前件部に三つの離散化された指標値からなる数値列が記述される．これより，ある一つの部分状態 (空間) $\mathcal{S}_u$ が指定される．また後件部には，各 ℓ $(= 1, \cdots, n_{\mathrm{a}})$ について前件部で表される部分状態 $\mathcal{S}_u$ における優先度属性の重み $w_{u\ell}$ をそれぞれ記述する．ここで各重みは $\{-n_\ell^{\mathrm{w}}, \cdots, n_\ell^{\mathrm{w}}\}$ のいずれかの整数値を取るものとする．図 6.3 にルール構成の一例を示す．以上で示したルール構成を「ルール構成 (a)」と呼ぶ.

6.3.1 で示したスケジューリングの手続きにおいて，優先度計算の対象となる作業はいずれも同一機械 (工程) 上で処理される．このとき，工程毎に異なるルール集合を用意することにより，きめ細かい意思決定を行うことができると考えられる．そこで，ルールの前件部が指定する状態空間として，上で示した状態指標に

(4) 作業が処理される工程番号 f_h

を加えた四つの指標値の直積空間から定義される新たな状態空間を考慮する．このようなルール構成を「ルール構成 (b)」と呼ぶ.

スケジューリングを行うにあたっては，n_{r} 個のルールから成る集合 $\{\mathrm{r}_1, \mathrm{r}_2, \cdots, \mathrm{r}_{n_{\mathrm{r}}}\}$ を用意しておく．各ディスパッチングのタイミングにおいて，生産システムの状態 (入力状態) と各ルールの前件部が比較され，入力状態が属する部分状態を前件部として持つようなルールが適用される．ここで，ルール集合が状態空間全体 $(\bigcup_u \mathcal{S}_u)$ に対応することを仮定しない

(各々の $\mathcal{S}_u$ に対して，対応する前件部を持つルールがルール集合中に存在するとは限らない)，ということに注意されたい．つまり，ルール集合は状態空間の一部を保持する．

なお，入力状態が属する部分状態を前件部として持つルールが無い場合には，ルール構成 (a) を用いる場合，入力状態 (四次元空間内の一点に対応) と，ルールの前件部が持つ部分状態を代表する点 (重心) とのユークリッド距離を計算し，この距離が最小となるルールを適用するものとする．これにより，状態空間の規模とは無関係にルール集合を構成することが可能となる．ルール構成 (b) を用いる場合，入力状態と状態指標 (4) が一致するルール群の中から，ルールの前件部が持つ部分状態を代表する点 (重心) とのユークリッド距離を計算し，この距離が最小となるルールを適用するものとする．この手続きは，前述の「工程毎に異なるルール集合を用意する」という考えに基づいている．

6.4　遺伝的機械学習によるルール学習方式

6.3 で示したルールを自動的に獲得・調整するために遺伝アルゴリズム (GA) [46] を用いたルール学習方式である遺伝的機械学習 (GBML) [31] を適用する．GBML は大きく二つのアプローチ，すなわち Michigan アプローチと Pitt アプローチに分類されるが，Michigan アプローチでは個体 (ルール) の適応度の算出にクレジット分配機構を用いており，計算機上に実現することは容易ではない．これに対して，Pitt アプローチでは予めルール集合を生成・獲得しておくといった枠組みになるものの，Michigan アプローチのようなクレジット分配システムを必要としないために実装が容易である．そこで，本研究では Pitt アプローチを用いた解法構成を考える．

GBML の適用にあたっては，ルール集合を一個体とみなし，各ルール集合を用いて得られる結果に対する評価値を適応度として GA を適用する Pitt アプローチの構成を考える．フレキシブル・ショップ問題に対して Pitt アプローチを適用する枠組みを図 6.4 に示す．

6.4.1　遺伝アルゴリズムの構成

GBML の適用に際しては，GA における個体の遺伝子表現，適応度の計算，世代交代モデルおよび遺伝演算子を具体化する必要がある．

(1) 個体の遺伝子表現

Pitt アプローチでは，ルール集合を GA における個体として取り扱う．そこでルール $\mathrm{r}_\ell\,(\ell = 1, \cdots, n_\mathrm{r})$ を複数個並べて表わされるルール集合:

$$\{\mathrm{r}_1, \mathrm{r}_2, \cdots, \mathrm{r}_{n_\mathrm{r}}\}$$

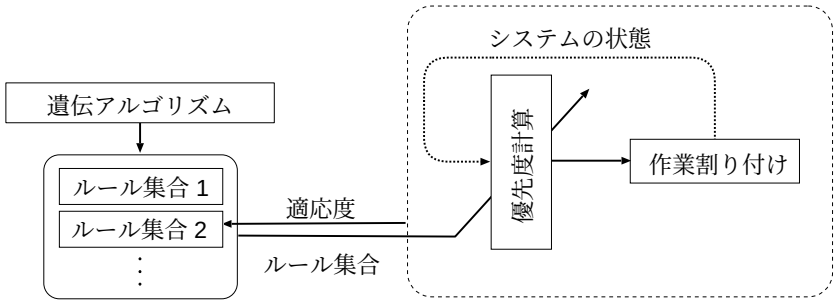


図 6.4: Pitt アプローチによるルール学習の枠組み

を個体の遺伝子表現とする.

(2) 適応度計算

個体の適応度はルール集合の評価値に対応している．本研究ではスケジュールの評価値, すなわち式 (6.3) で表される値をルール集合の評価値とする．なお，**6.3.2** で述べたように, 優先度計算において適用するルールが見つからない場合には，ペナルティとして十分大きな値を評価値とする.

(3) 世代交代モデルと遺伝演算子

GA の適用において，各世代における個体すなわちルール集合の多様性の維持の観点から, 世代交代モデルとして Steady-state GA [46] を用いる．Steady-state GA の適用においては, 個体群の中からランダムに二個体を抽出し，遺伝的操作を適用して，新たに一個体を生成する．新たに生成された子個体の適応度を計算し，その値が親個体群の最悪個体の適応度値よりも優れている場合には，最悪個体と子個体を入れ替えた個体群を次世代に残す．このとき, 遺伝的操作として次の二つを考える:

(a) 交叉: ランダムにペアリングされた二個体に対して多点交叉を行い，子個体を一つ生成する．この操作により，いずれかの親個体 (ルール集合) のルールを含む子個体が生成される.

(b) 突然変異: 個体の遺伝子座をランダムに一ヶ所選択し，突然変異確率 p_{m} で他の値に (前件部に関しては (突然変異を加えた後に) いずれかの部分状態 $\mathcal{S}_u$ に対応するように，また後件部に関しては $\{-n_\ell^{\mathrm{w}}, \cdots, n_\ell^{\mathrm{w}}\}$ の範囲に含まれるように) 変更する．この操作により，新たなルールを含む個体 (ルール集合) が獲得される.

表 6.1: 例題のパラメータ

	N^{M}	N^{J}	N	$(w_1,\, w_2)$
$\mathcal{E}^{\mathrm{A}}$	6	6	18	$(0.5,\, 0,5)$
$\mathcal{E}^{\mathrm{B}}$	12	9	45	$(0.5,\, 0,5)$
$\mathcal{E}^{\mathrm{C}}$	11	70	219	$(0.5,\, 0,5)$

表 6.2: GA のパラメータ

個体数 n_{p}	100
世代数 n_{g}	5000
突然変異率 p_{m}	0.1
ルール数 n_{r}	100

6.5　計算機実験

表6.1 に示すような 3 種類の問題のクラス ($\mathcal{E}^{\mathrm{A}}$，$\mathcal{E}^{\mathrm{B}}$ および $\mathcal{E}^{\mathrm{C}}$) を考え，各クラスに対してランダムに 20 個の例題 ($\mathrm{E}_i^{\mathrm{A}}$，$\mathrm{E}_i^{\mathrm{B}}$ および $\mathrm{E}_i^{\mathrm{C}}$ $(i = 1, \cdots, 20)$) を作成した．**6.3** および **6.4** で述べた解法をこれらの例題に適用し，(1) 特定の例題に対してルール学習を行った場合に，どのようなスケジュールが得られるか (ルールの学習状況)，(2) (1) で得られたルール集合の内容，および (3) ルール学習により獲得されたルール集合を (ルール学習に用いた例題とは異なる) 新たな例題に適用した場合に，どのようなスケジュールが得られるか (ルール集合の汎用性)，について検討する．

6.5.1　ルールの学習状況について

各例題に対して，ルール構成 (a) およびルール構成 (b) を用いた場合 (方法 GBML(a) および GBML(b) と呼ぶ) についてそれぞれ，乱数系列を変えて 10 回の試行を行った．ルールの設定に関して，状態の分割数を $|\mathcal{S}_u| = 10^3$ (GBML(a) の場合)，$|\mathcal{S}_u| = 10^3 \times N^{\mathrm{F}}$ (GBML(b) の場合) のように設定し，ルールの後件部の取り得る範囲を $n_\ell^{\mathrm{w}} = 4$ $(\ell = 1, \cdots, 4)$ とした．さらに，GBML を適用する際のパラメータを表6.2 のように設定した．GBML によるルール学習により，獲得されたルール集合がどのようなスケジュールを生成するかをみるために，GBML により獲得された最良スケジュールの評価値 f_i^{GBML} と，GA によりスケジュールを直接探索する方法 (方法 GA) を用いて得られた最良スケジュールの評価値 f_i^{GA} との比 $\rho_i\, (= f_i^{\mathrm{GBML}}/f_i^{\mathrm{GA}})$ を求めた．方法 GA の適用にあたっては，スケジューリング方法における作業 O_k の優先度 ϕ_k を遺伝子表現:

$$\{\phi_1, \phi_2, \cdots, \phi_N\}$$

として用いる．表6.3に各例題クラスに対する(20個の例題の)ρ_iの平均，分散および標準偏差を示す．表6.3において，例題クラス$\mathcal{E}^{\mathrm{A}}$については，商用の数理計画パッケージCPLEX[47]を用いて得られた最適スケジュールの評価値f_i^{CPLEX}との比ρ_i^* $(= f_i^{\mathrm{GBML}}/f_i^{\mathrm{CPLEX}})$の値を合わせて示しておく．表6.4には，方法GBML(a)およびGBML(b)のルール獲得に要した計算時間を示す．

表6.3より，方法GBML(a)およびGBML(b)により求められるルール集合は，

- 例題クラス$\mathcal{E}^{\mathrm{A}}$に対して，最適もしくは最適に近いスケジュールを生成する，
- 三つの例題クラスに対して，方法GAにより求められるスケジュールに対してほぼ同程度の(例題によっては優れた)スケジュールを生成する

ことが確認される．方法GBMLと方法GAでは，同じ決定(作業の優先度)に対してそれぞれ異なる探索空間が定められていることから，方法GBMLで用いたルール集合による探索が，決定そのものを探索空間とするものと同程度の性能を有することが分かる．また，方法GBML(a)とGBML(b)の結果を比較すると，ほぼ同程度のスケジュールを生成するようなルール集合が獲得されていることが分かる．

6.5.2 得られたルール集合について

ルール学習により得られた最良のルール集合について，計画期間中に優先度計算で用いられたルールの数(20個の例題の平均値)を表6.5に示す．なお，仮にディスパッチングの各タイミングにおいて互いに異なるルールが適用された場合，計画期間中に適用されたルールの数は作業数Nに等しくなる．さらに，スケジューリングの過程におけるルールの適用状況を見るために，例題$\mathrm{E}_1^{\mathrm{C}}$に対する各ディスパッチングのタイミングでの作業の優先度属性の重み(方法GBML(a)を用いた場合)を図6.5に示す．

表6.5より，方法GBML(a)を用いた場合，

- 三つの問題に対して，優先度計算に用いられるルールの数は少ない

ことが分かる．最良ルール集合により，最適もしくは準最適なスケジュールが得られていること(表6.3)および，提案方法により汎用的なルール集合が得られていること(**6.5.3**に後述)を考え合わせると，最適/準最適なスケジュールを得るための汎用的なルールが提案方法により絞り込まれていると考えられる．一方，方法GBML(b)を用いた場合，

- 方法GBML(a)を用いた場合よりも適用されたルールの数は多く，

工程毎に異なるルール集合を用意することにより，より多様なルールが適用されていることが分かる．

ルールの適用状況について見てみると，図6.5より，

- 作業の優先度属性の重みが時間とともに変化している

表 6.3: 計算結果 (評価値)

(1) 方法 GBML(a)

例題	比 ρ (ρ^*)			
	最小	最大	平均	標準偏差
$\mathcal{E}^{\mathrm{A}}$	0.96	1.13	1.01	0.03
	(1.00)	(1.18)	(1.04)	(0.05)
$\mathcal{E}^{\mathrm{B}}$	0.97	1.03	0.99	0.01
$\mathcal{E}^{\mathrm{C}}$	0.96	1.04	1.00	0.02

(2) 方法 GBML(b)

例題	比 ρ (ρ^*)			
	最小	最大	平均	標準偏差
$\mathcal{E}^{\mathrm{A}}$	0.95	1.13	1.01	0.03
	(1.00)	(1.18)	(1.03)	(0.05)
$\mathcal{E}^{\mathrm{B}}$	0.97	1.02	0.99	0.01
$\mathcal{E}^{\mathrm{C}}$	0.96	1.04	1.00	0.02

表 6.4: ルール学習に要した計算時間

(1) 方法 GBML(a)

例題	計算時間 [秒]
$\mathcal{E}^{\mathrm{A}}$	1.4
$\mathcal{E}^{\mathrm{B}}$	5.7
$\mathcal{E}^{\mathrm{C}}$	93.2

(2) 方法 GBML(b)

例題	計算時間 [秒]
$\mathcal{E}^{\mathrm{A}}$	2.1
$\mathcal{E}^{\mathrm{B}}$	10.9
$\mathcal{E}^{\mathrm{C}}$	865.7

表 6.5: 適用されたルールの数

(1) 方法 GBML(a)

例題	適用ルール数
$\mathcal{E}^{\mathrm{A}}$	9.0
$\mathcal{E}^{\mathrm{B}}$	7.2
$\mathcal{E}^{\mathrm{C}}$	7.9

(2) 方法 GBML(b)

例題	適用ルール数
$\mathcal{E}^{\mathrm{A}}$	12.0
$\mathcal{E}^{\mathrm{B}}$	14.2
$\mathcal{E}^{\mathrm{C}}$	18.1

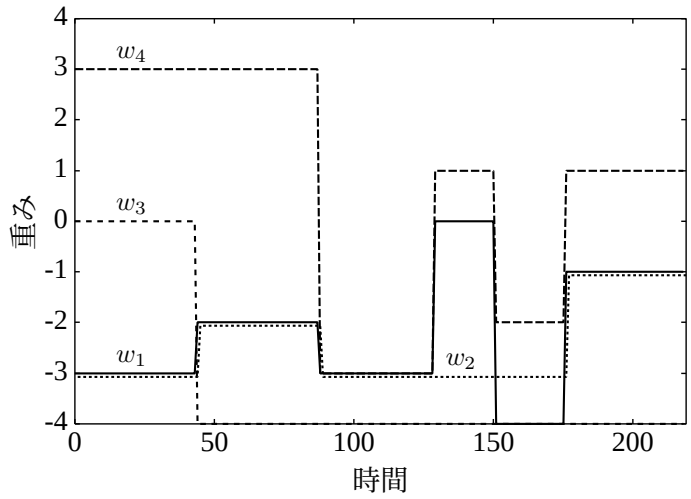


図 6.5: 優先度属性値の重み

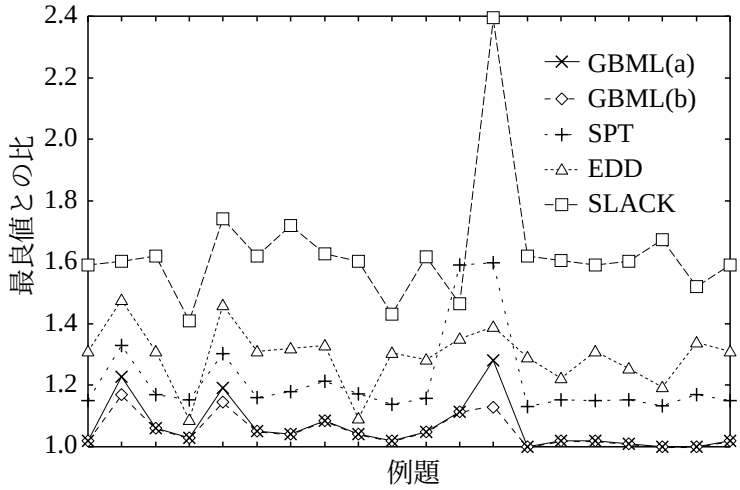
ことが分かる．すなわち，システムの状態に応じて各重みの値および (それに伴う) 重み同士の相対的な関係が変化しており，ルールを用いて優先度属性の重みを動的に切り替えることにより，多様なヒューリスティクスを内包するようなルール集合が獲得されていることが確認できる．

6.5.3 ルール集合の汎用性について

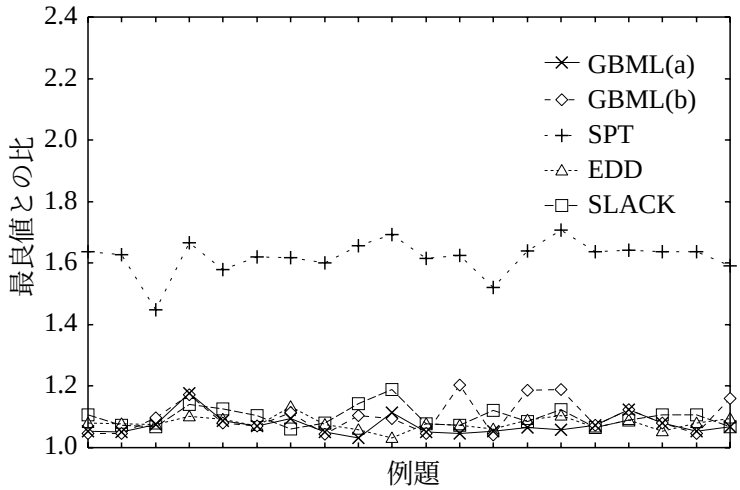
各例題クラスに対して，(学習のための) 例題を 20 個ずつ用意し ($\mathrm{E}_i^{\mathrm{LA}}$，$\mathrm{E}_i^{\mathrm{LB}}$ および $\mathrm{E}_i^{\mathrm{LC}}$)，これらの例題を用いてルール学習を行った (20 個の例題の平均評価値を各世代における適応度とする)．方法 GBML(a) および GBML(b) を用いて得られた最良ルール集合を，学習用の例題とは異なる (評価のための)20 個の例題 ($\mathrm{E}_i^{\mathrm{EA}}$，$\mathrm{E}_i^{\mathrm{EB}}$ および $\mathrm{E}_i^{\mathrm{EC}}$) にそれぞれ適用した．評価用の例題のそれぞれに (学習用の例題を用いて獲得された) ルール集合を適用した際の評価値 f_i^{R} を，これらの例題の最良評価値 f_i^* ($\mathrm{E}_i^{\mathrm{EA}}$ については最適スケジュール，$\mathrm{E}_i^{\mathrm{EB}}$ および $\mathrm{E}_i^{\mathrm{EC}}$ については方法 GA もしくは方法 GBML を用いて得られた最良スケジュール) との比 $\rho_i^\star\,(= f_i^{\mathrm{R}}/f_i^\star)$ を計算したものを図 6.6 に示す．なお，図 6.6 には，三つのヒューリスティクス (SPT: 処理時間最小作業優先則, EDD: 納期最早作業優先則および SLACK: 納期余裕最小作業優先則) を用いて得られた結果を合わせて示しておく．また，獲得されたルール集合のスケーラビリティ (獲得されたルール集合が，どのような規模の例題に対して有効か) の評価として，獲得された最良ルール集合を学習時とは異なるクラスの評価用例題へ適用した際の ρ および $\rho^\star$ の値を図 6.7 および図 6.8 へ示す．

図 6.6 より，獲得されたルール集合は，

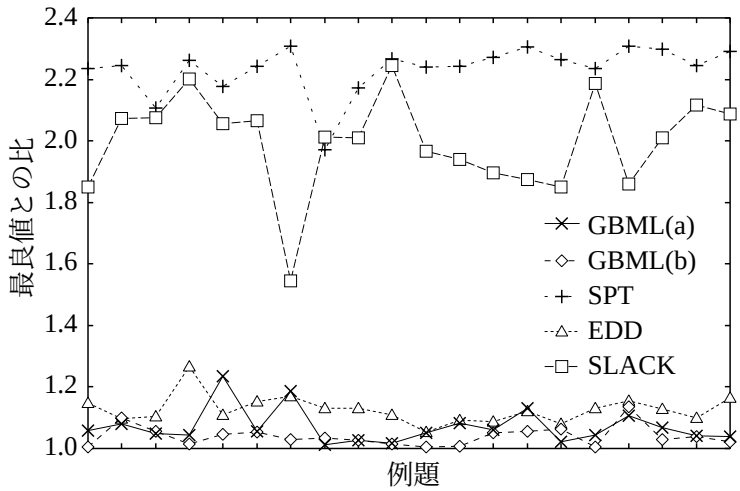
- 最適/準最適スケジュールから約 5% 程度劣ったスケジュールを平均的に得ている，



(a) $\mathcal{E}^{\mathrm{A}}$

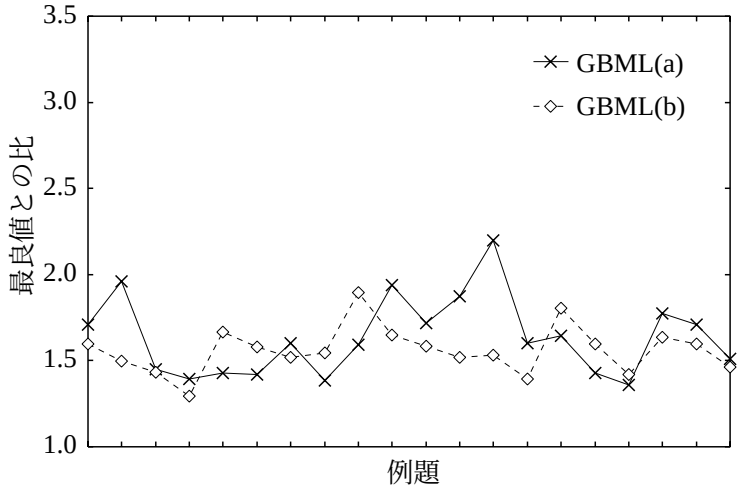


(b) $\mathcal{E}^{\mathrm{B}}$

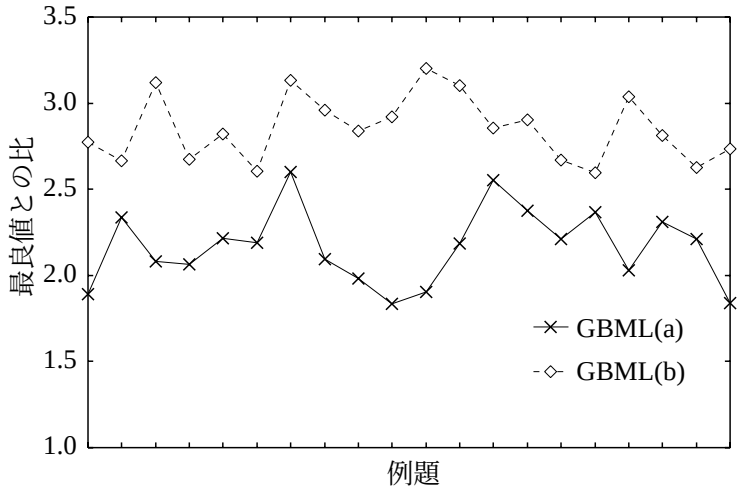


(c) $\mathcal{E}^{\mathrm{C}}$

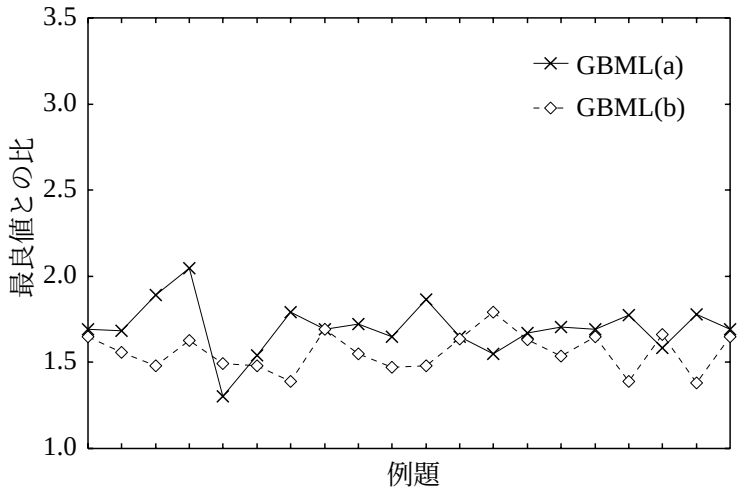
図 6.6: ルール集合のロバスト性



(a) 学習用例題: $\mathcal{E}^{\mathrm{A}}$，評価用例題: $\mathcal{E}^{\mathrm{B}}$

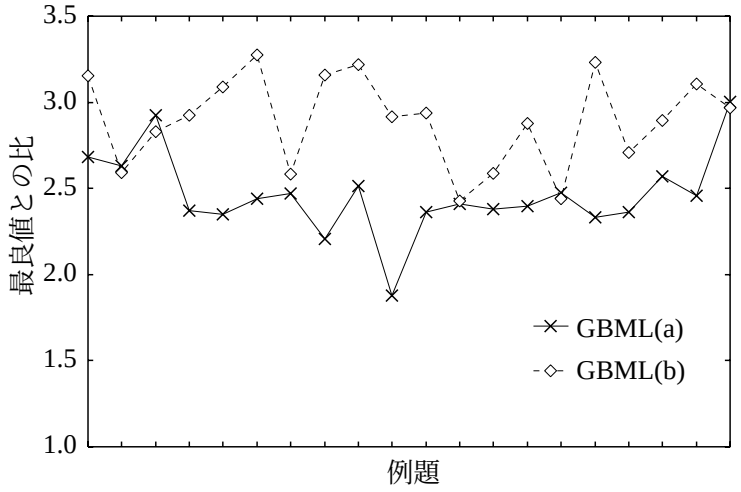


(b) 学習用例題: $\mathcal{E}^{\mathrm{A}}$，評価用例題: $\mathcal{E}^{\mathrm{C}}$

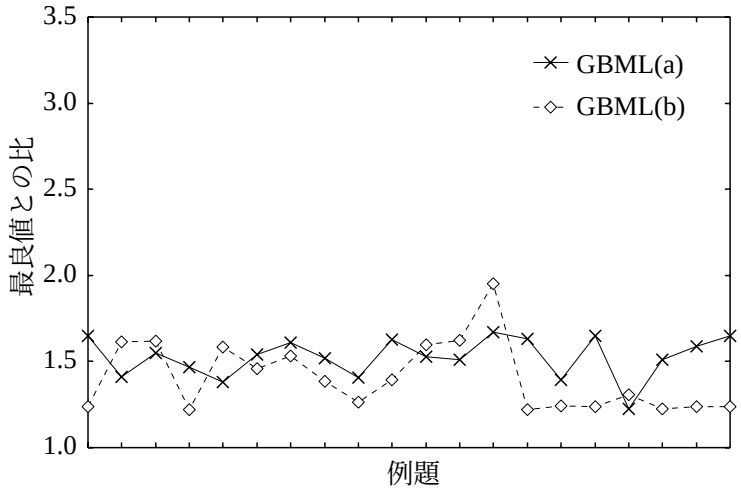


(c) 学習用例題: $\mathcal{E}^{\mathrm{B}}$，評価用例題: $\mathcal{E}^{\mathrm{A}}$

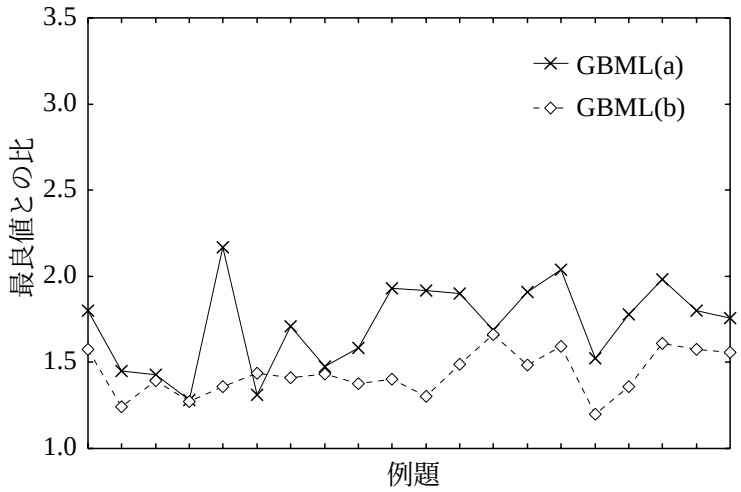
図 6.7: ルール集合のスケーラビリティ (1)



(a) 学習用例題: $\mathcal{E}^{\mathrm{B}}$，評価用例題: $\mathcal{E}^{\mathrm{C}}$



(b) 学習用例題: $\mathcal{E}^{\mathrm{C}}$，評価用例題: $\mathcal{E}^{\mathrm{A}}$



(c) 学習用例題: $\mathcal{E}^{\mathrm{C}}$，評価用例題: $\mathcal{E}^{\mathrm{B}}$

図 6.8: ルール集合のスケーラビリティ (2)

- EDD，SPT および SLACK に比べて，(例題に対して) 安定して良好なスケジュールを得ている

ことが分かる．これより，提案方法によって様々な状況 (問題) に対しても良好なスケジュールを生成し得る汎用的なルール集合が獲得されていると評価できる．

一方で，図 6.7 および図 6.8 より，獲得されたルール集合を学習時とは異なるクラスの例題へ適用した場合のパフォーマンス (スケーラビリティ) は優れていない．本章で示した枠組みでは，獲得されたルール集合が，ルール獲得時と類似するような状況 (問題) に対して優れた汎用性をもつことが確認されたものの，ルール獲得時と異なる状況に対して同様な性能を示すには，システムの状態の定義ならびに分割方法などを含む枠組みの見直しが必要であると思われる．

6.6　結言

本章では，第 3 章で示したスケジューリング・ルール獲得方式をフレキシブルショップ問題へ適用する方法を提案した．Pitt アプローチによるルール獲得時にあたっては，第 5 章での検討に基づき，シミュレータを構築することを考慮した．すなわち，仕事の機械割付けをヒューリスティックに決定した上で，機械上の作業の処理順序をルールを用いて決定するという方式をとる．またルール構成として，決定の際に用いられる問題の属性値の重みを後件部とするものを提案した．計算例を通して，提案方法により，様々な状況 (問題) に対しても良好なスケジュールを生成し得るような (汎用的な) ルール集合を獲得できることが分かった．

今後の課題として，

(a) 対象システムの状態の定義ならびに分割の方法について，さらにはそれらの自動獲得についての検討，

(b) スケーラビリティを有するようなルール集合の獲得方法についての検討，および

(c) 獲得されたルール集合の理論的な解析

などが挙げられる．

第 7 章

結論

本論文は，主として生産システムにおけるスケジューリング問題を対象としたスケジューリング・ルールの構成・設計方法および遺伝的機械学習に基づくルールの自動獲得・調整法の構築を目的とし，ルールの構成・獲得の一般的枠組み・方法論を提案するとともに，2 種類の現実的な型・規模の問題に対する提案法の具体的な適用方法を示した．以下，本論文で得られた結論を整理する．

第 2 章では，本論文が対象とするルール・ベースのスケジューリングについて，従来研究を整理するとともに，その一般的枠組み・方法論を示した．その際，ルール・ベースのスケジューリングが特に，仕事割り付けと同時に (ダイナミックに) 意思決定するリアルタイム方式との関連が深い点に注目し，ダイナミックにスケジュールを作成する方式，すなわちダイナミック・スケジューリングの範疇で，その位置付けを明確にした．

第 3 章では，第 2 章で示したスケジューリング・ルールを遺伝的機械学習に基づいて生成・調整するための計算モデルを示した．その際，シミュレーションに基づいて，事前 (オフライン的) にルール集合を獲得する Pitt アプローチの構成を考慮した．第 4 章および第 6 章で用いられる枠組みとして，ルールをスケジューリング状況に応じて意思決定するためのパラメータから構成し，パラメータ空間で最適あるいはロバストなルールを探索する方式を示した．

第 4 章では，第 3 章で示したリアルタイム・スケジューリングのためのルール獲得方式について，完成した製品を逐次在庫設備に振り分ける問題 (在庫設備選択問題) への適用方法を提案した．その際，ルール構成として，前件部に入庫する製品の属性および在庫設備の空き状況に関する条件を記述し，後件部に決定事項 (製品を割り付ける在庫設備番号) を直接記述したものを考慮した．計算例を通して，提案方法により効率的な運用を行うようなルール集合を獲得されていること，すなわち，製品の保管日数に関する分類に基づいて保管日数の短い製品を作業コストの低いバッファに入庫する傾向を有するルール集合が獲得されていることが確認された．

第 5 章では，在庫設備選択問題と比較して，より複雑な問題に位置づけられるフレキシブルショップ問題を取り上げ，第 6 章でルール獲得法を構成することを踏まえて，まず，問題の混合整数計画問題への定式化を示した．これにより，問題の決定事項および制約条件を明

確にした．さらに定式化に基づき，2 種類の決定事項 (仕事の機械割付け (決定 (A)) と機械上の作業の処理順序 (決定 (B))) を定める順序に関して 3 種類の方法を考案した．計算例を通して，決定 (A) を遺伝アルゴリズムを用いて探索するという形式で，探索の過程で選ばれた各探索点を，(決定 (A) を固定することにより得られる部分問題を解くことにより) 数理計画法により評価する，という階層的解法が，得られる解の質の良さ，安定性および計算時間の観点において優れていることが分かった．

第 6 章では，第 3 章で示したスケジューリング・ルール獲得方式をフレキシブルショップ問題へ適用する方法を提案した．Pitt アプローチによるルール獲得時には，第 5 章での検討に基づき，シミュレータを構築することを考慮した．すなわち，決定 (A) をヒューリスティックに決定した上で，決定 (B) をルールを用いて決定するという方式をとる．またルール構成として，決定の際に用いられる問題の属性値の重みを後件部とするものを提案した．計算例を通して，提案方法により，様々な状況 (問題) に対しても良好なスケジュールを生成し得るような (汎用的な) ルール集合を獲得できることが分かった．

以上より，本論文において，対象システムの状況に依存して決定を与える形式のルールを遺伝的機械学習によりオフラインに獲得する枠組みを提案し，大規模/複雑な例題に対しても汎用性などにおいて良好な性能を有するルール集合が獲得されることを示した．一方，本論文で提案した解法の有効性は，全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．特に対象システムの状態の定義ならびに分割数に関する指針もほとんどない状況である．さらには，獲得されたルール集合の理論的な解析も必要である．これらに関する検討が今後の課題である．一方，オンラインにスケジューリング・ルールを獲得・調整するアプローチについても今後，研究を進めていきたい．

謝 辞

本研究を進めるにあたり，終始適切なご指導およびご助言を賜わりました神戸大学工学部 北村新三教授に厚く御礼を申し上げます．また，懇切なるご指導およびご討論を戴きました 神戸大学工学部 藤井進教授，上原邦昭教授に謹んで感謝の意を表します．

神戸大学工学部 玉置久助教授には，本研究を進めるきっかけを与えて下さると共に，終始懇切・適切なご指導およびご鞭撻を賜りました．厚く御礼申し上げます．

研究全般にわたり，ご指導およびご討論を戴きました神戸大学国際文化学部 村尾元助教授に深く感謝の意を表します．神戸大学自然科学研究科 阿部重夫教授には，研究途上において懇切なご指導およびご教授を賜りました．慎んで感謝を申し上げます．生産計画および生産スケジューリングに関して，多くの有益なご助言およびご討論を戴きました株式会社神戸製鋼所 松田浩一博士，岩谷敏治氏，ならびに梅田豊裕博士に心から感謝の意を表します．日頃有益なご助言，また多大なるご協力を戴きました Fujitsu AMD Semiconductor Limited LLC の Edward Wu 氏には深く感謝の意を表します．スケジューリング理論に関してご教授およびご討論賜りました摂南大学工学部 諏訪晴彦助教授には，心から感謝を申し上げます．研究途上において有益なご助言と叱咤激励を賜りました株式会社創発システム研究所 中堀一郎博士に心から感謝の意を表します．

神戸大学工学部 北川郁技官には，研究環境の整備など，研究途上において公私ともに助けて戴きました．心から感謝いたします．日頃様々な方面でお世話になりました北村研究室ならびに阿部研究室の諸氏には心より感謝いたします．特に，豊田信一博士，大森清博 博士，稲元勉氏，亀山純一氏，門野成泰氏，木原健二氏，松本卓也氏，小野たまみ氏，川崎鑑太氏，丹治陽介氏には関連研究の勉強にご協力戴いたことを感謝します．

参考文献

1) M. Pinedo: *Scheduling: Theory, Algorithms, and Systems*, Prentice-Hall, 2nd edition, 2002.

2) J. Blazewicz, K. Ecker, E. Pesch, G. Schmidt and J. Weglarz: *Scheduling Computer and Manufacturing Processes*, Springer-Verlag, 2nd edition, 2001.

3) T. E. Morton and D. W. Pentico: *Heuristic Scheduling Systems*, John Wiley & Sons, 1993.

4) R. Shafaei and P. Brunn: Workshop scheduling using practical (inaccurate) data Part 1: The performance of heuristic scheduling rules in a dynamic job shop environment using a rolling time horizon approach, *Int. J. Prod. Res.*, Vol. 37, No. 17, pp. 3913–3925, 1999.

5) R. Shafaei and P. Brunn: Workshop scheduling using practical (inaccurate) data Part 2: An investigation of the robustness of scheduling rules in a dynamic and stochastic environment, *Int. J. Prod. Res.*, Vol. 37, No. 18, pp. 4105–4117, 1999.

6) M. J. Shaw, S. Park and N. Raman: Intelligent Scheduling with Machine Learning Capabilities – The Induction of Scheduling Knowledge, *IEE Trans.*, Vol. 24, No. 2, pp. 156–168, 1992.

7) C. Chiu and Y. Yih: A learning-based methodology for dynamic scheduling in distributed manufacturing systems, *Int. J. Prod. Res.*, Vol. 33, No. 11, pp. 3217–3232, 1995.

8) 柳浦睦憲, 茨木俊秀: 組合せ最適化問題に対するメタ戦略について, 電子情報通信学会論文誌 D-I, Vol. J83-D-I, No. 1, pp. 3–25, 2000.

9) D. E. Goldberg: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, 1989.

10) 玉置 久: 進化的アルゴリズム (2), *J. Signal Processing*, Vol. 4, No. 6, pp. 417–424, 2000.

11) 玉置 久, 越智通有, 荒木光彦: スケジューリング・ルール選択における状態フィードバックの試み, 計測自動制御学会論文集, Vol. 35, No. 3, pp. 428–434, 1999.

12) 鍋島一郎: スケジューリング理論, 森北出版, 1974.

13) 玉置 久: 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, 博士論文, 京都大学大学院工学研究科, 1993.

14) 関口恭毅, 木瀬 洋: スケジューリング理論の基礎と応用-I — 順序付けの基礎数理, システム/制御/情報, Vol. 44, No. 8, pp. 468–475, 2000.

15) 木瀬 洋: 生産スケジューリングの現状と動向, システム/制御/情報, Vol. 41, No. 3, pp. 92–99, 1997.

16) 黒田 充, 村松健児 (編): 生産スケジューリング, 朝倉書店, 2002.

17) P. Brucker: *Scheduling Algorithms*, Springer-Verlag, 3nd edition, 2001.

18) 西岡靖之: 状況対応型スケジューリング問題へのアプローチ, 計測と制御, Vol. 33, No. 7, pp. 571–574, 1994.

19) V. J. Leon, S. D. Wu and R. H. Storer: Robustness measures and robust scheduling for job shops, *IIE Trans.*, Vol. 26, No. 5, pp. 32–43, 1994.

20) S. V. Mehta and R. M. Uzsoy: Predictable Scheduling of a Job Shop Subject to Breakdowns, *IEEE Trans. Robitics and Automation*, Vol. 14, No. 3, pp. 365–378, 1998.

21) C. Bierwirth and D. C. Mattfeld: Production scheduling and rescheduling with genetic algorithm, *Evolutionary Computation*, Vol. 7, No. 1, pp. 1–17, 1999.

22) K. Miyashita and K. Sycara: CABINS: a framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair, *Artificial Intilligence*, Vol. 76, pp. 377–426, 1995.

23) 渡部 透, 西川郁子, 橋本泰典: オンライン・スケジューリングの方法論, 計測と制御, Vol. 33, No. 7, pp. 566–570, 1994.

24) 関口恭毅, 木瀬 洋: スケジューリング理論の基礎と応用-V — スケジューリング問題の近似解法, システム/制御/情報, Vol. 45, No. 4, pp. 213–219, 2001.

25) C. Rajendran and O. Holthaus: A comparative study of dispatching rules in dynamic flowshops and jobhshops, *European J. Operational Research*, Vol. 116, pp. 156–170, 1999.

26) K. C. Jeong and Y. D. Kim: A real-time scheduling mechanism for a flexible manufacturing system – using simulation and dispatching rules, *Int. J. Prod. Res.*, Vol. 36, No. 9, pp. 2609–2626, 1998.

27) E. Kutanoglu and I. Sabuncuoglu: An analysis of heuristics in a dynamic job shop with weighted tardiness objectives, *Int. J. Prod. Res.*, Vol. 37, No. 1, pp. 165–187, 1999.

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

29) 三宮信夫, 喜多 一, 玉置 久, 岩本貴司: 遺伝アルゴリズムと最適化, 朝倉書店, 1998.

30) 竹内 勝: 遺伝的アルゴリズムによる機械学習, 計測と制御, Vol. 32, No. 1, pp. 24–30, 1993.

31) S. F. Smith: *A Learning System Based on Genetic Algorithms*, PhD Thesis, University of Pittsburgh, 1980.

32) J. H. Holland and J. S. Reitman: Cognitive Systems Based on Adaptive Algorithms, *Pattern-directed Inference Systems* (D. A. Waterman and F. Hayes-Roth(eds.)), Academic Press, pp. 313–329, 1978.

33) 伊庭斉志: 遺伝的アルゴリズム, 医学出版, 2002.

34) 榊原一紀, 玉置 久, 村尾 元, 北村新三, 岩谷敏治, 松田浩一: リアルタイムスケジューリングに対する遺伝的機械学習アプローチ, 電気学会論文誌 C, Vol. 123, No. 4, pp. 823–831, 2003.

35) 一階良知, 井上雅晶, 大川剛直, 薦田憲久: 遺伝的アルゴリズムを用いた計画用知識学習方式, 電気学会 C 部門論文誌, Vol. 116, No. 5, pp. 577–583, 1996.

36) J. H. Holland: *Hidden Order: How Adaption Builds Complexity*, Addison-Wesley, 1995.

37) 吉見隆洋, 田浦俊春: 階層型分類子システムを用いた注視点制御過程の計算論的モデル, 計測自動制御学会論文集, Vol. 36, No. 10, pp. 842–851, 2000.

38) Y. Yang, S. Kreipl and M. Pinedo: Heuristics for Minimizing total weighted tardiness in Flexible Flow Shops, *J. Schduling*, Vol. 3, No. 2, pp. 89–108, 2000.

39) J. Cheng, Y. Karuno and H. Kise: A Shifting Bottleneck Approach for a Parallel-Machine Flowshop Scheduling Problem, *J. the Operations Research Society of Japan*, Vol. 44, No. 2, pp. 140–156, 2001.

40) K. E. Stecke: Formulation and Solution of Nonlinear Integer Production Planning Problems for Flexible Manufacturing Systems, *Management Science*, Vol. 29, No. 3, pp. 273–288, 1983.

41) H. Chen, J. Ihlow and C. Lehmann: A Genetic Algorithm for Flexible Job-Shop Scheduling, *Proc. the 1999 IEEE Int. Conf. Robotics & Automation*, pp. 1120–1125, 1999.

42) M. Mastrolilli and L.M. Gambardella: Effective Neighborhood Functions for the Flexible Job Shop Problem, *J. Scheduling*, Vol. 3, No. 1, pp. 3–20, 2000.

43) K. Mesghouni, S. Hammadi and P. Borne: Hybrid Representation for Genetic Algorithm to Solve Flexible Job Shop Scheduling, *Proc. the 15th IMACS World Congress on Scientific Computation*, pp. 433–438, 1997.

44) K. Sakakibara, H. Tamaki, H. Murao and S. Kitamura: Mathematical Modeling and Hybrid Solution for a Class of Flexible Shop Scheduling Problems, *Int. Symp. Scheduling 2002*, pp. 93–96, 2002.

45) 榊原一紀, 玉置 久, 村尾 元, 北村新三: フレキシブルショップ問題の数理定式化と進化的解法, 平成 14 年 電気学会電子・情報・システム部門大会, pp. 324–327, 2002.

46) T. Bäck, D. B. Fogel and Z. Michalewicz(eds.): *Evolutionary Computation 1*, Institute of Physics Publishing, 2000.

47) ILOG Inc.: *CPLEX 6.6*, http://www.ilog.co.jp/, 1999.

48) 榊原一紀, 玉置 久, 村尾 元, 北村新三: リアルタイム・スケジューリングのためのルールの構成と獲得・調整 — 遺伝的機械学習によるアプローチ, スケジューリング・シンポジウム 2003, pp. 104–109, 2003.

49) 榊原一紀, 玉置 久, 村尾 元, 北村新三: フレキシブルショップ問題への遺伝的機械学習アプローチ — リアルタイム・スケジューリングのためのルール獲得法, 情報処理学会論文誌数理モデル化と応用. 掲載予定.

50) B. Giffler and G.L. Thompson: Algorithms for solving production scheduling problems, *Operations Research*, Vol. 8, pp. 487–503, 1960.

本研究に関する発表

論文発表

(1) 榊原一紀, 玉置 久, 村尾 元, 北村新三, 岩谷敏治, 松田浩一, リアルタイムスケジューリングに対する遺伝的機械学習アプローチ, 電気学会論文誌C, Vol. 123, No. 4, pp.823-831, 2003.

(2) 榊原一紀, 玉置 久, 村尾 元, 北村新三, フレキシブルショップ問題への遺伝的機械学習アプローチ — リアルタイム・スケジューリングのためのルール獲得法, 情報処理学会論文誌数理モデル化と応用, 掲載予定.

(3) 榊原一紀, 玉置 久, 村尾 元, 北村新三, フレキシブルショップ・スケジューリング問題の数理計画モデルに基づくハイブリッド解法, システム制御情報学会論文誌, 掲載予定.

(4) K.Sakakibara, H.Tamaki, H.Murao, S.Kitamura, Metaheuristics Approach for Rule Acquisition in Flexible Shop Scheduling Problems, Post-Conference Volume of The Fifth Metaheuristics Int. Conference, 投稿中.

国際会議

(1) K.Sakakibara, H.Tamaki, H.Murao, S.Kitamura, "Mathematical Modeling and Hybrid Solution for a Class of Flexible Shop Scheduling Problems", *Int. Symp. Scheduling 2002*, pp.93-96, 2002.

(2) H.Tamaki, K.Sakakibara, H.Murao, S.Kitamura, "Modeling and Meta-Heuristic Solution for Flexible Shop Scheduling", *15th IFAC World Congress on Automatic Control*, CD-ROM paper, 2002.

(3) K.Sakakibara, H.Tamaki, H.Murao, S.Kitamura, "Toward a Real-Time Scheduling for a Class of Flexible Shop Problems", *SICE Annual Conference 2002*, pp.1379-1384, 2002.

(4) K.Sakakibara, H.Tamaki, H.Murao, S.Kitamura, "Metaheuristics Approach for Rule Acquisition in Flexible Shop Scheduling Problems", *The 5th Metaheuristics Int. Conference*, CD-ROM paper, 2003.

口頭発表

(1) 榊原一紀, 玉置 久, 阿部重夫, 北村新三, リアルタイム・スケジューリングに対する遺伝的機械学習アプローチ, 第 6 回創発システム・シンポジウム, pp.51-52, 2000.

(2) 玉置 久, 小野たまみ, 榊原一紀, 村尾 元, 北村新三, フレキシブルショップ・スケジューリング問題の数理モデル化と進化的解法, 電気学会情報システム研究会 (IS-01), pp.11-16, 2001.

(3) 榊原一紀, 玉置 久, 村尾 元, 北村新三, フレキシブルショップ問題の数理計画と進化型計算によるハイブリッド解法, スケジューリング・シンポジウム 2001, pp.161-166, 2001.

(4) 榊原一紀, 玉置 久, 村尾 元, 北村新三, リアルタイム・スケジューリングに対する遺伝的機械学習アプローチ, 計測自動制御学会関西支部シンポジウム, pp.71-74, 2001.

(5) 榊原一紀, 玉置 久, 村尾 元, 北村新三, フレキシブルショップ・スケジューリング問題の数理モデルと解法, 第 1 回サイバネティック・フレキシブル・オートメーションシンポジウム, pp.3-6, 2001.

(6) 榊原一紀, 玉置 久, 村尾 元, 北村新三, リアルタイム・スケジューリングのための遺伝的機械学習アプローチ — フレキシブルショップ問題を対象として, 第 8 回創発システム・シンポジウム, pp.103-104, 2002.

(7) 榊原一紀, 玉置 久, 村尾 元, 北村新三, フレキシブルショップ問題の数理定式化と進化的解法, 平成 14 年 電気学会 電子・情報・システム部門大会, pp.324-327, 2002.

(8) 榊原一紀, 玉置 久, 村尾 元, 北村新三, スケジューリング・ルール獲得のための遺伝的機械学習アプローチ — フレキシブルショップ問題の数理計画と進化型計算によるハイブリッド解法, スケジューリング・シンポジウム 2002, pp.134-139, 2002.

(9) 榊原一紀, 玉置 久, 村尾 元, 北村新三, GBML を用いたスケジューリング・ルールの獲得法 — フレキシブルショップにおけるリアルタイム・スケジューリングを目的として, 第 2 回サイバネティック・フレキシブル・オートメーションシンポジウム, pp.37-40, 2002.

(10) 榊原一紀, 玉置 久, 村尾 元, 北村新三, 遺伝的機械学習によるスケジューリング・ルールの獲得 — フレキシブルショップ問題を対象として, システムインテグレーション部門講演会, pp.287-288, 2002.

(11) 榊原一紀, 玉置 久, 村尾 元, 北村新三, フレキシブルショップ問題への遺伝的機械学習アプローチ — リアルタイム・スケジューリングのためのルール獲得法, 第 9 回 MPS シンポジウム, pp.93-98, 2003.

(12) 榊原一紀, 玉置 久, 村尾 元, 北村新三, リアルタイム・スケジューリングのためのルー

ルの構成と獲得・調整 — 遺伝的機械学習によるアプローチ, スケジューリング・シンポジウム 2003, pp.104-109, 2003.