The University of Osaka
Institutional Knowledge Archive

Title	柔軟な生産計画支援システムの実現のための知識情報処理に関する研究
Author(s)	川嶋, 一宏
Citation	大阪大学, 2013, 博士論文
Version Type	VoR
URL	https://hdl.handle.net/11094/27484
rights	
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

博第 76083

柔軟な生産計画支援システムの実現のための知識情報処理に関する研究

2013 年 1 月

川嶋　一宏

柔軟な生産計画支援システムの実現のための知識情報処理に関する研究
2013年1月
川嶋
博
69

柔軟な生産計画支援システムの実現のための知識情報処理に関する研究

提出先　　大阪大学大学院情報科学研究科

提出年月　2013 年 1 月

川嶋　一宏

内容梗概

本論文は，筆者が 1984 年から現在まで(株)日立製作所システム開発研究所，およびセキュリティトレーサビリティ事業部ならびに 2007 年から現在まで大阪大学大学院情報科学研究科マルチメディア工学専攻在学中に行ってきた柔軟な生産計画支援システムの実現のための知識情報処理に関する研究成果をまとめたものである．

生産計画業務には，計画対象とする製品や生産を行う装置に対して，生産工程や製造時間等の守らなければならない制約条件と，計画目標とする生産効率を評価する多種多様な評価指標がある．生産計画支援システムは，計画問題の定義として計画対象の制約条件と評価指標を取り込むことと，その制約条件に従って評価指標の値が高くなる計画を短時間に立案することが要求されるシステムである．しかも，計画対象の制約条件や評価指標は，市場ニーズの変化や生産工程の改善活動によって，頻繁に変化する．このため，生産計画支援システムには，計画問題定義の変更容易性とともに，計画を立案する計画立案手順にも計画問題定義の変更に対応した変更容易性が求められる．

本論文では，市場ニーズの変化や生産工程の改善活動によって起こる計画対象の制約条件や評価指標の変化に対応するため，生産計画支援システムの計画問題定義と計画立案手順の変更容易性を確保する知識情報処理について述べる．特に，計画対象の制約条件と評価指標，計画立案手順の計画知識は，日々の生産計画業務を行っている計画者が有していることから，計画者の変更容易性を確保する計画知識の記述形式と，その記述内容を生産計画支援システムに実装する情報処理方式を提案する．

第 1 章では，生産計画業務における生産計画支援システムの位置付けと，柔軟な生産計画支援システムの実現課題を示し，本研究の方針を述べる．

第 2 章では，計画問題定義の変更容易性に着目し，生産工程や製造時間等

の制約条件や評価指標の値を算出するために生産現場で用いられている数表と数式を示す．それらの数表と数式を生産計画支援システムに取り込む計画問題用の記述言語を導入し，その記述内容を処理効率の高い計算機プログラムに変換するプリコンパイル方式を提案する．

第 3 章では，計画立案手順の変更容易性に着目し，数理計画法等の求解アルゴリズムを固定的に適用することが困難なスケジューリング問題の計画立案手順を検討する．計画立案プログラムの記述内容を計画者が変更する部分とシステム開発者が開発保守する部分に分け，計画立案手順の変更容易性を確保する計画立案プログラムの構成方式と記述形式を提案する．

第 4 章では，計画対象の制約条件と評価指標の変化に対して計画立案手順を選択する計画立案ノウハウに着目し，計画者の選択履歴を教師データとして，計画立案手順を選択するルールを簡潔な知識表現で獲得する統計的集約型知識獲得方式を提案する．

第 5 章では，本研究の成果をまとめ，今後の課題を示す．

研究業績目録

A. 学術論文誌論文

1. 川嶋一宏，薦田憲久，原田俊一，三森定道：知識型計画支援システム向業務論理記述言語用プリコンパイラ，情報処理学会論文誌，Vol.28，No.9，pp.975-986 (1987).

2. 大場みち子，薦田憲久，川嶋一宏：知識型計画システムにおける制約指向型説明機能，情報処理学会論文誌, Vol.31，No.11，pp.1688-1695 (1990).

3. 川嶋一宏，薦田憲久：知識型計画システムにおける統計的集約型知識獲得方式，情報処理学会論文誌, Vol.34, No.5, pp.864-872 (1993).

B. 国際会議

1. K. Kawashima, N. Komoda, S. Harada, and S. Mitsumori: Precompiler for Process Logic Description Language in Intelligent Planning Support System, in *Proc. of 1987 Annual International Computer Software & Applications Conference (COMPSAC'87)*, pp.649-655 (1987).

2. K. Kawashima and N. Komoda: Multivariate Analytical Knowledge Acquisition Method for Knowledge based Planning System, in *Proc. of IEEE International Conference on Industrial Electronics Control and Instrumentation (IECON'91)*, pp.1622-1627 (1991).

3. K. Kawashima and N. Komoda: An Information Management System in Inter-organization Supply Chain by Secure RFID Tag, in *Proc. of the 12th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2007)*, pp.301-308 (2007).

C. 学会等講演

1. 川嶋一宏，薦田憲久，原田俊一，福田正廣：柔軟な計画を可能とする計画システム開発支援システム用業務論理記述言語とプリコンパイル方式，情報処理学会第 32 回全国大会, 4W-6, pp.1897-1898 (1986).

2. 川嶋一宏，薦田憲久，原田俊一，福田正廣：知識型計画支援システム向業務論理記述言語用プリコンパイラ，情報処理学会第 33 回全国大会, 3W-5, pp.2065-2066 (1986).

3. 川嶋一宏，薦田憲久，原田俊一：知識型計画支援システム向業務論理記述言語用プリコンパイラの最適化機能，情報処理学会第 34 回全国大会, 1T-8, pp.1089-1090 (1987).

4. 川嶋一宏，薦田憲久，原田俊一：知識型計画支援システム向業務論理記述言語用プリコンパイラの拡張機能，情報処理学会第 35 回全国大会, 6D-5, pp.2637-2638 (1987).

5. 川嶋一宏，薦田憲久，原田俊一：知識型計画支援システムの多段処理機能，情報処理学会第 36 回全国大会, 7Q-1, pp.1585-1586 (1988).

6. 川嶋一宏，原敬市，薦田憲久：知識型計画支援システムＨＰＧＳによるスケジューリング問題記述方式，情報処理学会第 39 回全国大会, 2B-4, pp.155-156 (1989).

7. 原敬市，川嶋一宏，薦田憲久：知識型計画支援システムＨＰＧＳによるスケジューリングシステムの開発，情報処理学会第 39 回全国大会, 2B-5, pp.157-158 (1989).

8. 薦田憲久，川嶋一宏，原敬市：知識型計画支援システムＨＰＧＳによるスケジューリングシステム, 計測自動制御学会第 15 回システムシンポジウム, pp.361-366 (1989).

9. 原敬市，川嶋一宏，大場みち子，薦田憲久：知識型計画システムの開発手順，情報処理学会第 40 回全国大会, 2D-5, pp.238-239 (1990).

10. 川嶋一宏，原敬市，薦田憲久，大場みち子：知識型計画システム用データ基本操作機能，情報処理学会第 40 回全国大会, 2D-6, pp.240-241 (1990).

11. 大場みち子，薦田憲久，川嶋一宏：知識型計画システムにおける計画結果説明機能，情報処理学会第 40 回全国大会, 2D-7, pp.242-243 (1990).

12. 川嶋一宏，薦田憲久：計画システムにおける統計的集約型知識獲得方式の提案，情報処理学会第 41 回全国大会, 5K-1, pp.2-43-2-44 (1990).

13. 大場みち子，薦田憲久，川嶋一宏：計画向け制約指向型説明機能の説明文選択方式，情報処理学会第 41 回全国大会, 5K-2, pp.2-45-2-46 (1990).

14. 川嶋一宏，薦田憲久，原敬市，大場みち子：知識型計画システム，計測自動制御学会第 7 回産業システムシンポジウム, pp.5-8 (1990).

15. 大場みち子，薦田憲久，川嶋一宏，原敬市：知識型スケジューリングシステムにおける最適化機能, 計測自動制御学会第 16 回システムシンポジウム, pp.161-166 (1990).

16. 川嶋一宏，薦田憲久：知識型計画システムにおける統計的集約型知識獲得方式の提案, 計測自動制御学会第 12 回知能システムシンポジウム, pp.39-44 (1990).

17. 大場みち子，薦田憲久，原敬市，川嶋一宏：対話型スケジューリングシステムにおける改善知識型最適化方式，情報処理学会第 42 回全国大会, 2N-2, pp.1-289-1-290 (1991).

18. 原敬市，大場みち子，薦田憲久，川嶋一宏：最適化機能を有する知識型スケジューリングシステムの開発，情報処理学会第 42 回全国大会, 2N-3, pp.1-291-1-292 (1991).

19. 川嶋一宏，薦田憲久：知識型計画システムにおける知識獲得機能の拡張，情報処理学会第 42 回全国大会, 2N-4, pp.1-293-1-294 (1991).

20. 川嶋一宏，原敬市：学習機能を有する知識型スケジューリングシステムの開発，電気学会電子・情報・システム部門第 1 回全国大会, D-4-6, pp.362-363 (1991).

21. 川嶋一宏，薦田憲久：サプライチェーンにおけるセキュア型電子タグを用いた企業間情報管理システム，電気学会情報システム研究会，IS-07-35, pp.19-21 (2007).

D. 単行本，解説，等

1. 原敬市，薦田憲久，川嶋一宏，大場みち子：知識型スケジューリングシステムの開発，日立マイコン技報, Vol.4, No.1, pp.56-61 (1990).

2. アーバンプロデュース 出版部編：次世代生産管理システムの構築と運用，アーバンプロデュース (1992) (Ⅲ-3 スケジューリングシステムの導入と支援ツールの章 pp.687-709 を担当).

3. 川嶋一宏，浜口強，前田章，坂下正洋：金融機関におけるデータベースマーケティングの展開，日立評論, Vol.77, No.6, pp.31-34 (1995).

4. 川嶋一宏：技術探索 電子マネーが生活を変える，電気学会誌, Vol.116, No.9, pp.603-606 (1996).

5. 日立製作所・新金融システム推進本部編：図解よくわかる電子マネー，日刊工業新聞 (1996) (第 2 部モンデックス pp.111-154 を担当).

6. K. Kawashima: Survey on the Current Status and Market Size of Electronic Commerce for 2004 (Information Economy Outlook 2005), in *Proc. of e-Biz Expo2005 Conference*, pp.155-173 (2005).

7. 川嶋一宏：電子商取引と情報セキュリティ，カードウエーブ, Vol.19, No.10, pp.124-125 (2006).

8. 川嶋一宏：次世代電子商取引について，カードウエーブ，Vol.21，No.4, pp.62-63 (2008).

9. 川嶋一宏：電子タグ普及活動について，カードウエーブ，Vol.21，No.10, pp.58-59 (2008).

10. 川嶋一宏，鳥屋尾彰：次世代サプライチェーンについて，カードウエーブ, Vol.22, No.4, pp.58-59 (2009).

目　次

第1章 序論 1
1.1 研究の背景 1
1.2 関連研究 3
1.3 本研究の方針 6
1.4 本論文の構成 8
第2章 計画問題定義言語とプリコンパイル方式 9
2.1 緒言 9
2.2 計画問題定義 10
2.2.1 生産計画問題の定義内容 10
2.2.2 計画問題定義の表現形式 12
2.2.3 計画者による計画問題定義記述の課題 13
2.3 計画問題定義言語 14
2.4 プリコンパイラの機能 18
2.5 メモリ割付による計算量の削減方式 22
2.5.1 プリコンパイラの最適化機能 22
2.5.2 メモリ割り付け計算量最小化問題 28
2.5.3 解探索の考え方 29
2.5.4 メモリ割り付け計算量削減アルゴリズム 35
2.5.5 動作例と評価 36
2.6 プリコンパイラの全体構成 40
2.7 提案方式の評価 42
2.7.1 計画問題記述言語の評価 42
2.7.2 プリコンパイル方式の評価 43
2.8 結言 46

第 3 章 知識型スケジューリング用計画立案プログラム構成方式と記述形式 49
3.1 緒言 49
3.2 生産スケジューリングの概要 50
3.2.1 生産スケジューリング問題の特徴 50
3.2.2 生産スケジュールの立案方法 53
3.3 柔軟な計画立案プロクラムの実現方式 55
3.3.1 実現の考え方 55
3.3.2 計画立案プログラムの構成方式と記述形式 58
3.3.3 スケジューリング関数と割付基盤プログラム 61
3.3.4 知識型スケジューリングシステムのシステム構成 64
3.4 食品加工スケジューリング問題への適用と評価 66
3.4.1 計画問題定義と計画立案手順 67
3.4.2 計画立案手順の変更と計画結果の変化 70
3.4.3 考察 72
3.5 結言 73
第 4 章 統計的集約型知識獲得方式 75
4.1 緒言 75
4.2 知識獲得のアプローチと課題 76
4.2.1 知識獲得の基本構成 76
4.2.2 実現上の課題 78
4.3 提案方式 MAKAM の概要 80
4.4 提案方式 MAKAM のアルゴリズムと動作例 83
4.5 提案方式の評価 86
4.5.1 フローショップスケジューリング問題 87
4.5.2 探索空間と計画性能の評価 87
4.5.3 考察 91
4.6 結言 92
第 5 章 結論 93
5.1 本研究のまとめ 93
5.2 今後の研究課題 94
謝辞 97
参考文献 99

第1章 序論

1.1 研究の背景

製造業では，製品の多様化や製品寿命の短命化が進み，製品や生産量の変化に柔軟に対応できる生産システムを構築することが求められている[1-3]．このため，各企業は柔軟性をもつ製造制御システム[4-6]を生産現場に導入し，製品やその生産すべき量の変化に応じて製造装置や生産工程等を変更することで，その変化に即応可能な製造制御を実現している．製造制御での柔軟性が増すにつれ，製品や生産量の変化にあわせた生産工程の改善活動が行われるようになり，生産工程や装置等に頻繁な変更が起こり，生産割当や生産スケジュール等の生産計画の立案を支援する生産計画支援システムにも，その変化に対応可能な柔軟性が求められている[7-11]．

生産計画業務では，図 1.1 に示すように，日々変化する製品の生産量，生産を行う製造装置，生産工程にあわせ，計画者が生産計画支援システムを用い，要求された生産量を生産効率よく製造する計画を立案する．

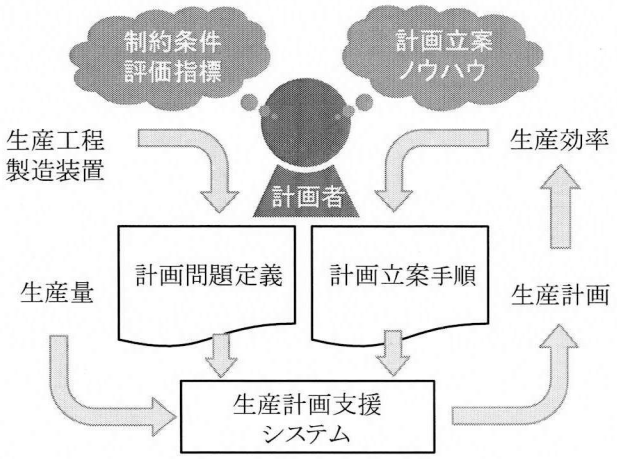

図 1.1　生産計画業務における生産計画支援システムの位置付け

生産計画支援システムには，計画対象とする製品や装置に対して生産工程や製造時間等の守らなければならない制約条件，計画目標とする生産効率の評価指標等の計画問題を定義する情報（以下，計画問題定義と呼ぶ）と，定義された問題を解くためのノウハウやアルゴリズムに関する情報（以下，計画立案手順と呼ぶ）が，プログラムや設定値等の形で埋め込まれる．生産計画支援システムの柔軟性を確保するためには，計画問題定義ならびに計画立案手順の変更容易性が必要であるが，従来の生産計画支援システムでは，計画問題定義と計画立案手順が混在してプログラムコードの中に埋め込まれ，その変更はきわめて困難である．

これに対して，特定の計画業務向けの生産計画支援システム[12-19] では，計画対象固有の計画問題定義や計画立案手順の一部を設定するテーブルを設け，それらの設定値を取りこんでいるが，変更可能な範囲が限定的である．計画立案手順として数理計画法の求解アルゴリズム[20-28]を適用することも考えられる．しかしながら，標準的なパッケージ[29-32]では，適用する求解アルゴリズムが既存のアルゴリズムに限定され，計画対象の制約条件や評価指標の変化に追従できない．しかも，生産計画問題には製品や生産工程ごとに異なる制約条件や多種多様な評価指標が多数存在する．その変化を事前に予測し，その変化に対応するすべてのプログラムを開発するには多大なコストと時間がかかる．

また，知識工学を適用したエキスパートシステム[33-37]では，計画問題定義や計画立案手順の記述にルール表現の適用が検討されているが，記述可能な範囲が限定的である．この表現では，生産計画で重要となる定量的な制約条件や評価指標の記述ができない．しかも，生産計画問題の多くが大規模な組み合わせ最適化問題であり，その組み合わせ計算の中でルールによる推論処理が行われると，計算処理性能の問題が発生する．このため，計算処理性能を意識して，多くの部分を汎用の計算機言語で書かざるを得ないのが実情である[38].

汎用の計算機言語で記述された計算機プログラムの変更には変更仕様書を作成して，計算機の専門家であるシステム開発者に依頼することが必要となり，計画者による計画問題定義や計画立案手順の変更容易性を阻害する．柔軟な生産計画支援システムの実現には，計画問題定義や計画立案手順の知識を有した計画者による記述内容の変更容易性と膨大な組み合わせ計算に対する計算処理性能を考慮することが必要である．

しかも，生産計画では，製品や生産工程が変化し，制約条件や評価指標を変更

する場合には，新たに適切な計画立案手順を検討し，その手順をシステムに実装する必要がある．この計画立案手順は，多くの生産現場で熟練の計画者が検討し，頭の中にノウハウの形で持っていることが多く，明文化されていない．その抽出と実装に多大の時間とコストがかかる．これに対しては，計画立案手順のヒアリング[39]や，教師データからの機械学習[40, 41]が考えられる．しかしながら，製品や生産工程が変化するたびに，計画者へのヒアリングや教師データの収集を実施するのでは，計画立案手順を迅速に変更することが実現できない．柔軟な生産計画支援システムの実現には，計画立案手順を変更する計画知識の獲得を考慮することが必要である．

こうした背景から柔軟な生産計画支援システムの課題として，計画問題定義と計画立案手順の変更容易な記述方法と計算処理性能，計画立案手順変更に関する知識の獲得方法と，それらの実現方式を取り上げる．

1.2 関連研究

計画問題定義と計画立案手順の記述方法と計算処理性能，計画立案手順の知識獲得方法の関連研究を以下に示す．

(1) 計画問題定義と計画立案手順の記述方法

生産計画支援システムの柔軟性確保に対するアプローチには，数理計画法の求解アルゴリズムを用いて計画立案手順を固定化し，計画問題定義の変更容易性を取り入れようとした数理的アプローチ[29-32]と，計画問題定義と計画立案手順を固定化せず，変更容易な知識表現を用いて計画者の頭の中にある計画問題定義と計画立案手順の知識を取り入れる知識工学的アプローチ[33-37]がある．

数理的アプローチの数理計画問題記述簡易言語[32]では，求解アルゴリズムが前提とする制約式や評価式の係数に対する設定値計算の処理記述に算術演算可能な簡易言語を導入している．簡易言語で定義された処理内容から設定値を自動計算することにより，制約式や評価式の係数の変更容易性を確保しようとするアプローチである．このアプローチでは，求解アルゴリズムの制約式や評価式の係数の意味を理解しなければ，設定値の計算内容を正しく記述することができない．しかも，その記述形式は生産現場で用いられている製造時間や段取

時間等を定義する数表や数式の表現形式とは異なる．このため，数理計画法の専門家でなければ，設定値の計算内容に関する記述の変更に時間がかかる．

生産計画では，製品や装置ごとに製造時間や段取時間等の計画問題定義が異なる．その記述には条件分岐や論理演算も必要であるが，この言語ではそれらの計算処理を記述することができない．制約式や評価式の計算に条件分岐や論理演算が含まれるようになると，線形計画法の求解アルゴリズムの適用が困難となり，非線形計画法の求解アルゴリズムに拡張することが必要となる．このアプローチでは，計画立案手順が既存の求解アルゴリズムに限定されるとともに，条件分岐や論理演算が含まれる計画問題定義の変更容易性を確保することができない．

一方，知識工学的アプローチは，計画問題定義や計画立案手順の記述に理解容易な表現形式を用いることにより，それらの変更容易性を確保するアプローチである．このアプローチの計画型エキスパートシステム[33-37]では，計画問題定義と計画立案手順の記述形式として IF-THEN 形式のルール表現を導入している．しかし，この形式では「IF 製品品種が[a]ならば，THEN 製造装置は[B1]とする」というパターン処理の記述に限定される．製造時間や段取時間等の定量的な値を算定する計算処理は汎用の計算機言語で記述し，その処理を呼び出すという実装が行われている．IF-THEN 形式のルール表現では制約条件や評価指標の定量的な値を算定するための数表や数式を記述することができない．

大規模な生産計画問題になると，製品と装置の組み合わせのパターンを記述するだけでも膨大な数のルールを必要とする．そのルール増大に対しては計画立案時に適用するルールの優先度が導入されている[42-44]．しかしながら，ルールの優先度で定義できる内容は限定的であるとともに，ルールの優先度記述を導入すると，守らなければならない計画問題定義の記述であるか，計画立案手順の中で優先するルールの記述であるかが，混然一体化する．個々のルール記述が理解容易な表現であっても，ルール全体の変更容易性が失われる．

しかも，知識工学的アプローチにはルールのパターン処理に時間がかかるという問題がある．この問題に対し，知識ベースシステムの高速処理方式[45]では事前にコンパイル可能なルールのパターン処理を機械語に変換し，処理効率の向上を図っている．しかしながら，大規模な生産計画問題になると，製品や装置の組み合わせが増大し，パターン処理の膨大な繰り返しが必要となる．任意の組み合わせに対するパターン処理だけを機械語に変換しても，実用的な処理時間で計画を立案することができない[38]．

こうした処理性能の問題に対し，ルールベース型の作業スケジュール方式[46]では，「納期の厳しい製品が多ければ，納期の厳しい製品を優先して割り付ける」という割付状態に応じて計画立案手順をディスパッチする部分だけにルール記述を取り入れている．その他の部分は，汎用の計算機言語で処理効率の良いプログラムを開発することで，計画立案手順の変更容易性と実用的な処理性能を確保する方式を提案している．しかしながら，割付状態を認識するための「納期の厳しい製品」といった特徴の抽出（以下，特徴抽出と呼ぶ）や「納期の厳しい製品を優先して割り付ける」といった計画立案手順を実装したプログラムにも制約条件や評価指標の変化にあわせた変更が求められる．

これらの数理的アプローチや知識工学的アプローチでは，記述範囲が限定され，計画問題定義と計画立案手順の多くの部分を，汎用の計算機言語で処理効率を意識してプログラム開発する必要があり，計画問題定義と計画立案手順の変更容易性を阻害する要因となっている．

(2) 計画立案手順の知識獲得

生産計画の業務知識には，計画対象とする製品や生産工程の制約条件や評価指標等の計画問題定義に関わる知識と，計画立案過程で得られる計画立案手順に関する知識がある．計画者の頭の中にあるこれらの知識の獲得に関しては，従来から数多くの研究が行われている[39-41, 47-53]．

知識ベースシステム構築のための知識獲得法[50]では，計画業務全体の知識を計画問題の枠組みに基づき，計画目的，制約条件，立案手順等に分け，計画者にヒアリングすることが提案されている．計画問題定義の知識は対象とする製品や生産工程に起因する知識であり，計画対象の専門家から論理的根拠を詳細にヒアリングすることで計画問題定義を得ることが可能である．しかしながら，計画立案手順に関わる知識は，計画立案を繰り返す過程で得られる経験的知識であり，論理的根拠が少なく，その知識にはあいまいさや誤りを含む．このため，インタビューを繰り返しても，「納期の厳しい製品が多ければ，納期の厳しい製品を優先して割り付ける」のように，定性的なディスパッチングルールしか得られない．これら知識を定量的に表現できなければ，生産計画支援システムで計画を立案できない．

こうした計画立案手順の獲得の困難性に対して，ディスパッチングルールの生成[50]では，作業時間と納期余裕時間の和が最小となる割付が厳しい仕事を

優先するという特定の評価尺度を導入することで，平均納期遅れ時間と最大納期遅れ時間が最適となるディスパッチングルールを得ている．しかしながら，生産計画の計画目的は，平均納期遅れ時間や最大納期遅れ時間の最小化だけでなく，生産余裕がある場合には作業時間の平準化という目的もある．割付が厳しい仕事を優先するというルールだけでなく，生産計画の多様な評価指標に対応することが可能で汎用的なルール獲得方法が必要である．

汎用的なルール獲得方法としてバージョン空間法[52]による知識表現の一般化手法[53]が提案されている．しかしながら，教師データにあいまいなデータや誤りを含むと，一般化が不十分となり，まとまった知識表現の獲得ができない．このため，ディスパッチングルールの獲得に適用しても，教師データに含まれるあいまいなデータや誤りによって，獲得されるルールの数が増大し，計画者による直接変更を可能とする簡潔なルールが獲得できない．

以上のように，これらの関連研究では頻繁に変化する計画対象の制約条件と評価指標を生産計画支援支援システムに迅速に取り込み，取り込んだ計画問題定義の変化にあわせて計画立案手順を変更し，迅速に計画を立案することができない．

1.3 本研究の方針

本研究の方針を図 1.2 に示す．

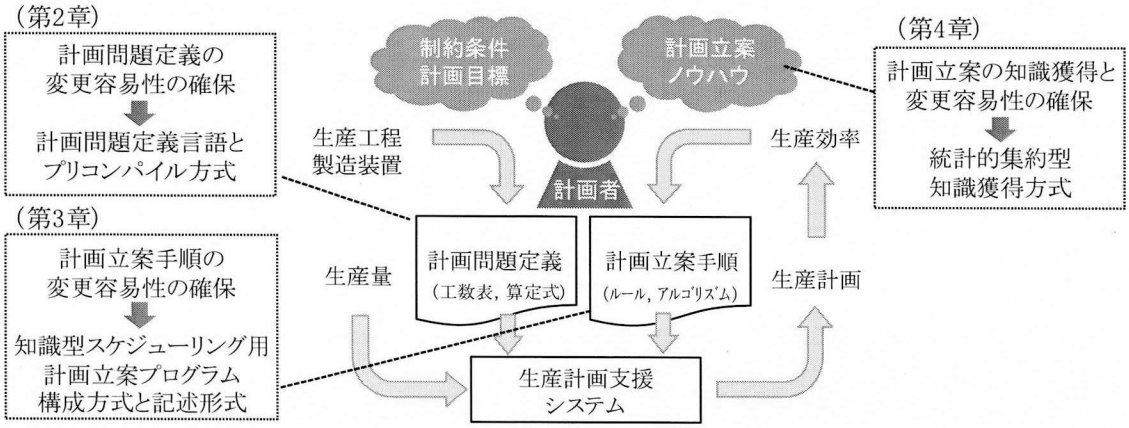

図 1.2　柔軟な生産計画支援システム実現のための研究方針

(1) 計画問題定義言語とプリコンパイル方式

第 1 の研究課題として，生産計画支援システムの柔軟性を確保するために，計画対象の制約条件や評価指標を定量的に定義する計画問題定義の変更容易性を確保する記述言語と計算機プログラムへの変換方式を研究する．

計画問題定義の変更容易な記述言語として，制約条件や評価指標の計算に用いる数表や数式の記述に生産現場で用いられている表現を取り入れる．特に，情報処理の専門家ではない計画者が記述することを考慮する．生産現場の専門用語を用い，理解や変更が容易な順序（計算機の処理順序に制約されない記述順序）での記述を可能とする計画問題定義用の記述言語を提案する．

計画問題定義は，計画対象（製品や装置等）の組み合わせに対する制約条件と評価指標の条件評価計算の定義である．計算機プログラムの変換方式として，組み合わせ計算を効率よく処理するために，手続き型プログラムに変換するプリコンパイル方式を検討し，手続き型言語のコンパイル技術では最適化されない組み合わせ計算の多重な繰返し処理の最適化方式を提案する．

(2) 知識型スケジューリング用計画立案プログラム構成方式と記述形式

第 2 の研究課題として，計画問題定義言語で定義された制約条件や評価指標に従い計画を立案する計画立案手順の変更容易性を確保するために，計画立案手順を実装する計画立案プログラムの構成方式と記述形式を研究する．

特に，生産計画の中でシステム化ニーズが高く，個別の制約条件を数多く有し，計画立案手順として求解アルゴリズムを固定的に適用することが困難な生産スケジューリング問題を対象とする．

変更容易な計画立案手順として，割付状態に応じて割付方法を選択するディスパッチングルール（以下，割付戦略決定ルールと呼ぶ）を導入する．「特徴抽出，戦略決定，割付更新」を繰り返す計画立案手順を検討し，変更容易なプログラムの構成方式と記述形式を提案する．

(3) 統計的集約型知識獲得方式

第 3 の研究課題として，計画問題定義の変化に合わせて計画立案手順を変更する計画者の経験的な知識である割付戦略決定ルールの獲得を研究する．経験的に得られる割付戦略決定ルールは，熟練した計画者にインタビューを繰り返しても，定量的な表現を含むルールを得ることが困難である．

そこで，計画者が割付状態を表示し，割付方法を選択して計画を立案する操作画面を設け，その操作履歴を教師データとして知識獲得する方式を検討する．その教師データには，同じ割付状態でも異なる割付方法を選択する教師データ（以下，あいまい教師データと呼ぶ）と，誤った割付方法を選択する教師データ（以下，誤り教師データと呼ぶ）が含まれることを前提とする．統計解析手法とルール表現を獲得するバージョン空間法を用い，簡潔なルール表現を獲得する統計的集約型知識獲得方式 MAKAM（Multivariate Analytical Knowledge Acquisition Method）を提案する．

1.4 本論文の構成

本論文は 2 章以降を以下のように構成する．

第 2 章では，文献[54-60]に基づき，計画問題定義言語とそのプリコンパイル方式について述べる．まず，生産現場で用いている制約条件や評価指標の数表や数式の表現を示し，その内容を記述する計画問題定義言語を提案する．次に，その記述内容から多重な繰り返し処理を最適化するプリコンパイル方式とそのアルゴリズムと動作例を説明する．そして，提案方式により自動生成するプログラムとプログラマが作成するプログラムの変更容易性と処理性能の比較により，本方式の有効性を示す．

第 3 章では，文献[61-70]に基づき，知識型スケジューリングシステム用プログラム構成方式と記述形式について述べる．まず，対象とする生産スケジューリングの概要を説明する．次に，計画立案手順の変更容易性を確保するためのプログラム構成とその記述形式と，スケジューリング問題用の共通プログラムライブラリを提案し，知識型スケジューリングシステムに実装する．これらにより，計画立案プログラムの変更箇所が局所化され，計画立案手順の変更容易性が確保可能であることを示す．

第 4 章では，文献[71-76]に基づき，統計的集約型知識獲得方式 MAKAM について述べる．まず，誤り教師データやあいまい教師データが割付戦略決定ルールの獲得に及ぼす問題点を示す．次に，判別分析とバージョン空間法を用いる MAKAM を提案する．これらにより，戦略決定ルール獲得の実験結果を示し，計画者が直接修正しやすい簡潔な戦略決定ルールを獲得できることを示す．

第 5 章では，結論として本研究で得られた成果を要約し，今後の課題を述べる．

第2章

計画問題定義言語とプリコンパイル方式

2.1 緒言

本章では，生産計画支援システムの計画問題定義の変更容易性を確保する方式として，計画問題定義記述言語とプリコンパイル方式を提案する．

生産計画の計画問題定義は，計画対象とする製品や装置によって異なり，その制約条件や評価指標の条件評価計算を行う数式や数表は，計画対象の変化に合わせて，生産計画業務を行う計画者が変更し，計画立案に使用する．この数式や数表による条件評価処理は，計画対象の組み合わせ計算となり，大規模な計画問題では，組み合わせの数が増え，膨大な計算量を必要とする．このため，計画問題定義の条件評価処理は，計算機の処理を記述する手続き型言語[77，78]を用いて，システム開発者が処理効率の良い手続き型プログラムを開発し，生産計画支援システムに取り込まれている．

このプログラム開発では，使用する計算機の主記憶領域や処理速度を考慮し，手続き型言語[77，78]のコンパイラ[79]で最適化されない，組み合わせ計算の多重な繰り返し処理の最適化を行う必要がある．最適化された多重な繰り返し処理の手続き型プログラムを，情報処理の専門家ではない計画者が変更することは困難な作業である．システム開発者に依頼すれば，プログラムの変更は可能であるが，変更仕様書や変更プログラムの作成に時間がかかる．

そこで，生産計画業務での数式や数表の表現形式を取り入れ，処理効率を意識することなく，計画者が計画問題定義プログラムを直接変更することが可能な計画問題用の記述言語と，その記述内容から処理効率の良い手続き型プログラムを生成するプリコンパイル方式を検討する．

以下，第 2.2 節では生産計画における計画問題定義の内容を示し，第 2.3 節では計画問題定義言語を提案する．第 2.4 節ではプリコンパイラの必要機能を示し，第 2.5 節ではプリコンパイラの最適化の考え方とそのアルゴリズムを提案する．第 2.6 節では提案アルゴリズムを実装したプリコンパイラの全体構成を示し，第 2.7 節では提案言語を実問題に適用評価するとともに．プログラマが作成したプログラムとの比較からプリコンパイル方式の有効性を示す．

2.2 計画問題定義

本節では，生産計画での計画問題の定義内容と表現形式を説明し，計画者による計画問題定義記述の課題を示す．

2.2.1 生産計画問題の定義内容

生産計画問題は，製品や生産工程の制約条件に従い，生産効率等の評価指標の値を最大化もしくは最小化する問題である．制約条件や評価指標は対象とする製品や生産工程によって異なる．最適な計画を定量的に立案するにはそれらの制約条件や評価指標を定式化することが必要である．

生産計画問題の定式化の例を図 2.1 に示す．この計画問題は生産ロットの製品を製造する生産ラインと製造順序を立案する問題である．なお，生産ロットは一括して製品を製造する単位で，製造する製品品種と製造数が決められている．

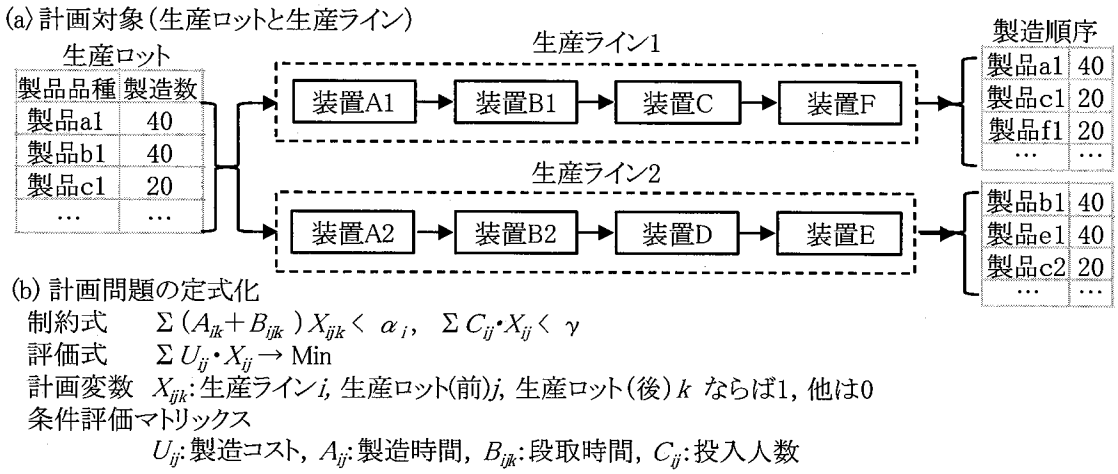

図 2.1 生産計画問題の定式化と計画問題定義

この生産ラインの製造時間は，生産工程の装置と製品品種によって異なる．製造する製品品種を切り替えるには段取時間がかかり，その時間は製品品種や装置によって異なる．この計画問題には生産ラインの稼働時間制約と作業員の人数制約の制約条件と，総製造コストを最小化するという評価指標がある．

この計画問題は，生産ライン i で製造する生産ロット j を決定する組み合わせ計画問題と，生産ライン i 上の製造順序（生産ロット j と生産ロット k の製造順序）を決定する順序計画問題との複合型の整数計画問題である．この問題は制約式の製造時間 A_{ij}, 段取時間 B_{ijk}, 評価式の製造コスト U_{ij} 等の値を計算することにより，計画（0-1 計画問題の解 X_{ijk}）を立案することが可能となる．以下，計画問題を定義する制約式や評価式の製造時間 A_{ij}, 段取時間 B_{ijk}, 製造コスト U_{ij} 等を条件評価マトリックスと呼ぶ．

計画問題の定式化では，考慮すべきすべての条件評価マトリックスを抽出し，その算定方法を正確に定義することが重要である．例えば，図 2.1 の段取時間 B_{ijk} を考慮しなければ，この問題は総製造コストを最小化する生産ロットと製造ラインの組み合わせ計画問題となる．製造順序を考慮する必要がなくなり，最適解を求めることが容易となる．しかしながら，段取時間を考慮しない最適解を求めても，段取時間の違いによって生産ロットの製造順序や生産ラインの入れ替えが起こり，その最適解は使えない．計画立案手順を検討するには，まず，考慮すべき条件評価マトリックスの算定方法を正確に定義することが極めて重要である．

条件評価マトリックスの算定方法は様々な計画対象（生産ロット，生産ライン，作業員等）の組み合わせによって定義される．計画対象の組み合わせの数によって，2 次元や 3 次元の条件評価マトリックスの算定方法が必要となる．それらの算定方法は，計画対象の属性値（製品品種や装置名等の値）によっても異なる．しかも，取り扱う計画対象の要素数（生産ロットの数）が多く，その組み合わせの条件評価には膨大な繰り返し計算を必要とする．条件評価マトリックスの算定方法を定義するとともに，その計算処理には処理効率の良いプログラムも必要である．

本章では，計画問題定義の変更容易性に着目し，このような条件評価マトリックスの算定方法を正確に記述するための計画問題定義言語と，その記述内容から生産計画支援システムに取り込む処理効率の良い手続き型言語の計算機プログラムを自動生成するプリコンパイラを検討する．

2.2.2 計画問題定義の表現形式

生産計画の条件評価マトリックスの算定方法は，計画対象の属性値を用いる数表や数式の形式で記述される．その記述例を表 2.1 と図 2.2 に示す．

(1) 数表による計画問題定義の記述内容

表 2.1 は製造時間を算定するための標準工数表である．標準工数は製品や装置の種類によって異なる単位当たり製造時間を定義するテーブルである．このテーブルは，専門用語や定義内容の違いによって，製造工数や単位製造時間とも呼ばれる．以下，表形式で定義される計画問題定義を業務テーブルと呼ぶ．

表 2.1　業務テーブル（標準工数）の例

装置名 / 製品品種	装置 A1	装置 A2	装置 B1	装置 B2	装置 C1	装置 C2	装置 D1	装置 D2
a	5	3	5	3	3	3	-	—
b	—	—	5	4	5	4	5	5
c	4	3	—	—	5	4	4	3
d	3	4	—	—	—	—	4	3
・・・	・・・	・・・	・・・	・・・	・・・	・・・	・・・	・・・

※単位［分/個］，—：生産工程の製造作業なし（製造時間の割付不要）

生産計画の業務テーブルは，装置の製造時間に関するものだけではなく，作業人数，材料，部品，工具に関する業務テーブルもある．生産計画には計画問題定義の内容によって，このような業務テーブルが多数存在する．

(2) 数式による計画問題定義の記述内容

計画問題定義の内容として製造時間(a)と段取時間(b)の例を図 2.2 に示す．この数式は計画対象の属性や業務テーブルの値を用いて記述される．

(a) 製造時間の例

製造時間＝標準工数×製造数

(b) 段取時間の例

段取時間＝洗浄時間＋調整時間＋部品交換時間

(c) 段取時間の変更例

段取時間＝MAX（洗浄時間，調整時間，部品交換時間）

図 2.2　数式による表現

製造時間(a)は，標準工数（分／個）と製品の生産量（個）の積であることが示されている．この数式により，表 2.1 の標準工数を参照して計画対象の属性値（製品品種，装置名）から製造時間を算定することが可能である．

また，段取時間(b)は 3 つの作業時間の和（作業が順次に実行される場合の合計時間）であることが示されている．このような数式表現で計画問題定義を整理しておくことにより，生産工程の変更（例えば，3 つの段取作業の並列実行への変更）に対して，段取時間の定義の変更（図 2.2 の(b)から(c)）へ迅速に対応可能である．以下，これらの数式を条件評価式と呼ぶ．

生産計画の条件評価式は計画対象の専門用語（図 2.2 の洗浄時間，調整時間等）の変数名を用いて記述される．これらの変数がどの計画対象（製品や装置）に依存するかは計画対象の専門家でなければ判断できない．例えば，調整時間の変化が製品品種の違いによって起こるか，それとも装置種別の違いによって起こるかは計画対象を把握した計画者でなければわからない．しかも，その変化を生産計画支援システムに迅速に取り込むには，計画者自身が条件評価式や業務テーブルを記述し，変更することが求められる．

2.2.3 計画者による計画問題定義記述の課題

生産計画の計画者は計画対象の専門家ではあるが，情報処理の専門家であるとは限らない．情報処理の専門家ではない計画者が計画問題定義の記述内容を変更するには，計画者に対する記述内容の理解容易性や変更容易性が求められる．しかも，計画者の記述内容を計算機に直接取り込むには記述内容（以下，計画問題定義プログラムと呼ぶ）の正確性も記述課題となる．

このため，計画者による計画問題定義プログラムの記述では，

① 計画者の理解容易性を高める生産現場の専門用語を用いた記述

② 計算機の計算順序に制約されない変更容易性のある記述

③ 計画要素間の関係を正確に記述する計画対象の組み合わせの記述

が必要とされる．

一方，条件評価マトリックスの算定には計画対象の属性データ（以下，対象データと呼ぶ）が必要である．計画問題定義プログラムの入力データ（対象データ）や出力データ（条件評価マトリックス）を，生産計画支援システム上で取り扱うにはそれらのデータ属性や領域等の定義が必要である．以下，入出力データの変数名・属性・領域を定義する表を変数名和英対応表と呼ぶ．

2.3 計画問題定義言語

本節では，計画問題定義プログラムの条件評価式と業務テーブルの記述仕様，その入出力データを取り扱うための変数名和英対応表を提案する．

(1) 条件評価式の記述言語と業務テーブルの記述仕様

提案する記述仕様には条件評価式の記述言語と業務テーブルの記述形式がある．条件評価式と業務テーブルの例を図 2.3 に示す．詳細な条件評価式記述の構文グラフを図 2.4 と図 2.5 に示す．

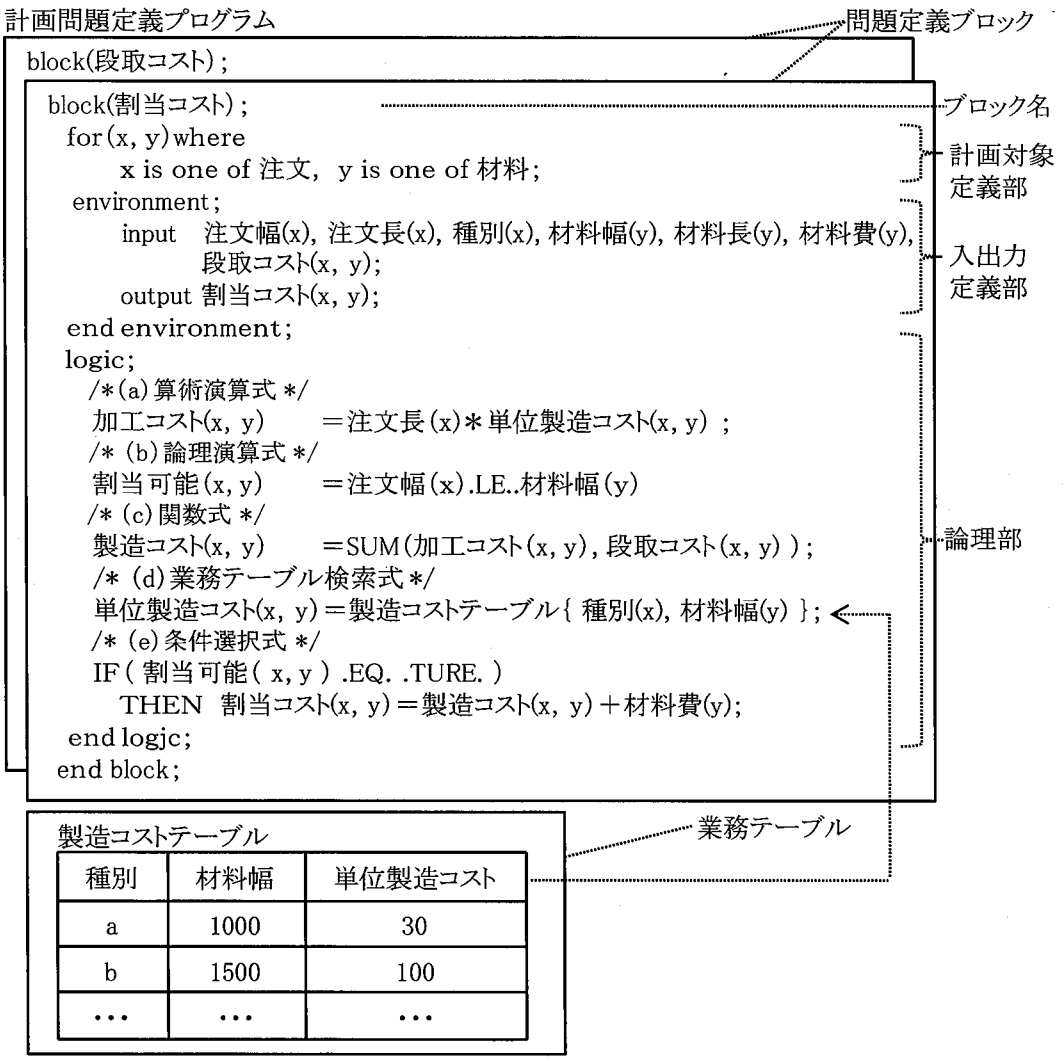

図 2.3 計画問題定義の記述形式例

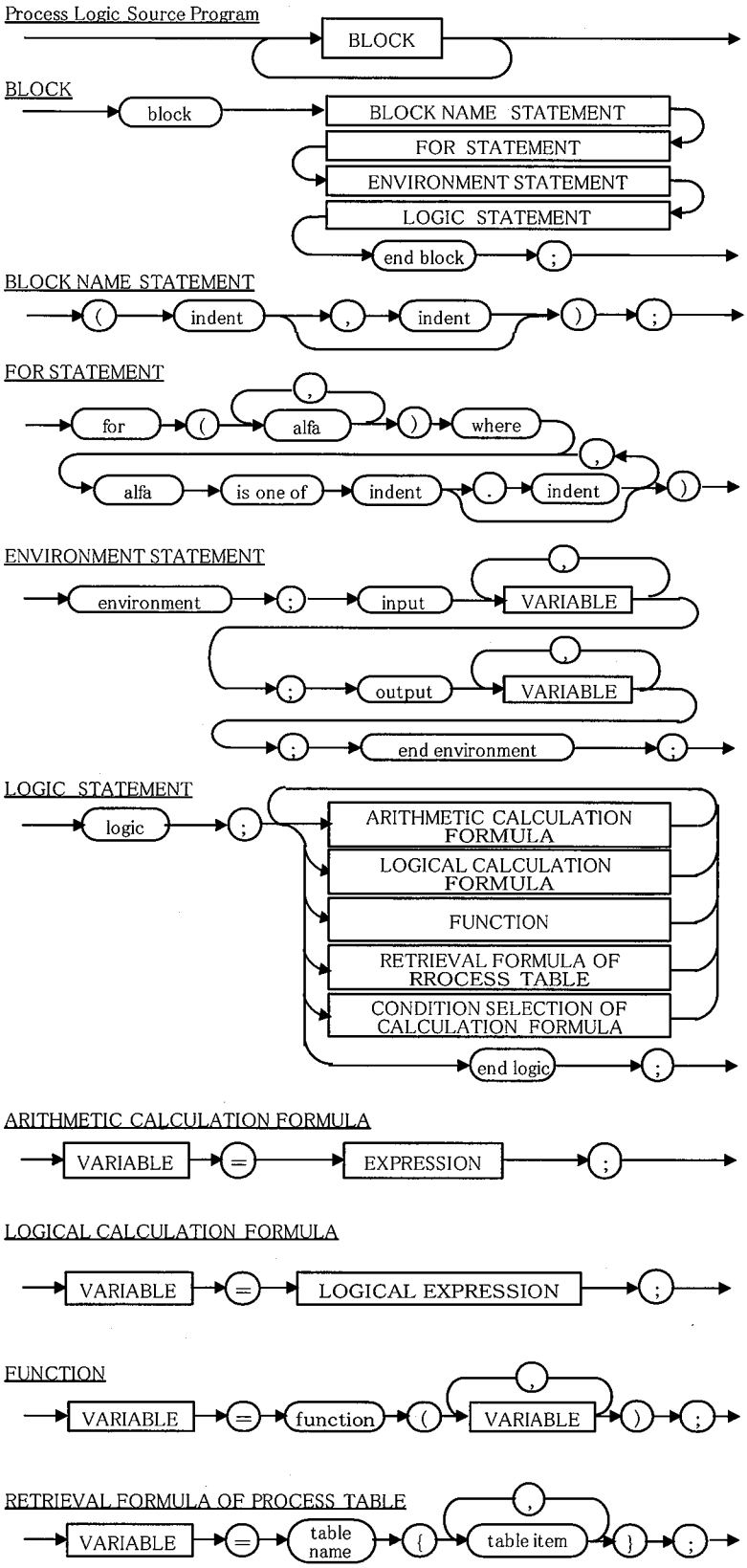

図 2.4　構文グラフ（その１）

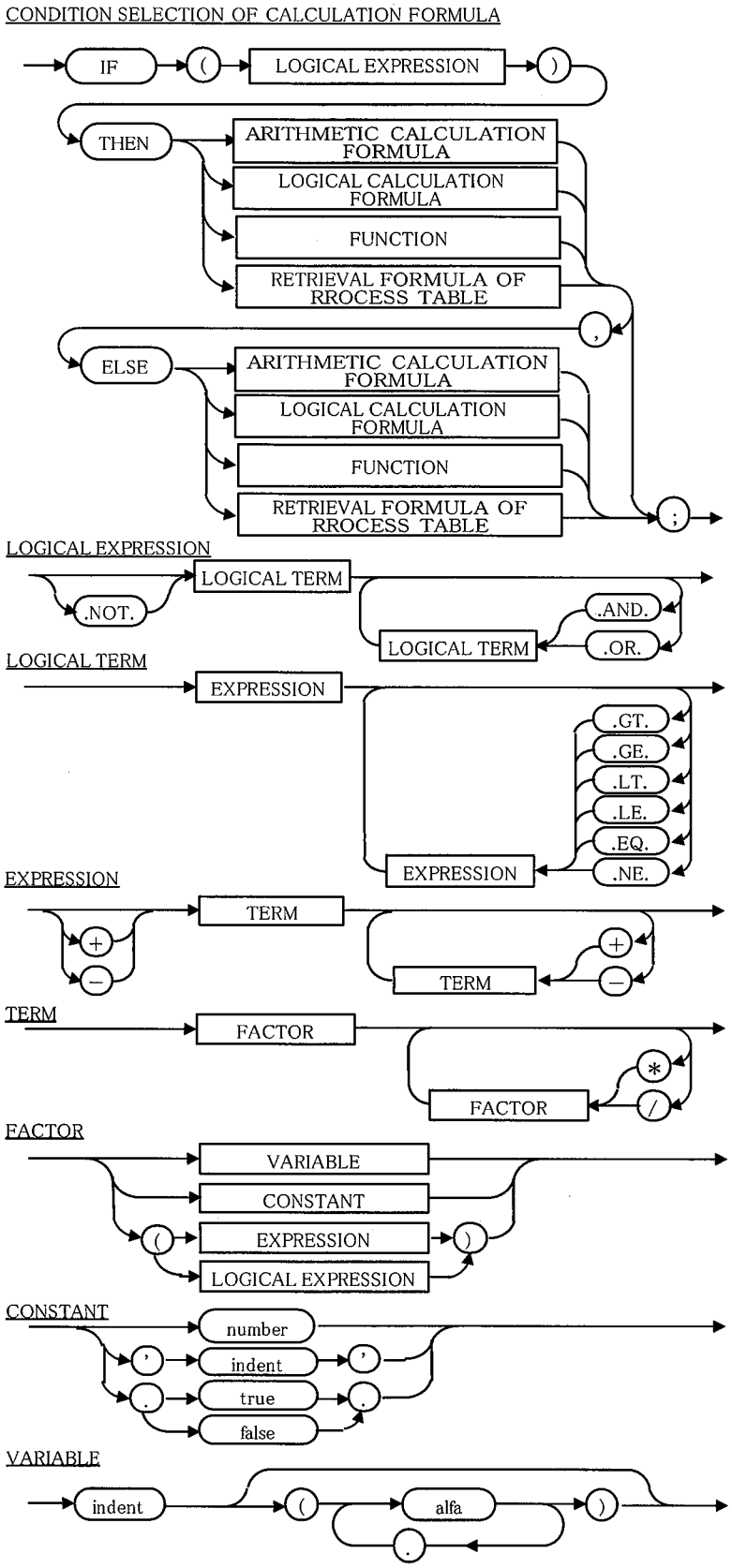

図 2.5　構文グラフ（その２）

計画問題定義プログラムは，記述する単位（ブロック）の名称を定義するBLOCK 文，計画対象を定義する FOR 文，各ブロックの入出力変数を定義するENVIRONMENT 文，条件評価式を記述する LOGIC 文を用いて記述する．LOGIC 文は 5 種類の演算式を用い，ブロックの入力変数から出力変数を計算する条件評価式を記述する．5 種類の演算式を以下に示す．

(a) 算術演算式

目的関数，優先度等の評価式を四則演算によって記述する．

(b) 論理演算式

制約条件や評価指標の条件評価を論理演算によって記述する．

(c) 関数式

算術演算式や論理演算式では記述できない計算を関数によって記述する．

(d) 業務テーブル検索式

標準工数，作業順序等の基礎となる業務テーブルの検索式を記述する．

(e) 条件選択式

条件付きの計算式を記述する．

各演算式は日本語変数名（和名）を用いて記述し，変数名の直後には計画対象の組み合わせを明示化するサフィックス（図 2.3 の x, y）を付記する．

一方，業務テーブル（図 2.3 の表）は，テーブル名，データ項目名，データ部から構成する．業務テーブル検索は，業務テーブル検索式の入力変数名によって検索条件項目が指定され，業務テーブル検索式の出力変数名の値が検索される．

(2) 変数名和英対応表

計画問題定義プログラムの入力変数となる対象データの総称や項目と，出力変数となる条件評価マトリックスの変数名を定義した表が変数名和英対応表である．その例を表 2.2 に示す．変数名和英対応表には計画問題定義プログラムで使用する日本語変数名と計算機処理で取り扱うための英文字変数名，データ属性，領域（データ件数とデータ長）をシステム開発者が定義する．

この対応表に，計画者が日々使用している日本語変数名を登録することで，専門用語を用いた計画問題定義プログラムの記述を可能とする．これにより，それらの変数のデータ属性定義文やデータ入出力文の不要化も図り，計画問題定義プログラムの記述内容を削減する．

表 2.2　変数名和英対応表

定義区分	日本語変数名	英文字変数名	属性	領域
対象データ総称(計画対象)	注文	CHUMON	FILE	100
対象データ項目(入力変数)	注文幅	CHUMON_HABA	数値	3
	注文長さ	CHUMON_NAGASA	数値	4
	種別	SHUBETU	文字	2
対象データ総称(計画対象)	材料	ZAIRYO	FILE	1000
対象データ項目(入力変数)	材料幅	ZAIRYO_HABA	数値	3
	材料長さ	ZAIRYO_NAGASA	数値	4
	材料費	ZAIRYO_HI	数値	4
条件評価マトリックス	割当コスト	WARIATE_COST	数値	3

条件評価式の記述言語には，計算機処理用の変数定義文，入出力文，算定式の計算順序，条件評価マトリックス計算の繰り返しの指定等の計算手続きの記述形式を設けない．計画問題定義プログラムを計算機プログラムに変換する過程で，計算機処理のための計算手続きを補い，条件評価マトリックスの繰り返し処理の最適化を行う．これにより，計算手続きの記述と計算処理の最適化の不要化を図り，情報処理の専門家ではない計画者に対して記述内容の理解容易性や変更容易性を確保する．

2.4　プリコンパイラの機能

本節では，前節の提案言語で記述した計画問題定義プログラムを計算機用の手続き型プログラムへ変換するプリコンパイラの主な機能を説明する．

計画問題定義プログラムは，変数名和英対応表（表 2.2）で定義した対象データの組み合わせから条件評価マトリックスの値を計算するプログラムである．この条件評価マトリックスの計算は対象データの組み合わせ計算であり，条件評価マトリックスのすべての値を求めるには計算の繰り返しが必要である．対象データのデータ件数が増えると，計算量が増大する．このため，対象データの組み合わせ計算を効率よく計算機処理する手続き型プログラムに変換することが必要である．しかも，計画問題定義プログラムの変更が頻繁に発生するため，実行プログラムへの変換時間を短縮することも必要である．

図 2.6 の計画問題定義プログラムの例と，図 2.7 の計画問題定義プログラムの変更から実行までのプロセスを用いて，プリコンパイラの機能を説明する．

なお，図 2.6 の計画問題定義プログラムは，計算式を入力変数から出力変数に向かう計算機の処理順序（以下，計算順序と呼ぶ）で記述した図 2.3 の例に対して，計算順序とは逆の順序で計算式を記述した例である．

計画問題定義プログラム

```
block (割当コスト) ;
  for (x, y) where
      x is one of 注文, y is one of 材料;
  environment;
      input   注文幅(x), 注文長さ(x), 種別(x),
              材料幅(y), 材料長さ(y), 材料費(y);
      output 割当コスト(x, y);
  end environment;
  logic;
    割当コスト(x, y) = 製造コスト(x, y) + 材料コスト(y) ;
    製造コスト(x, y) = 注文面積(x) * 単位製造コスト(x, y);
    注文面積(x)     = 注文幅(x) * 注文長さ(x);
    材料コスト(y)    = 単位材料コスト(y) * 12;
    単位材料コスト(y) = 材料長さ(y) * 材料費(y);
    単位製造コスト(x, y)
              = 製造コストテーブル{種別(x), 材料幅(y)} ;
  end logic;
end block;
```

業務テーブル

製造コストテーブル

種別	材料幅	単位製造コスト
a	1000	30
b	1500	100
・・・	・・・	・・・

図 2.6　計算式の記述順序が計算順序と異なる計画問題定義プログラムの例

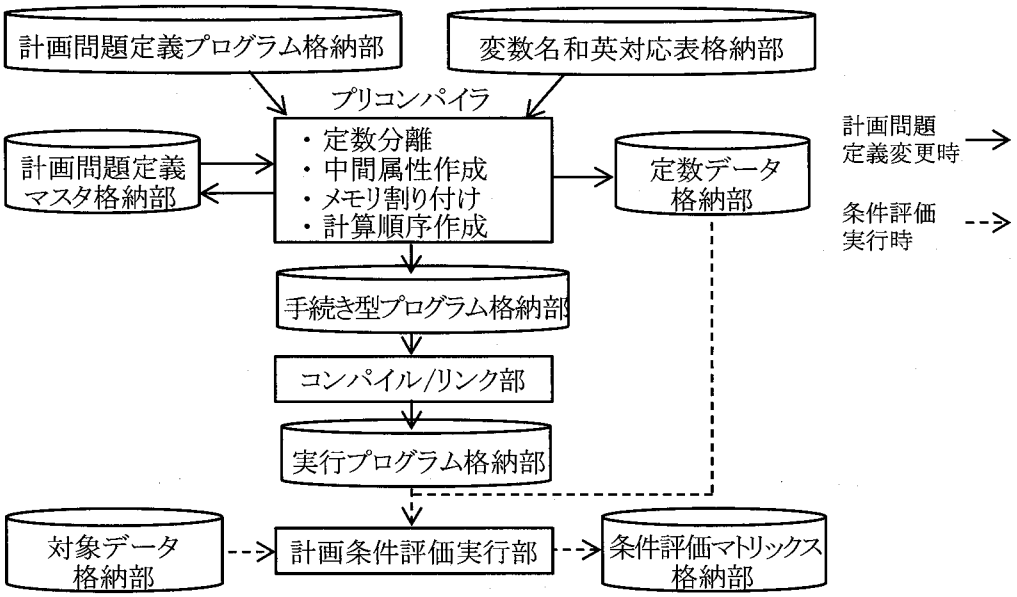

図 2.7　計画問題定義プログラムの変更と実行のプロセス

(1) 定数分離機能

計画問題定義プログラム（図 2.6 の条件評価式と業務テーブル）には数多くの定数を含む．計画問題定義の変更はその定数に対する変更が最も多い．定数の変更だけであれば，手続き型プログラムの生成，コンパイル，リンクを行う必要はない．そこで，計画問題定義プログラムから定数を分離，定数データ格納部に格納し，その定数をパラメータとして読み込む手続き型プログラムを生成する定数分離機能をプリコンパイラに設ける．

この機能は，変更前の計画問題定義プログラムを計画問題定義マスタ格納部に登録し，マスタとの比較により，計画問題定義プログラムの変更内容を判別し，定数変更に対するコンパイル・リンクのオーバヘッドを削減する機能である．しかも，定数をプログラムから分離し，実行時に定数を主記憶領域へ直接ロードする手続き型プログラムを生成することによって，プログラム記憶領域から定数をロードする手続き型プログラムよりも処理時間の短縮が可能となる．

(2) 中間属性推定機能

計画問題定義プログラムの計算式は入力変数と出力変数だけでなく，中間変数（図 2.6 の製造コスト，材料コスト等）を用いて記述することが可能である．これらの変数を計算機で取り扱うには変数定義（データ属性とデータ長の定義）が必要である．入力変数（対象データ）と出力変数（条件評価マトリックス）は，他のプログラムとのインターフェースとなることから，変数名和英対応表（表 2.2）でデータ属性とデータ長を定義する．一方，中間変数は，プリコンパイラに中間変数の属性推定機能を設けることで，中間変数の変数定義文の記述不要化が可能とする．

なお，中間変数の属性推定は，入力変数のデータ属性と中間変数の計算式の種別（算術演算式，論理演算式等）から中間変数のデータ属性（数値，論理値，文字列等）を推定し，入力変数のデータ長から中間変数のデータ長（整数：2 バイト，実数：4 バイト等）を推定する．

(3) メモリ割り付け機能

計画問題定義プログラムの記述内容に従い，すべての変数に対象データの件数に対応した多次元の配列型のメモリ（以下，配列型メモリと呼ぶ）を割り付けると，プログラム全体では大量のメモリ量が必要となる．

例えば，図 2.6 の中間変数（例えば，製造コスト(x, y)，実数 4 バイト）に，対象データの件数（注文 100 件，材料 1,000 件）に対応した配列型メモリを割り付けると，0.4M バイトのメモリ量を必要とする．注文の件数が 1,000 件に増加すると，1 つの変数でも必要なメモリ量は 4M バイトに増加する．

使用可能なメモリ量が少ない計算機環境では，計画問題定義プログラムのすべての変数に配列型メモリを割り付けることができない．ターゲット計算機の使用可能なメモリ量（以下，許容メモリ量と呼ぶ）を超えない手続き型プログラムに変換する必要がある．このため，プリコンパイラでは，許容メモリ量を超えない範囲で，組み合わせ計算の繰り返し処理に有効な変数に対し，配列型メモリを割り付けるメモリ割り付け機能が必要である．この機能は，計算式の計算順序にも関係するため，この実現方式は次節で詳細に説明する．

(4) 計算順序作成機能

計画問題定義プログラムは，計画者が理解もしくは記述しやすい順序で，計算式を記述することが可能である．このため，プリコンパイでは，計算機が処理可能な順序（以下，計算順序と呼ぶ）に計算式を並び換える必要がある．

図 2.6 の例は，出力変数から入力変数に向かう順序（割当コスト，製造コスト，材料コスト，注文面積，単位材料コスト，単位製造コスト）で，割当コストを計算する 6 つの計算式を記述した例である．割当コストのコスト構造を検討する計画者にとって理解容易な順序であるが，計算機の計算順序とは異なる．

このため，プリコンパイラでは，「割当コストの計算は製造コストと材料コストの計算の後に行う」という計算の半順序を用いて，計算式の計算順序を作成する．この半順序関係に制約されない計算式（例えば，製造コストと材料コストの計算式）の計算順序は，どちらの計算式を先に計算することも可能である．プリコンパイラは，効率的な組み合わせ計算が可能な計算順序を作成して，すべての計算式の計算順序（全順序）を指定する手続き型プログラムを作成する．

また，計画問題定義プログラムの変数名の後に記述されているサフィックス（x, y）は対象データ（x：注文，y：材料）の組み合わせで計算することを示している．繰り返し処理の順序を指定している記述ではない．プリコンパイラは，効率的な組み合わせ計算が可能な繰り返し処理の順序を作成することが必要である．この機能は，メモリ割付機能と密接に関係するため，この実現方式は次節で詳細に説明する．

2.5 メモリ割付による計算量の削減方式

2.5.1 プリコンパイラの最適化機能

条件評価の組み合わせ計算では繰り返しの順序や計算式の計算順序によって必要なメモリ量と計算量の関係が変化する．以下，繰返計算処理におけるメモリ量と計算量の関係を示し，プリコンパイラの最適化機能を説明する．

(1) 計算量最小プログラミングとメモリ量最小プログラミング

計画問題定義プログラムから計算機プログラムを作成するアプローチには，計算量最小プログラミングとメモリ量最小プログラミングがある．計画問題定義プログラムのすべての変数の値を格納する配列型メモリの割り付けが可能であれば，不要な繰返計算を含まない計算量最小プログラムが作成可能である．一方，計算式のすべてを多重な繰り返しの中で実行し，計算結果だけを 2 次記憶装置へ逐次出力するならば，変数に配列型メモリを割り付ける必要がないメモリ量最小プログラムが作成可能である．

3 次元の計画問題定義プログラムの例（図 2.8）を用い，その計算量最小プログラムの例を図 2.9 に示す．なお，図 2.8 は，3 計画対象（x：注文，y：材料，z：装置），12 入力変数，6 中間変数（変数 1 ~変数 6），1 出力変数，7 条件評価式の例である．

```
block(出力変数);
 for (x, y, z) where  x is one of 注文, y is one of 材料, z is one of 装置;
 environment; input … … …;  output …; end environment;
 logic;
   出力変数(x, y, z) = 変数1(x) ＋変数3(z) ＋変数4(x, y) ＋変数5(x, z) ＋変数6(y, z);
   変数6(y, z)       = 変数2(y) ＋変数3(z) ＋入力変数24(y) ＋入力変数34(z);
   変数5(x, z)       = 変数1(x) ＋変数3(z) ＋入力変数14(x) ＋入力変数33(z);
   変数4(x, y)       = 変数1(x) ＋入力変数13(x)＋入力変数23(y);
   変数3(z)          = 入力変数31(z)＋入力変数32(z);
   変数2(y)          = 入力変数21(y)＋入力変数22(y);
   変数1(x)          = 入力変数11(x)＋入力変数12(x);
 end logjc;
end block;
```

図 2.8　3 次元の計画問題定義プログラムの例

```
                    (プログラム)                                    … 繰返実行回数   ー必要計算回数
DO x = 1 TO Nx;                                  /* 計算ステップ1 */    …Nx
  READ(入力変数11(x), 入力変数12(x), 入力変数13(x), 入力変数14(x));                    ーNx
END;
DO x = 1 TO Nx;                                  /* 計算ステップ2 */    …Nx
  変数1(x) =入力変数11(x)+入力変数12(x);                                              ーNx
END;
DO y = 1 TO Ny;                                  /* 計算ステップ3 */    …Ny
  READ(入力変数21(y), 入力変数22(y), 入力変数23(y), 入力変数24(y));                    ーNy
END;
DO y = 1 TO Ny;                                  /* 計算ステップ4 */    …Ny
  変数2(y) = 入力変数21(y)+入力変数22(y);                                             ーNy
END;
DO z = 1 TO Nz;                                  /* 計算ステップ5 */    …Nz
  READ(入力変数31(z), 入力変数32(z), 入力変数33(z), 入力変数34(z));                    ーNz
END;
DO z = 1 TO Nz;                                  /* 計算ステップ6 */    …Nz
  変数3(z) =入力変数31(z)+入力変数32(z);                                              ーNz
END;
DO x = 1 TO Nx; DO y = 1 TO Ny;                  /* 計算ステップ7 */    …NxNy
  変数4(x, y) = 変数1(x)+入力変数13(x)+入力変数23(y);                                 ーNxNy
END; END;
DO x = 1 TO Nx; DO z = 1 TO Nz;                  /* 計算ステップ8 */    …NxNz
  変数5(x, z) = 変数1(x)+変数3(z)+入力変数14(x)+入力変数33(z);                        ーNxNz
END; END;
DO y = 1 TO Ny; DO z = 1 TO Nz;                  /* 計算ステップ9 */    …NxNz
  変数6(y, z) = 変数2(y)+変数3(z)+入力変数24(y)+入力変数34(z);                        ーNxNz
END; END;
DO x = 1 TO Nx; DO y = 1 TO Ny; DO z = 1 TO Nz;  /* 計算ステップ10 */   …NxNyNz
  出力変数(x, y, z) = 変数1(x)+変数3(z)+変数4(x, y)+変数5(x, z)+変数6(y, z);          ーNxNyNz
END; END; END;
DO x = 1 TO Nx; DO y = 1 TO Ny; DO z = 1 TO Nz;  /* 計算ステップ11 */   …NxNyNz
    WRITE(x, y, z, 出力変数(x, y, z));                                                ーNxNyNz
END; END; END;
```

図 2.9　計算量最小プログラム

計算量最小プログラム（図 2.9）は，変数 i の計算式ごとに計画対象の組み合わせで繰り返し処理を行う計算ステップ S_i を設けて，計算式の計算順序で実行するプログラムである．図 2.9 のプログラムの右側に繰返実行回数と各計算式の必要繰返回数を示す．

このプログラムは，計算式ごとの計算ステップ S_i で，必要な繰り返しの処理だけを実行することにより，不要な繰り返し処理の実行を含まないプログラムである．しかしながら，計算ステップ S_i ごとに変数 i の計算結果（入力データを含む）を M_i 次元の配列型メモリ（計算ステップ 1 の入力変数 11(x) 等）に格納し，次の計算ステップ S_{i+1} 以降にその計算結果を受け渡す必要がある．このため，計画問題定義プログラムに含まれる変数の総数 N_v 個に対し，計算量最小プログラムでは N_v 個の変数のすべてに計算結果を格納する M_i 次元の配列型メモリ（入力変数 11(x)，変数 1(x)，変数 4(x, y) 等）が必要であり，計算量は必要最小であるが，必要とするメモリ量は最大となる．

計画要素の数が N_x= 1,000, N_y =100, N_z =10, 1 変数のメモリを 4 バイトとした場合の必要メモリ量と繰返実行回数を表 2.3 に示す．図 2.9 のプログラムの実行には，約 4.5M バイトが必要である．N_x, N_y, N_z がともに 1,000 件であれば，4G バイト以上のメモリが必要となる．

表 2.3　計算量最小プログラム(図 2.9)のメモリ量と繰返実行回数

評価項目	評価式　　＝　評価値
①必要メモリ量	(5Nx＋5Ny＋5Nz＋NxNy＋NxNz＋NyNz＋NxNyNz)×4 ＝ 4,466,200
②繰返実行回数※	Nx ＋Ny ＋Nz ＋NxNy＋NxNz＋NyNz＋NxNyNz　　＝ 1,112,110

計画対象の要素数 Nx=1,000, Ny=100, Nz=10, ※比較のため入出力処理を除く

この計算量最小プログラミングは，変数 i の次元数 M_i, 計画対象 k の要素数 N_k, 変数の総数 N_v が少ない場合，実行可能なプログラムの生成が可能である．しかしながら，それらの数が増加すると，必要とするメモリ量が増大し，許容メモリ量を超え，実行可能なプログラムの生成が不可能となる．このため，必要メモリ量を抑えたプログラミングが必要である．メモリ量最小プログラムの例を図 2.10 に示す．

メモリ量最小プログラム (図 2.10) は，入力変数にデータを読み込みながら，多重な繰返計算の中ですべての計算を行うプログラムである．このプログラムでは，出力変数の計算結果を 2 次記憶装置に逐次出力するため，すべての配列型メモリ（図 2.9 の入力変数 11(x)，変数 1(x), 変数 4(x, y) 等）が不要である．

```
                    (プログラム)                          … 繰返実行回数   －必要計算回数
DO x = 1 TO Nx;                                          …Nx
  READ(入力変数11, 入力変数12, 入力変数13, 入力変数14);                    －Nx
 DO y = 1 TO Ny;                                         …NxNy
    READ(入力変数21, 入力変数22, 入力変数23, 入力変数24);                  －Ny
    DO z = 1 TO Nz;                                      …NxNyNz
      READ(入力変数31, 入力変数32, 入力変数33, 入力変数34);                 －Nz
      変数1＝入力変数11＋入力変数12;                                        －Nx
      変数2=入力変数21+入力変数22;                                         －Ny
      変数3=入力変数31＋入力変数32;                                        －Nz
      変数4=変数1＋入力変数13＋入力変数23;                                  －NxNy
      変数5=変数1＋変数3＋入力変数14＋入力変数33;                           －NxNz
      変数6=変数2＋変数3＋入力変数24＋入力変数34;                           －NyNz
      出力変数 =変数1＋変数3 ＋変数4 ＋変数5 ＋変数6;                       －NxNyNz
      WRITE(x, y, z, 出力変数);
END; END; END;
```

図 2.10　メモリ量最小プログラム

このため，メモリ量最小プログラミングでは，変数の総数 N_v が爆発的に増大しない限り，許容メモリ量を超えることがない．必要メモリ量と計算式の繰返実行回数を表 2.4 に示す．

表 2.4　メモリ量最小プログラム(図 2.10)のメモリ量と繰返実行回数

評価項目	評価式	＝　評価値
①必要メモリ量	(5＋5＋5＋1＋1＋1＋1)×4	＝　76
②繰返実行回数※	7NxNyNz	＝ 7,000,000
③必要繰返回数	Nx ＋Ny ＋Nz ＋NxNy＋NxNz＋NyNz＋NxNyNz	＝ 1,112,110
④不要繰返回数	繰返実行回数－必要繰返回数	＝ 5,887,890

計画対象の要素数 Nx=1,000, Ny=100, Nz=10, ※比較のため入出力処理を除く

このプログラミングでは，計画対象 k の要素数 N_k，変数の総数 N_v が増加しても，実行可能なプログラムの生成が可能である．特に，要素数 N_k や変数の総数 N_v が多くなる生産計画では，すべての変数に配列型メモリを割り付けることが困難であり，このプログラミングが有用となる．しかしながら，このプログラミングでは不要な繰り返しの計算実行を数多く含む．例えば，変数 1 の計算は計画対象 x に依存し，必要計算回数が N_x 回であるにもかかわらず，$N_xN_yN_z$ の多重な繰り返しの中で実行される．このため，変数 1 の計算だけでも $N_xN_yN_z-N_x$ 回の不要な繰返計算が実行される．この不要な繰り返し回数を不要繰返回数と呼ぶ．計画対象の要素数 N_x= 1,000, N_y=100, N_z=10 の場合でも，図 2.10 のプログラムには約 588 万回の不要繰返回数が含まれる．

(2) メモリ不要最適化とメモリ必要最適化

手続き型プログラムのコンパイラは，プログラマが作成した計算手続き（プログラム）に対して，繰り返し（ループ）の中では不変な計算をループ外に移動する等の最適化を行う．例えば，メモリ量最小プログラム（図 2.10）の変数 1 の計算式は一番内側の z ループには不変な式であり，z ループの外へ移動可能である．さらに，変数 1 の計算式は内側から 2 番の y ループにも不変な計算であり，y ループの外にも移動可能である．その結果，変数 1 の計算式は一番外側の x ループだけで繰り返される計算ステップに移動され，変数 1 の計算は必要以上の繰返計算の実行が行われないプログラムとなる．この計算式の移動

では，変数1に計算結果を格納する配列型メモリを割り付ける必要はない．以下，この移動をメモリ不要最適化と呼ぶ．

これに対して，変数2の計算はyだけに依存する入力変数21，22より計算される値であり，必要繰返回数はN_y回である．変数2の計算式もzループには不変な計算式としてzループの外に移動可能であるが，yループには不変な計算式ではなく，手続き型言語のコンパイラでは，yループの外へ移動不可である．このため，変数2の計算を多重ループのyループ内側の繰返計算ステップで実行するとN_xN_y回の計算実行が繰り返される．この繰返実行回数をN_y回だけとするには，多重ループの外側に移動する必要がある．この最適化には，多重な繰返計算から独立した必要回数と同じ繰返計算を実行する計算ステップ（独立ループ）を作り，変数2の計算結果を格納する配列型メモリを割り付けることが必要である．以下，この配列型メモリを割り付けて計算式を移動することをメモリ必要最適化と呼ぶ．

(3) メモリ割付による計算量の削減

手続き型プログラムのコンパイラは，プログラマが作成したプログラムのメモリ割付や繰返計算を変更する最適化を行わない．このため，プログラマは，計算量最小プログラムの必要メモリ量が許容メモリ量を超える場合，計画対象kの繰返計算や変数iへ配列型メモリの割り付けを検討し，メモリ不要最適化とメモリ必要最適化の使い分けを行い，許容メモリ量を超えない範囲で，計算量が最小となるプログラムを作成する．最適なプログラムの例を図2.11に示す．

図2.11の最適なプログラムは，メモリ量最小プログラム(図2.10)からメモリ不要最適化を用いて，変数1，変数4の計算式を出力変数の多重ループの繰返計算の中で必要回数と同じ繰返し回数となる計算ステップに移動している．一方，変数2，変数3，変数6の計算式は，メモリ必要最適化を用いて，出力変数の多重ループから独立した計算ステップに移動している．また，変数5は，多重ループの中で，必要繰返回数と同じ繰返しとなる新しいループ（以下，従属ループと呼ぶ）作り，必要なメモリを割付，計算式を移動している．さらに出力変数の値を2次記憶装置へ逐次出力することでメモリ量を削減している．

必要メモリ量と繰返実行回数を表2.5に示す．このプログラムの計算量は計算量最小プログラムと同じ繰返実行回数であり，不要繰返のない計算量最小プログラムである．図2.9（計算量最小プログラム）の必要メモリ量は約4.5M

バイトであるのに対し，配列型メモリ（図 2.10 の入力変数 23(y), 33(z), 34(z), 変数 3(z)，変数 5(z)，変数 6(y, z)）を最適に割り付けることにより，その必要メモリ量を 4.17K バイトに削減可能である．出力変数の値を 2 次記憶装置へ逐次出力することによる必要メモリ量の削減効果が大きいが，出力変数以外の 2 次元の配列型メモリは変数 6(y, z)のみに割り付けることで計算が可能である．出力変数のメモリ量を除いて比較すると，計算量最小プログラムの必要メモリ量は 466K バイトであるのに対し，この最適なプログラムの必要メモリ量は 4.17K バイトであり，計算量最小プログラムの必要メモリ量の約 100 分の 1 である．

```
                    (プログラム)                          … 繰返実行回数   －必要計算回数
DO z = 1 TO Nz;                                          …Nz
  READ(入力変数31, 入力変数32, 入力変数33(z), 入力変数34(z));              －Nz
  変数3(z) = 入力変数31＋入力変数32;                                       －Nz
END;
DO y = 1 TO Ny;                                          …Ny
  READ (入力変数21, 入力変数22, 入力変数23(y), 入力変数24);               －Ny
  変数2 = 入力変数21+入力変数22;                                           －Ny
  DO z = 1 TO Nz;                                        …NyNz
    変数6(y, z) =変数2 ＋変数3(z)＋入力変数24＋入力変数34(z);              －NyNz
END; END;
DO x = 1 TO Nx;                                          …Nx
  READ(入力変数11, 入力変数12, 入力変数13, 入力変数14);                   －Nx
  変数1 = 入力変数11＋入力変数12;                                          －Nx
  DO z = 1 TO Nz;                                        …NxNz
    変数5(z)=変数1＋変数3(z)＋入力変数14＋入力変数33(z);                  －NxNz
  END;
  DO y = 1 TO Ny;                                        …NxNy
    変数4 = 変数1 ＋入力変数13＋入力変数23(y);                            －NxNy
    DO z = 1 TO Nz;                                      …NxNyNz
      出力変数=変数1＋変数3(z)＋変数4＋変数5(z) ＋変数6(y, z);            －NxNyNz
      WRITE(x, y, z, 出力変数);                                          －NxNyNz
END; END; END;
```

図 2.11　最適なプログラムの例

表 2.5　最適なプログラム(図 2.11)のメモリ量と繰返実行回数

評価項目	評価式	＝	評価値
①必要メモリ量	(5＋4＋Ny＋3＋2Nz＋1＋Nz＋NyNz＋1)×4	＝	4,176
②繰返実行回数※	Nx ＋Ny ＋Nz ＋NxNy＋NxNz＋NyNz＋NxNyNz	＝	1,112,110
③必要繰返回数	Nx ＋Ny ＋Nz ＋NxNy＋NxNz＋NyNz＋NxNyNz	＝	1,112,110
④不要繰返回数	繰返実行回数－必要繰返回数	＝	0

計画対象の要素数 Nx=1,000, Ny=100, Nz=10, ※比較のため入出力処理を除く

2.5.2 メモリ割り付け計算量最小化問題

プリコンパイラの最適化問題は，配列型メモリの割り付けに許容される最大メモリ量 W_{max} 以下で，計算量 E（計算と入力の繰返実行回数）を最小化する問題である．本問題の制約式と評価式を以下に示す．

(1) 制約式

(a) メモリ制約：配列型メモリのメモリ量 $W \leqq W_{max}$

$W = \sum W_i$ （変数 $i = 1 \sim N_v$）

$W_i = d_i \prod_{P_{ik}V_{ik}=1} N_k P_{ik} V_{ik}$ （計画対象 $k = 1 \sim M_{max}$）

M_{max}：計画対象 k の数（計画対象の組み合わせ数の最大数）

N_v：変数 i の総数，d_i：変数 i の 1 要素を格納するメモリ量，

N_k：計画対象 k の要素数（対象データごとのデータ件数），

P_{ik}：変数 i の必要繰返（計画対象 k に対する繰返計算が必要である場合 1，不要である場合 0）

V_{ik}：変数 i の配列型メモリ割付（計画対象 k に対する配列型メモリを割り付ける場合 1，割り付けない場合 0）

(b) 計算順序制約：変数 i の計算ステップ $S_i > S_j$ 但し $C_{ij}=1$ となる変数 ij

C_{ij}：構造マトリックス変数 i（左辺）の計算式の実行に変数 j（右辺）の値が必要である場合 1，不要である場合 0）

(2) 評価式

計算量の最小化：実行される計算量 $E \rightarrow \mathrm{Min}$

$E = \sum E_i$ （変数 $\mathrm{i} = 1 \sim N_v$）

$E_i = e_i \prod_{F_{ik}=1} N_k F_{ik}$ （計画対象 $\mathrm{k} = 1 \sim M_{max}$）

e_i ：変数 i の計算式の単位計算量，

F_{ik}：変数 i の計算の繰返実行（計算ステップ S_i での計画対象 k による繰返計算をする場合 1，しない場合 0）

この問題の探索空間（変数 i の配列型メモリ割付 V_{ik}）は，N_v 個の変数 i に M_i 次元の配列型メモリを割り付ける組み合わせ，2 の $\sum M_i (i = 1 \sim N_v)$ 乗となる．図 2.7 の 3 次元の計画問題定義の場合，入出力変数も含める 1 次元の変数が 15，2 次元の変数が 3，3 次元の変数が 1 で，2^{24} 通り（約 10^7 通り）である．変数の総数 N_v が 30，平均次元数 M_{ave} が 2 の場合，2^{60} 通り（約 10^{18} 通り）となる．

この問題は，変数 i の配列型メモリ割付 V_{ik} と計算式の繰返実行 F_{ik} の計算ステップ S_i を決定し，その計算順序に従ってプログラムを生成する問題である．計算量最小プログラムは配列型メモリ割付 V_{ik} ＝必要繰返 P_{ik}，繰返実行 F_{ik}＝必要繰返 P_{ik} とする解で，メモリ量最小プログラムは配列型メモリ割付 V_{ik} ＝0，繰返実行 F_{ik}＝1 とする解である．この２つの解の間に，メモリ制約と計算順序制約を満たし，計算量 E が最小となる解（図 2.11）が存在する．

プリコンパイラの最適化では，配列型メモリ割付 V_{ik} の膨大な組み合わせの中から，メモリ制約と計算式の計算順序制約を満足し，全体の計算量 E が最小となるプログラム（メモリ割付，繰返計算，計算順序）を探索する．

2.5.3 解探索の考え方

プリコンパイラには計画対象の要素数 N_k や計画問題定義プログラムの変数の総数 N_v が増加しても，最大メモリ量 W_{max} を超えない実行可能なプログラムを出力することが求められる．以下，メモリ不要最適化とメモリ必要最適化を使い分け，最大メモリ量 W_{max} 内で計算量を最小とする解探索の考え方を説明する．

(1) 解探査空間の絞り込み

メモリ不要最適化とメモリ必要最適化の使い分けでは計画問題定義のすべての変数の配列型メモリ割付 V_{ik} を検討する必要はない．多重ループの繰返計算，計画問題定義の計算構造，計画対象の要素の数によって，メモリ不要最適化とメモリ必要最適化が適用可能な変数は異なるが，その差異に関わらず，同じ計画対象の組み合わせの計算の中で中間結果となる変数には配列型メモリを割り付ける必要はない．以下，配列型メモリを割り付ける必要がない変数を計画問題定義の構造マトリックスと条件評価マトリックス生成ツリーを用いて説明する．

図 2.6 の計画問題定義の構造マトリックス C を表 2.6 に示す．構造マトリックス C は計算式の入出力関係を 0，1 で示す 2 値マトリックスである．変数 i の計算式で入力となる変数 j がある場合 C_{ij} を 1 としそれ以外を 0 とする．これによって，任意の変数 i（例えば，割当コスト）を計算する前に C_{ij} が 1 となる変数 j（例えば，製造コストと材料コスト）の計算が実行されていなければならないという計算の半順序を示す．この構造マトリックス C を用いて，計画問題定義プログラムの計算構造を表現した条件評価マトリックス生成ツリー T を図 2.12 に示す．

条件評価マトリックス生成ツリーTは，変数をノード，変数間の参照関係をアークで表現する．各ノードiに，その変数iの1件当たりデータを格納するメモリ量d_i，計算式の単位計算量e_i，どの計画対象kで繰り返すかを示す必要繰返p_iのラベルつける．必要繰返p_iは，計画対象kの繰り返しが必要である場合は1，必要でない場合は0とし，異なるp_iを持つ変数間のアークは点線とする．同一のp_iを持つ変数は同じ計画対象kによる繰返しが必要である．

表 2.6　構造マトリックス Cの例

左辺i ＼ 右辺j	割当コスト	製造コスト	注文面積	材料コスト	単位材料コスト	単位製造コスト	注文幅	注文長さ	種別	材料幅	材料長さ	材料費
割当コスト	0	1	0	1	0	0	0	0	0	0	0	0
製造コスト	0	0	1	0	0	1	0	0	0	0	0	0
注文面積	0	0	0	0	0	0	1	1	0	0	0	0
材料コスト	0	0	0	0	1	0	0	0	0	0	0	0
単位材料コスト	0	0	0	0	0	0	0	0	0	0	1	1
単位製造コスト	0	0	0	0	0	0	0	0	1	1	0	0

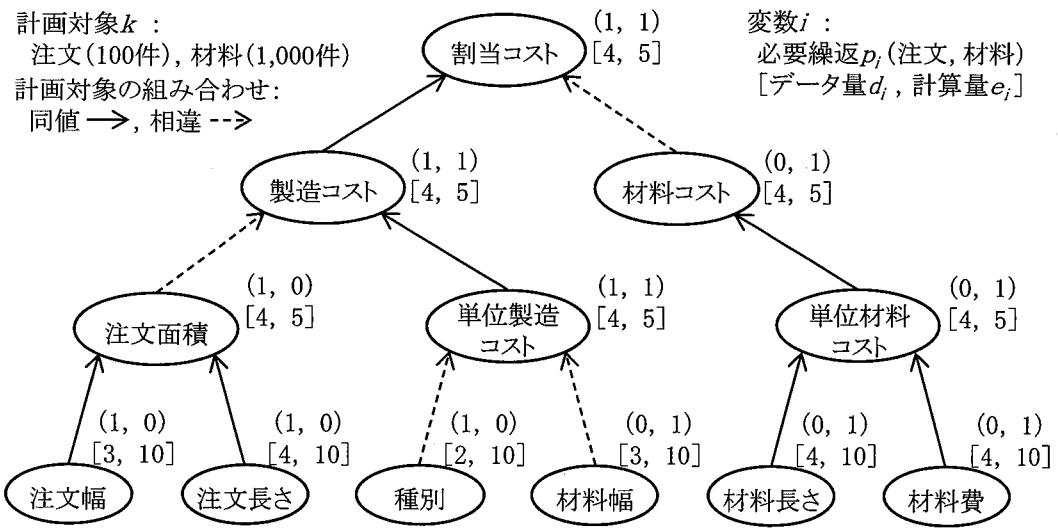

図 2.12　条件評価マトリックス生成ツリーの例

図 2.12 において，必要繰返p_iの値（1, 0）を持つ変数は，計画対象（注文）の要素数（100 件）に対する 100 回の繰返しが必要であり，必要繰返p_iの値（1, 1）を持つ変数iは，計画対象（注文）の要素数（100 件）と計画対象（材料）の要素数（1,000 件）の組み合わせに対する 10 万回の繰返しが必要である．

一方，同じ必要繰返となり，同じループの中で計算される変数で，他の繰返し計算から参照されない変数は配列型メモリを割り付ける必要はない．例えば，必要繰返 p_i の値が（0, 1）となる変数の中で，材料長さ，材料費，単位材料コストの変数は材料長さと材料費の値を入力，単位材料コストと材料コストの計算を行い，その計算結果だけを配列型メモリに格納すれば，これ以降の計算ステップでは材料コスト以外の配列型メモリは不要である．この計算ステップで中間結果となる変数（例えば，材料長さ，材料費，単位材料コスト）はメモリ必要最適化とメモリ不要最適化の如何に関わらず，配列型メモリが不要である．以下，他の繰返し計算から参照される変数をルート変数 r と呼び，1つのルート変数 r と他の繰返し計算から参照されない変数からなる計算構造（部分集合）をサブツリー T_r と呼ぶ．なお，同じ計画対象の組み合わせを持つ変数の集合であっても，他の計画対象の組み合わせの計算から参照される変数は別のサブツリー T_r とする．図 2.12 の生成ツリーは，ルート変数 r が割当コスト，材料コスト，注文面積，種別，材料となるサブツリー T_r から構成される．

このように，配列型メモリ割付変数の解探索では計算式の移動をサブツリー T_r 単位で考え，ルート変数 r に対するメモリ割付を行うことで，解探索空間を削減する．図 2.12 の例では 12 変数のメモリ割付問題から 5 つサブツリー T_r のルート変数 r のメモリ割付問題となり，探索空間が 2^{12} から 2^5 に削減される．

(2) サブツリー T_r の計算順序制約

この問題の解法を考える上で，任意のサブツリー T_r の移動は計算順序制約 C_{ij} に従い，変数 i の計算に必要とする変数 j の計算を先に実行することが必要である．先に実行する変数 j の計算式や入力処理が移動可能でなければ，その変数 j の値を必要とする変数 i の計算式の移動は不可能である．特に，メモリ必要最適化で計算式を移動する際には，先に計算する計算式の変数に配列型メモリを割り付けることが必要である．高次元の計算式から移動しようとすると，最大メモリ W_{max} を超え，低次元の計算式の移動ができなくなる．

このため，任意のサブツリー T_r の移動は，まず，ルート変数 r が 1 次元の計画対象，次に，2 次元の計画対象の組み合わせ，そして，3 次元の計画対象の組み合わせのように，低次元のルート変数 r からサブツリー T_r の移動が可能であるかの解探索を行う．これにより，解探索の途中で割付可能なメモリがなくなった場合でも実行可能なプログラムを生成することが可能である．

(3) 多重ループの繰返順序

条件評価マトリックス計算では，繰り返しの順序（以下，繰返順序と呼ぶ）によって，メモリ不要最適化とメモリ必要最適化の適用可能変数が異なり，配列型メモリ量 W が変化する．繰返順序の異なるプログラムを図 2.13 に示す．

なお，繰返順序は条件評価マトリックス計算の多重ループの外側から内側に向かう順序で［外側の計画対象＜内側の計画対象］と表記する．条件評価マトリックス（割当コスト）を計算する図 2.13 のプログラム a の繰返順序は［材料＜注文］，プログラム b の繰返順序は［注文＜材料］と示す．

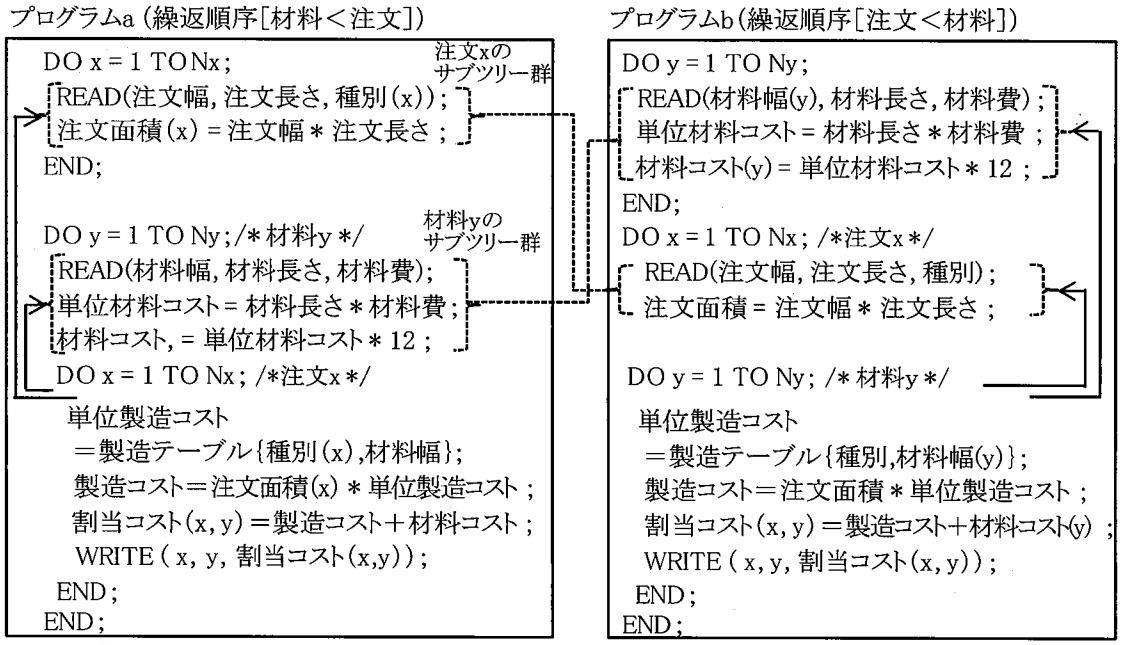

図 2.13　多重ループの繰返順序と配列型メモリの割り付け

プログラム a の繰返順序［材料＜注文］においては，①材料のサブツリー T_r（材料幅と材料コスト）にメモリ不要最適化を適用し，多重ループ内で計算量の最小化を図り，②注文のサブツリー T_r にメモリ必要最適化を適用し，ルート変数 r（注文面積と種別）に配列型メモリ（600 バイト）を割り付け，多重ループの外に独立ループを作り，計算量の最小化が図られている．

一方，プログラム b の繰返順序［注文＜材料］においては，①注文のサブツリー T_r（注文面積と種別）にメモリ不要最適化を適用し，多重ループ内で計算量の最小化を図り，②材料のサブツリー T_r にメモリ必要最適化を適用し，

そのルート変数 r（材料幅と材料コスト）に配列型メモリ（7,000 バイト）を割り付け，多重ループの外に独立ループを作り，計算量の最小化が図られている．

プログラム a の繰返順序［材料＜注文］では，600 バイトの配列型メモリを割り付けるだけで計算量の最小化が可能であるのに対し，プログラム b の繰返順序［注文＜材料］では 7,000 バイトの配列型メモリを割り付ける必要がある．

このように繰返順序の違いによって，最終的に必要となる配列型メモリ量 W が変化する．プリコンパイラの最適化ではメモリ必要最適化で必要となる配列型メモリ量 W が小さい繰返順序を選択して計算式の移動を行う．

(4) 高次元マトリックスの計算式の移動

最大次元が 2 次元の場合，図 2.13 のように繰返順序候補は 2 通りで，移動する計算式は 1 次元の計算式のみの移動となる．メモリ不要最適化による削除可能メモリ量 W_{del} が最大化となる繰返順序によってメモリ必要最適化で追加するメモリ量 W_{add} が最小となる．メモリ不要最適化による多重ループ内での移動（以下，多重ループ内移動と呼ぶ）と，メモリ必要最適化による独立ループへの移動（以下，独立ループ移動と呼ぶ）を行うだけで，計算量 E が最小となる解を最大メモリ量 W_{max} の範囲で探索することが可能である．

これに対し，最大次元が 3 次元以上になると，多重ループ内移動と独立ループ移動に加え，多重ループ内に新たなループを作って計算式を移動する従属ループ移動（図 2.11 の変数 5）が発生する．この従属ループ移動は多重ループの繰返順序の一部を共用することで，計画対象の組み合わせのすべてに配列型メモリを割り付ける必要がなく，少量のメモリ量で計算式を移動することが可能である．任意の繰返順序における計算式の移動方法を表 2.7 に示す．

表 2.7 は，最大次元数 M_{max} の計画問題定義で出現する計画対象の組み合わせの集合 $\boldsymbol{\alpha}$ に対し，多重ループ内移動となる計算式の変数の集合 $\boldsymbol{\beta}$，従属ループ移動となる計算式の変数の集合 $\boldsymbol{\gamma}$，独立ループ移動となる計算式の変数の集合 $\boldsymbol{v}$ を示す表である．表 2.7 の集合 $\boldsymbol{\alpha}$ の要素は，出現する可能性がある計画対象のすべての組み合わせであり，その要素数は 2 次元の場合 3 要素，3 次元の場合 7 要素，4 次元の場合 15 要素となる．これらの要素は，集合 $\boldsymbol{\beta}$，集合 $\boldsymbol{\gamma}$，集合 $\boldsymbol{v}$ の 3 集合に分かれる．計画対象の組み合わせが繰返順序の半順序となる要素（集合 $\boldsymbol{\beta}$ の要素）は，多重ループの中でメモリ不要最適化が可能な要素であり，その要素数は最大次元数 M_{max} と等しくなる．

表 2.7　最大次元の繰返順序と移動方法の関係

計画対象の組み合わせと移動方法 ＼ 最大次元数 M_{max}	2次元 繰返順序 [x<y]	3次元 繰返順序 [x<y<z]	4次元 繰返順序 [x<y<z<w]
移動対象となる計算式の計画対象の組み合わせ集合 $\boldsymbol{\alpha}$	x, y, xy ($_2C_1+_2C_2=3$)	x, y, z, xy, xz, yz, xyz ($_3C_1+_3C_2+_3C_3=7$)	x, y, z, w, xy, xz, xw, yz, yw, zw, xyz, xyw, xzw, yzw, xyzw ($_4C_1+_4C_2+_4C_3+_4C_4=15$)
多重ループ内移動の対象（繰返順序の半順序集合 $\boldsymbol{\beta}$）	x, xy ($M_{max}=2$)	x, xy, xyz ($M_{max}=3$)	x, xy, xyz, xyzw ($M_{max}=4$)
繰返順序の半順序とならない計画対象の組み合わせ集合	y (3-2=1)	y, z, xz, yz (7-3=4)	y, z, w, xz, xw, yz, yw, zw, xyw, xzw, yzw (15-4=11)
従属ループ移動の対象（半順序の部分適用集合 $\boldsymbol{\gamma}$）	(0)	xz (4-3=4)	xz, xw, xyw, xzw, (11-7=4)
独立ループ移動の対象（半順序の部分適用不可集合 $\boldsymbol{\nu}$）	y ($_1C_1=1$)	y, z, yz ($_2C_1+_2C_2=3$)	y, z, w, yz, yw, zw, yzw ($_3C_1+_3C_2+_3C_3=7$)

※ x, y, z：計画対象　　(集合の要素数)

一方，繰返順序の半順序とならない要素（集合 $\boldsymbol{\beta}$ 以外の要素）にはメモリ必要最適化が必要である．その中で，集合 $\boldsymbol{\gamma}$ の要素は，繰返順序の半順序を部分的に利用可能な要素であり，繰返順序の一番目の計画対象(図 2.7 の計画対象 x)を含む要素となる．これに対して，集合 $\boldsymbol{\nu}$ の要素は繰返順序の部分的な利用が不可能な要素であり，繰返順序の一番目の計画対象を含まない集合となり，その要素数は $M_{max}-1$ の集合 $\boldsymbol{\alpha}$ の要素数と等しくなる．

(5) 厳密解と準最適解の探索

3 次元以上の最適化では，2 次元以上の計算式の移動も発生する．2 次元以上の計算式の移動が発生すると，移動した計算式間で多重ループを構成する繰返順序での最適化を行うことが必要となる．

表 2.7 の 3 次元の場合，最適化対象（集合 $\boldsymbol{\alpha}$ の 7 要素）は，集合 $\boldsymbol{\beta}$ の 3 要素，集合 $\boldsymbol{\gamma}$ の 1 要素，集合 $\boldsymbol{\nu}$ の 3 要素に分かれる．集合 $\boldsymbol{\gamma}$ の要素 xz は 2 次元であるが，繰返順序の一番目の計画対象 x を共用することで，従属ループ（例えば，図 2.11 の変数 5 の z ループ）が 1 次元となり，繰返順序は考慮する必要がない．一方，集合 $\boldsymbol{\nu}$ の要素は 2 次元の最適化と同等の要素となる．移動する計算式が 2 次元以上になると，移動計算式間での繰返順序を用いて最適化を行うことによって，厳密解を求めることが可能である．

高次元の最適化では，M_{max} 次元の多重ループの繰返順序，($M_{max}-1$) 次元の独立ループの繰返順序，($M_{max}-2$) 次元の従属ループの繰返順序に対する

最適化を行うことによって，厳密解を求めることが必要である．しかしながら，その探査空間は繰返順序候補数 $M!$の総積となり，2 次元では 2 (2!×1!) 通り，3 次元では 12 (3!×2!×1!) 通り，4 次元では 288 (4!×3!×2!×1!) 通りなる．

厳密解を求めるために再帰的な探索を行っても，独立ループに移動した M_{max}−1 次元以下の変数の中で，計算結果を格納する M_{max}−1 次元の配列型メモリを削減することができない．その最適化によって得られるメモリ量削減の効果は M_{max}−2 次元以下の小さな配列型メモリの削減となる．

そこで，プリコンパイラの最適化では，厳密解の探索よりプリコンパイル時間を優先し，最大次元 M_{max}の繰返順序による解探査のアルゴリズムとする．

2.5.4 メモリ割り付け計算量削減アルゴリズム

メモリ量最小プログラムを初期解（V_{rk}=0，F_{rk}=1）として，最大次元 M_{max}の繰返順序によるサブツリーT_rのルート変数 rごとに配列型メモリ V_{rk}を付け，計算量 Eを削減するアルゴリズムを図 2.14 に示す．

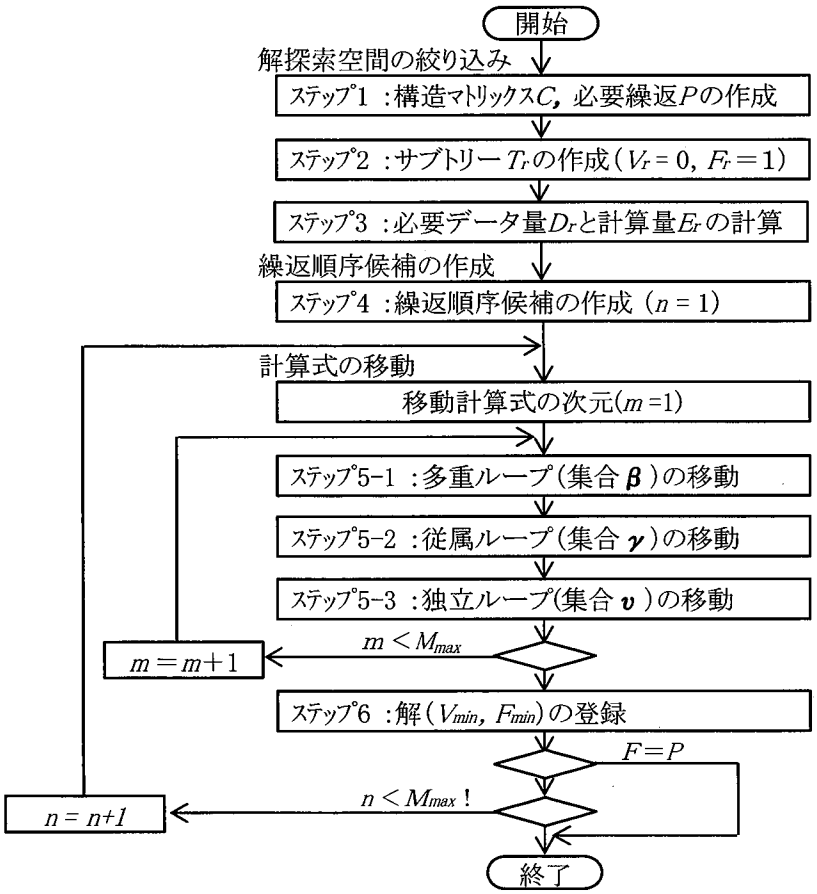

図 2.14　メモリ割付計算量削減アルゴリズム

(1) 解探索空間を絞り込み：ステップ 1～ステップ 3

構造マトリックス C_{ij} と必要繰返 P_{ik} により，サブツリー T_r を作成し，サブツリー T_r ごとの単位計算量 e_r を集計する．なお，初期解のメモリ割付 $V_{rk}=0$，繰返実行 $F_{rk}=1$ とする．

(2) 繰返順序候補の作成：ステップ 4

最大次元 M_{max} の多重ループの繰返順序候補を作成し，不要化が図れる削減可能メモリ量 W_{del} の大きい繰返順序候補から計算式の移動を行う．

(3) 計算式の移動：ステップ 5

計算式の移動は次元の低い変数の計算式から行う．同じ次元の中では，多重ループ（集合 $\boldsymbol{\beta}$），従属ループ（集合 $\boldsymbol{\gamma}$），独立ループ（集合 $\boldsymbol{v}$）の順序で計算式を移動する．さらに，同じ集合の中では，単位計算量 e_r が大きいルート変数の計算式から移動する．

なお，集合 $\boldsymbol{\gamma}$ のルート変数 r は，従属ループとする計画対象 k の $V_{rk}=1$ とし，そのメモリ量を確保できる場合に計算式を移動する．集合 $\boldsymbol{v}$ のルート変数 r は，$V_{rk}=P_{rk}$ となる配列型メモリ量 W が確保可能な場合に計算式を移動し，移動した計算式の繰返実行 $F_{rk}=P_{rk}$ とする．

(4) 解の登録：ステップ 6

単位計算量 e_r と繰返実行 F_{rk} から全体の計算量 E を計算し，得られた計算量 E が最小計算量 E_{min} より小さければ，多重ループの繰返順序，メモリ割付 V_{rk}，繰返実行 F_{rk} を解として登録する．なお，出力変数以外のすべてのルート変数 r の F_{rk} が P_{rk} となった場合には最適化を終了する．F_{rk} が P_{rk} とならない場合には，次の繰返順序候補で計算式の移動を行う．その中で，もっとも小さな計算量 E となるメモリ割付 V_{rk}，繰返実行 F_{rk} の解を探索する．

2.5.5 動作例と評価

2 次元の計画問題定義（図 2.6）と 3 次元の計画問題定義（図 2.8）を用いた提案アルゴリズムの動作例を表 2.8 と表 2.9 に示す．動作条件として，入出力変数のメモリ量 d_i は変数名和英対応表（表 2.2）の領域（データ長）を用い，中間変数のメモリ量 d_i は 4 バイト，最大メモリ量 W_{max} を 1M バイトとする．

計算量 e_iは 1 計算式の計算量を 5，1 データ入力の計算量を 10 とする．出力変数 iの配列型メモリ量 W_iや計算量 E_iは変数と比べてはるかに大きくなるため，計算結果は 2 次記憶に出力し，出力変数に配列型メモリを割り付けない．また，出力変数の計算量 e_iは全ケースで同じあり，計算量の削減効果は出力変数の計算量を除外した計算量 E' と最終の割付メモリ量 Wを比較する．

(1) 2 次元の条件評価マトリックスの動作例と評価

表 2.8 の 2 次元（注文 N_x : 100 件，材料 N_y : 1,000 件）の動作には 2 通りの繰返順序候補があり，繰返順序候補［y<x］を選択することで 7,000 バイトが削除可能となる．これにより，追加メモリ量 W_{add}が 600 バイトで計算量 E' が初期解の 7,500,000 から最小値 43,500 に削減される．

表 2.8　2 次元の計画問題定義の動作例

ステップ4

候補番号 n	繰返順序候補	ルート変数r 集合β	集合γ	集合υ	削除可能メモリ量 W_{del}	追加メモリ量 W_{add}
1	[y<x]	2, 5	ϕ	3, 4	7,000	600
2	[x<y]	3, 4	ϕ	2, 5	600	7,000

ステップ5(n:1)

候補番号 n	移動次元 M	移動方法	移動サブツリー T_r	単位計算量 e_r	メモリ割付 V_{rk}	ルート変数メモリ量 W_r	割付メモリ量 W	繰返実行 F_{rk}	繰返計算量 E'
1	1	β	2	30	(0, 0)	0	0	(0, 1)	4,530,000
			5	10	(0, 0)	0	0	(0, 1)	3,540,000
		υ	3	25	(1, 0)	400	400	(1, 0)	1,042,500
			4	10	(1, 0)	200	600	(1, 0)	43,500

※計画対象 x:注文, y:材料, ルート変数r 1:割当コスト, 2:材料コスト, 3:注文面積, 4:種別, 5:材料幅
単位計算量e_r:移動サブツリーT_rに含まれる計算量e_iの総和, W_r :ルート変数rの配列型メモリ量

(2) 3 次元の条件評価マトリックスの動作例と評価

表 2.9 の 3 次元（注文 N_x : 1,000 件，材料 N_y : 100 件，装置 N_z : 10 件）の動作には 6 通りの繰返順序候補がある．繰返順序候補中で，削除可能メモリ量 W_{del}の最大値は繰返順序候補の（451,960 バイト）である．

繰返順序候補に対するメモリ量 W，繰返計算量 E' の関係を表 2.10 に示す．最小の計算量 E' を得るまでの計算式移動に必要となる配列型メモリのメモリ量 Wは，繰返順序候補 1 で 5K バイト，繰返順序候補 6 で 417K バイトである．最小の計算量 E' を得るためには，変数 4，5，6 のいずれかの変数に 2 次元の配列型メモリを割り付ける必要がある．

表 2.9　3 次元の計画問題定義の動作例

ステップ4

候補番号 n	繰返順序候補	ルート変数r 集合 β	集合 γ	集合 υ	削除可能メモリ量 W_{del}	追加メモリ量 W_{add}
1	[x<y<z]	1, 13, 14, 4	5	2, 23, 24, 3, 33, 34, 6	451,960	5,360
2	[x<z<y]	1, 13, 14, 5	4	2, 23, 24, 3, 33, 34, 6	451,600	5,720
3	[y<x<z]	2, 23, 24, 4	6	1, 13, 14, 3, 33, 34, 5	405,160	52,160
4	[z<x<y]	3, 33, 34, 5	6	1, 13, 14, 2, 23, 24, 4	43,720	413,600
5	[y<z<x]	2, 23, 24, 6	4	1, 13, 14, 3, 33, 34, 5	401,200	56,120
6	[z<y<x]	3, 33, 34, 6	5	1, 13, 14, 2, 23, 24, 4	40,120	417,200

ステップ5(n:1)

候補番号 n	移動次元 M	移動方法	移動サブツリー T_r	単位計算量 e_r	メモリ割付 V_{rk}	ルート変数メモリ量 W_r	割付メモリ量 W	繰返実行 F_{rk}	繰返計算量 E'
1	1	β	1	25	(0, 0, 0)	0	0	(1, 0, 0)	125,025,000
			13	10	(0, 0, 0)	0	0	(1, 0, 0)	115,035,000
			14	10	(0, 0, 0)	0	0	(1, 0, 0)	105,045,000
		υ	3	25	(0, 0, 1)	40	40	(0, 0, 1)	80,045,250
			33	10	(0, 0, 1)	40	80	(0, 0, 1)	70,045,350
			34	10	(0, 0, 1)	40	120	(0, 0, 1)	60,045,450
			2	25	(0, 1, 0)	400	520	(0, 1, 0)	35,047,950
			23	10	(0, 1, 0)	400	920	(0, 1, 0)	25,048,950
			24	10	(0, 1, 0)	400	1320	(0, 1, 0)	15,049,950
	2	β	4	5	(0, 0, 0)	0	1320	(1, 1, 0)	10,549,950
		γ	5	5	(0, 0, 1)	40	1360	(1, 0, 1)	5,599,950
		υ	6	5	(0, 1, 1)	4000	5360	(0, 1, 1)	604,950

※計画対象 x:注文, y:材料, z:装置, ルート変数r:出力変数, 中間変数1～6, 入力変数13, 14, 23, 24, 33, 34
単位計算量e_r:移動サブツリーT_rに含まれる計算量e_iの総和, W_r:ルート変数rの配列型メモリ量

繰返順序候補 1 の変数 6 に 2 次元の配列型メモリを割り付けることで 412K バイトのメモリが削減可能である．この削減効果は与えられるデータ件数によっても異なる．材料と装置の件数が 5 倍に増えた場合の削減効果は 25 倍（10M バイト）となる．最大メモリ量 W_{max}（1M バイト）を超え，計算式移動を中断する場合でも，削除可能メモリ量 W_{dell}が最大となる繰返順序候補によって最小の計算量 E' が得られる可能性が高い．このため，提案アルゴリズムでは，削除可能メモリ量 W_{dell}が最大となる繰返順序候補から計算式移動を行い，最大メモリ量 W_{max}を超えない範囲で，計算量 E' が最小となる解（プログラム）を導出している．この最適化の動作結果（3 次元）により得られる PL/I 言語に基づく生成コードの例を図 2.15 に示す．

図 2.15 の独立ループ移動の変数は 11 変数（変数 2, 3, 6, 21～24, 31～34）である．このプログラムを，図 2.11 の最適なプログラムと比べると，変数 2, 24 の計算結果を格納するための配列型メモリ（800 バイト）が割り付けられたままであり，メモリ量最小ではない．本来，変数 2，24 の計算結果を多重ループへ

受け渡す必要はなく，変数2，24の値を使った変数6の計算結果を配列型メモリ（2次元の配列型メモリ）に格納するだけでよい．

この厳密解を得るには，独立ループに移動した2次元以下の計算式の最適化が必要である．しかしながら，その削減効果は1次元以下の配列型メモリ（800バイト）の削減となる．その削減効果は最大次元 M_{max} の繰返順序による2次元の配列型メモリの削減効果（変数4：400Kバイトと変数5：40Kバイト）からすると大きな値ではない．提案アルゴリズムでは，厳密な最適化を行わず，コンパイル時間を優先している．

表 2.10　繰返順序候補のメモリ量と計算量の関係

候補1 [x＜y＜z]			候補2 [x＜z＜y]			候補3 [y＜x＜z]		
移動変数	メモリ量 W	繰返計算量 E'	移動変数	メモリ量 W	繰返計算量 E'	移動変数	メモリ量 W	繰返計算量 E'
ϕ	0	150,000,000	ϕ	0	150,000,000	ϕ	0	150,000,000
1	0	125,025,000	1	0	125,025,000	2	0	125,002,500
13	0	115,035,000	13	0	115,035,000	23	0	115,003,500
14	0	105,045,000	14	0	105,045,000	24	0	105,004,500
3	40	80,045,250	3	40	80,045,250	3	40	80,004,750
33	80	70,045,350	33	80	70,045,350	33	80	70,004,850
34	120	60,045,450	34	120	60,045,450	34	120	60,004,950
2	520	35,047,950	2	520	35,047,950	1	4,120	35,029,950
23	920	25,048,950	23	920	25,048,950	13	8,120	25,039,950
24	1,320	15,049,950	24	1,320	15,049,950	14	12,120	15,049,950
4	1,320	10,099,950	5	1,320	10,099,950	4	12,120	10,549,950
5	1,360	5,599,950	4	1,720	5,599,950	6	12,160	5,554,950
6	5,360	604,950	6	5,720	604,950	5	52,160	604,950

候補4 [z＜x＜y]			候補5 [y＜z＜x]			候補6:[z＜y＜x]		
移動変数	メモリ量 W	繰返計算量 E'	移動変数	メモリ量 W	繰返計算量 E'	移動変数	メモリ量 W	繰返計算量 E'
ϕ	0	150,000,000	ϕ	0	150,000,000	ϕ	0	150,000,000
3	0	125,000,250	2	0	125,002,500	3	0	125,000,250
33	0	115,000,350	23	0	115,003,500	33	0	115,000,350
34	0	105,000,450	24	0	105,004,500	34	0	105,000,450
2	400	80,002,950	3	40	80,004,750	2	400	80,002,950
23	800	70,003,950	33	80	70,004,850	23	800	70,003,950
24	1,200	60,004,950	34	120	60,004,950	24	1,200	60,004,950
1	5,200	35,029,950	1	4,120	35,029,950	1	5,200	35,029,950
13	9,200	25,039,950	13	8,120	25,039,950	13	9,200	25,039,950
14	13,200	15,049,950	14	12,120	15,049,950	14	13,200	15,049,950
5	13,200	10,099,950	6	12,120	10,054,950	6	13,200	10,054,950
6	13,600	5,104,950	4	16,120	5,554,950	5	17,200	5,104,950
4	413,600	604,950	5	56,120	604,950	4	417,200	604,950

```
APPL:PROC OPTIONS(MAIN);
 DCL（入力変数11, 入力変数12, 入力変数13, 入力変数14)…;
 DCL（入力変数21, 入力変数22, 入力変数23(100), 入力変数24(100)) …;
 DCL（入力変数31, 入力変数32, 入力変数33(10), 入力変数34(10)) …;
 DCL（変数1, 変数2(100), 変数3(10), 変数4, 変数5(10), 変数6(100, 10), 出力変数) …;
 DO z = 1 TO Nz;
   READ（入力変数31, 入力変数32, 入力変数33(z), 入力変数34(z)) ;
   変数3(z) = 入力変数31＋入力変数32;
 END;
 DO y = 1 TO Ny;
   READ（入力変数21, 入力変数22, 入力変数23(y), 入力変数24(y)) ;
   変数2(y) = 入力変数21+入力変数22;
 END;
 DO y = 1 TO Ny;
   DO z = 1 TO Nz;
     変数6(y, z) = 変数2(y)＋変数3(z)＋入力変数24(y)＋入力変数34(z);
   END;
 END;
 DO x = 1 TO Nx;
   READ（入力変数11, 入力変数12, 入力変数13, 入力変数14) ;
   変数1 = 入力変数11＋入力変数12;
   DO z = 1 TO Nz
     変数5(z) = 変数1＋変数3(z)＋入力変数14＋入力変数33(z);
   END;
   DO y = 1 TO Ny;
     変数4 = 変数1 ＋入力変数13＋入力変数23(y);
     DO z = 1 TO Nz;
       出力変数 = 変数1＋変数3(z)＋変数4＋変数5(z) ＋変数6(y, z);
       WRITE(x, y, z, 出力変数) ;
     END;
   END;
 END;
END APPL ;
```

図 2.15　生成コード（手続き型プログラム）の例

2.6 プリコンパイラの全体構成

提案方式を実装したプリコンパイラの全体構成を図 2.16 に示す．以下，プリコンパイラの字句解析部，構文解析部，最適化部，コード生成の処理内容を説明する．なお，計画問題定義プログラムの変更は定数の変更が多く，定数分離機能を実現するため，変更前にコード生成した計画問題定義プログラムをマスタとして，計画問題定義マスタ格納部に格納する．

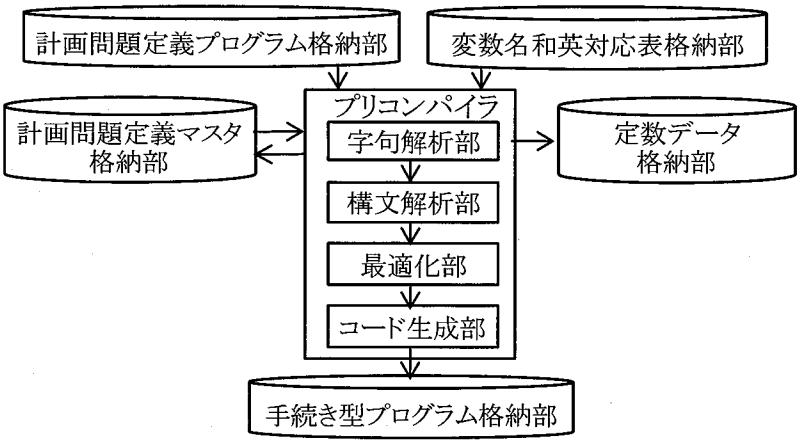

図 2.16　プリコンパイラの全体構成

(1) 字句解析部

字句解析部は,計画問題定義プログラム格納部のソースプログラムを入力し,字句解析を行い計画問題定義プログラムに含まれる定数を定数データ格納部に出力する．さらに，入力したソースプログラムとマスタの内容を比較し，定数以外が一致していれば，定数変更としてプリコンパイラの処理を終了する．

(2) 構文解析部

構文解析部は変数名和英対応表を用いて入力したソースプログラムの構文チェックを行う．正しい構文であれば，使用されている計画対象，入力変数，中間変数，出力変数，計算式等の登録を行う．

(3) 最適化部

前節で説明したアルゴリズムに従い，繰返順序を決定し，変数にメモリを割り付け，計算式を移動し，繰返し実行される計算の計算量を最小化する．

(4) コード生成部

コード生成部は，出力する手続き型プログラムの言語仕様に合わせ，計画問題定義の入力変数，中間変数，出力変数等のデータ定義文，対象データの入力文，パラメータ化した定数の入力文，繰り返し計算の指定（ループ文），計算式，条件評価マトリックスの出力文を作成し，生成コードを出力する．

なお,コード生成後に入力したソースプログラムを計画問題定義マスタ格納部に格納し，次のプリコンパイル時のマスタとして使用する．

2.7 提案方式の評価

計画問題定義言語とプリコンパイラ方式の適用効果を確認するため，計画問題定義言語の実業務への適用と，プリコンパイラのオブジェクト性能とコンパイル性能の評価を行った．評価結果を以下に示す．

2.7.1 計画問題記述言語の評価

鉄鋼の連続加工処理設備に対する製造順序計画等の２つの計画業務[80]に，提案言語を提供し，実業務での計画問題定義の変更容易性を評価した．その結果を以下に示す．

(1) 理解容易性の向上

提案言語は，計算機の処理手順を意識せず，専門用語（日本語変数名）による業務知識（計画問題定義）の記述が可能である．生産計画支援システムの説明会において計画問題定義の記述内容（製造順序の決定要因となる段取ロス等）の細部に渡る意見や質問が管理者クラスの専門家や幹部から寄せられた．従来のプログラムリストによる説明では，計算機の処理手順の説明に終始し，このような業務知識に関する質疑は起こり得ない現象であった．提案言語による記述内容は，はじめて見る幹部や専門家でも，理解することが早くかつ容易であった．

(2) 変更容易性の実現

提案言語による記述内容は，計画立案を行っている計画者が理解することも容易であった．説明会以降，計画者による計画問題定義プログラムの開発と変更が行われた．計画問題定義プログラムの条件評価式や業務テーブルの変更をシステム開発者に依頼することが不要となった．計画者による計画問題定義プログラムの直接変更により，頻繁に起こる製品や生産工程の変化に対して，生産計画支援システムに計画問題定義の変更を迅速に取り込むことが可能となった．

(3) 変更工数の削減効果

これらの理解容易性と変更容易性によって，計画者による変更仕様の抽出と変更仕様書の作成，システム開発者による手続き型プログラムの変更とテストを実施するという従来の変更作業の工数を大幅に削減することが可能となった．

特に，提案言語による理解容易な記述内容を変更仕様書とすることが可能となり，計画者による変更仕様書の作成が不要となった．また，変更した計画問題定義プログラムから手続き型プログラムを自動生成することにより，システム開発者による手続き型プログラムの変更作業も不要となった．しかも，計画立案処理過程の条件評価マトリックスを出力変数に指定することで，条件評価の計算結果の確認が可能となり，計画者による変更テスト（計算結果の確認）を非常に早く実施することが可能となった．これにより，本事例では，従来の変更作業と比べ，変更工数を 1/4～1/3 に削減可能であった．

2.7.2 プリコンパイル方式の評価

プリコンパイラの適用可能性を確認するために，生成した手続き型プログラムのオブジェクト性能（条件評価マトリックスの計算処理時間）と，計画問題定義プログラムから手続き型プログラムへのコンパイル性能（コンパイル時間とコード変換率）を評価する．なお，使用計算機は 1974 年に発表された(株)日立製作所製のメインフレームコンピュータ M-180（最大主記憶容量 8MB，平均命令実行時間 310ns）である．

(1) オブジェクト性能の評価

オブジェクト性能として，システム開発者（プログラマ）が作成する手続き型プログラムとプリコンパイラが生成する手続き型プログラム（生成コード）の処理性能を比較評価する．注文を材料に割り付ける計画問題の計画問題定義プログラム（入力変数 19，中間変数 8，出力変数 1，業務テーブル 1，計算式 9 式）を用い，3 名のプログラマが開発したプログラムの処理時間を比較した．評価結果を図 2.17 に示す．プログラム 1（プログラミングを始めて 2 年目のプログラマのプログラム）をのぞいて，熟練したプログラマによるプログラム 2，3 および生成コードの処理時間はほぼ同等である．プログラム 1 にはデータの入出力や計算の中間結果を記憶しておく配列型メモリがなく，入力データをファイルから 1 件ずつ読み込むメモリ量最小プログラミングである．配列型メモリを考慮しなくても大量のデータ処理が可能であるが，手続き型プログラムのコンパイラではメモリ割り付けによる繰り返し計算の最適化がなされない．このため，プログラム 1 は，プログラム 2，3 や生成コードより，入力データの処理で不要な繰り返し処理が実行され，処理時間が 5.5 倍となった．

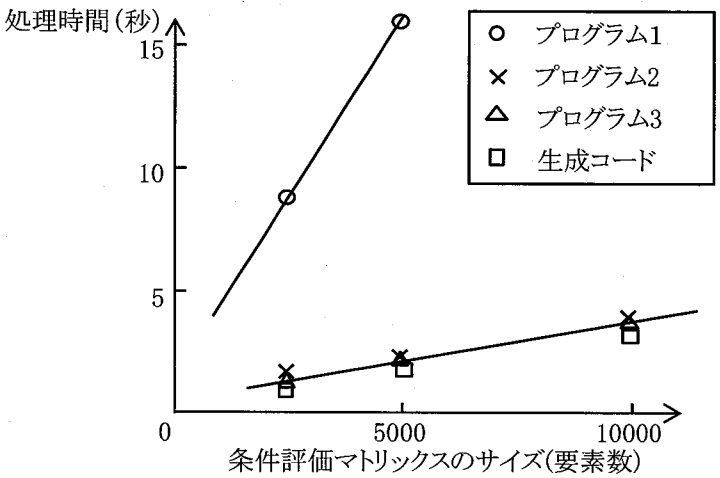

図 2.17　プリコンパイラのオブジェクト性能

プログラム 2，3 と生成コードで処理時間を比較すると，プログラム 2 では，繰返計算の中からの条件検索式の移動が考慮されていないため，計算量が大きくなり，生成コードより処理時間がかかる．また，プログラム 3 も利用可能な最大量のメモリを割り付けており，処理時間がかかっている．生成コードには，プログラマによって異なる計算手続きの記述（未移動の計算式や余分な計算手続きの記述）がなく，プログラマが作成した手続き型プログラム以上の処理性能を確保することが可能である．

(2) プリコンパイル時間の評価

計画問題定義プログラム(BLOCK 文, FOR 文, ENVIRONMENT 文, LOGIC 文を含むソースコード）のステートメント数とプリコンパイル時間（論理変更時と定数変更時のコード生成時間）を図 2.18 に示す．

評価した 5 つの計画問題定義プログラム（100 行～400 行）で，論理変更時（計算式，変数名，定数等の変更時）のプリコンパイル時間は 2～5 秒，定数変更時（定数の変更時）のプリコンパイル時間は 0.5～1.5 秒である．特に，定数変更時は手続き型プログラムのコンパイル，リンクが不要であり，プリコンパイル処理終了後，すぐに条件評価マトリックスの計算処理を実行することが可能である．プリコンパイル時間は変数や定数の数によっても異なるが，プリコンパイル処理は数秒程度で完了し，システム開発者に依頼するよりも短時間である．計画者による計画問題定義プログラムの計算式や定数の迅速な変更が可能であることを確認した．

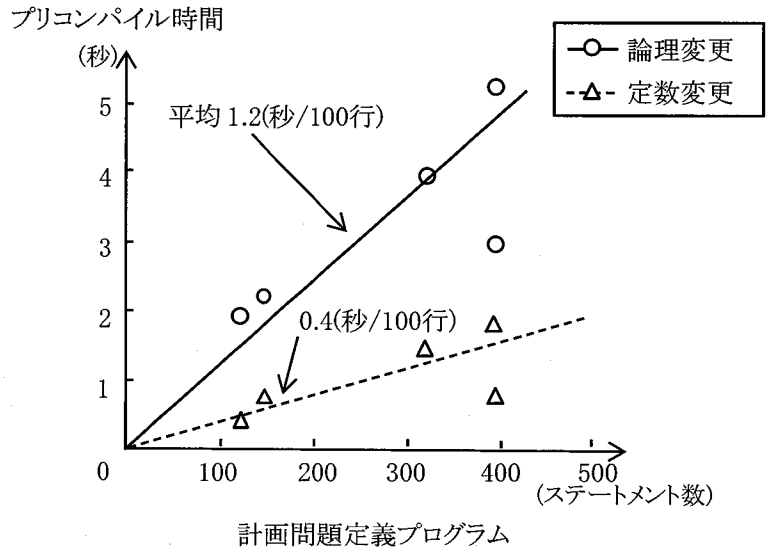

図 2.18　プリコンパイル時間

(3) コード生成量

計画問題定義プログラムの計算式数とプリコンパイラの生成コードのステートメント数の関係を図 2.19 に示す. 生成コードのプログラムステートメント数は変数と定数の数にも依存するが，適用した 5 つの計画問題定義プログラム（8 計算式～54 計算式）では，1 計算式当たりの平均生成コードのステートメント数 24 であった.

生成コードのプログラムステートメント（計算手続き）には，入出力変数，中間変数，定数パラメータの宣言文，入出力データの入出力文，定数の入力文，計算式，繰返計算範囲の指定文等の計算手続きが含まれる．1 つの計算式が 6 つの変数，5 つの定数から構成されると，プリコンパイラは，計算に必要なメモリを確保するための宣言文（11 ステートメント）と，それらのデータ入力文（10 ステートメント），計算式，繰返計算範囲の指定文等を生成する．これにより，1 つの計算式から 20 倍を越えるプログラムステートメントが生成される.

生成コード（手続き型プログラム）では，数多くの計算手続きの中に条件評価マトリックスの計算式が埋もれるのに対して，提案言語による記述には計算手続きの記述が不要であり，計画者は計算式や定数を変更するだけで計画問題定義を変更することが可能である.

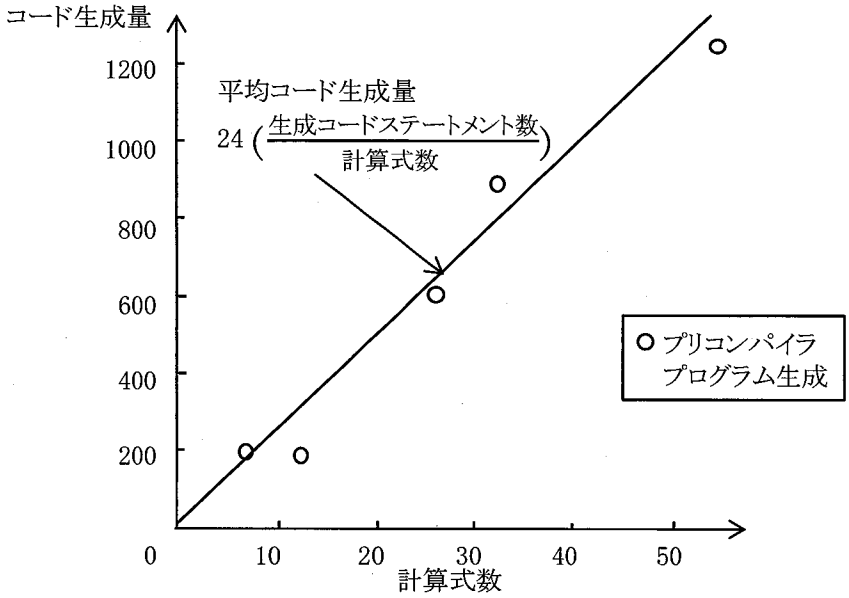

図 2.19　コード生成量

2.8 結言

本章では，対象固有の計画問題定義の内容（業務テーブルや条件評価式）を説明し，情報処理の専門家でない計画者でも変更容易な計画問題定義プログラムの記述が可能な計画問題定義言語と，計画問題特有の多重な繰返計算の計算量を削減するプリコンパイル方式を提案した．

提案言語による記述内容は，計画対象の制約条件や評価指標の変化を把握している計画者が直接変更することが可能であり，計画者による計画問題定義の変更容易性を確認した．また，計画問題定義プログラムから計算機プログラムを自動生成することにより，システム開発者の手を介さず，計画問題定義の変更が可能となった．特に，提案言語は記述者以外にも理解容易であり，変更仕様書の作成や手続き型プログラムの開発が不要となり，変更工数が削減できることを確認した．

本方式のプリコンパイラが自動生成した手続き型プログラムは，熟練したプログラマが開発したプログラムと同等以上の計算処理性能を有することを確認した．これにより，計画問題定義プログラムの変更において，プログラムの計算処理性能を考慮する必要がなく，情報処理の専門家でない計画者でも容易に計画問題定義の変更が可能であることを確認した．

しかも，変更の母体となる計画問題定義プログラムの記述内容として，変数

へのメモリ割付，計算式の計算順序，組み合わせ計算のための繰返順序等の計算手続きの記述が不要であり，そのコード量（ステートメント数）が少ない．手続き型プログラムを変更するよりも提案言語の記述内容を変更することが容易であることも確認した．

計画問題定義言語とプリコンパイル方式を採用した知識型計画支援システムHPGS（Hitachi Flexible＆Intelligent Planning Support System）が，汎用の生産計画支援システムとして製品化[81, 82]され．鉄鋼，自動車等の生産計画支援システムに適用された．従来，生産計画支援システムは，計画者が仕様を検討し，仕様書を作成して，システム開発者が計画問題定義の手続き型プログラムを開発してきたが，計画者による計画問題定義プログラムの開発が可能となり，生産計画支援システムの開発・保守の容易化という効果を上げた[83]．

本章で述べた計画問題定義言語とプリコンパイル方式は．計画問題定義プログラムで定義された条件評価マトリックスの計算処理を計算機で処理するための方法である．生産計画支援システムには，条件評価マトリックスの値から計画を立案する計画立案プログラムも必要である．計画立案プログラムとして線形計画法の求解プログラムや個別開発したプログラムを固定的に使用可能な生産計画支援システムでは，計画問題定義の変更容易性の確保に本章で提案した計画問題定義言語とプリコンパイル方式を適用可能である．

しかしながら，制約条件や評価指標が頻繁に変化する生産計画では，求解プログラムや個別開発プログラムを固定的に使用することが困難となることが多い．柔軟な生産計画支援システムの実現には，計画問題定義の変更容易性を確保することだけでなく，計画立案プログラムの変更容易性を確保することも課題である．そこで，次章では，制約条件や評価指標が頻繁に変化するスケジューリング問題を対象として，計画立案プログラムの変更容易性を実現するプログラム構成方式とその記述形式を提案する．

第3章

知識型スケジューリング用計画立案プログラム構成方式と記述形式

3.1 緒言

本章では，計画問題定義の変化とともに計画立案手順の変更が必要となる生産スケジューリング問題に対し，計画立案手順の変更容易性を確保する知識型スケジューリング用計画立案プログラムの構成方式と記述形式を提案する．

生産スケジューリング問題は，生産ロットの製造を実施する装置と装置上での製造順序を立案し，生産ロットの製造時間を装置の時間軸上に割り付けていく問題である．生産ロットの数が実用規模の数十～数百となると，生産ロットと装置の組み合わせやその製造順序の候補が膨大となり，すべての候補を評価する汎用的な列挙法では，実用的な時間で最適解を得ることが不可能な問題である．このため，特定の生産形態や評価指標に適した割付方法として，数多くのディスパッチングルールが提案されている[84-90]．これらの割付方法は，最適解を得る方法ではないが，計画対象の評価指標に対して適切な割付方法を適用すれば，実用的な時間で計画立案を処理することが可能である．

しかしながら，その計画立案プログラムには，生産形態や評価指標に適した割付方法のプログラムや，適切な割付方法を選択するための割付状態の特徴を抽出するプログラムが数多く必要となる．評価指標に適した割付方法，割付状態を認識する特徴抽出，適切な割付方法の選択は，計画者の計画立案手順の知識であり，計画者によるプログラムの変更容易性が必要となる．そこで，本章では計画立案プログラムの変更容易な構成と記述形式を検討する．

以下，第 3.2 節では本章で扱う生産スケジューリング問題の特徴と計画立案方法を示す．第 3.3 節では変更容易性を確保する計画立案プログラムの構成方式と記述形式，スケジューリング問題用のプログラムライブラリを提案し，提案方式を実装した知識型スケジューリングシステムの構成を示す．第 3.4 節では食品加工の生産スケジューリングへ適用し，提案方式の変更容易性を示す．

3.2 生産スケジューリングの概要

本章では，対象とする計画問題の特徴，計画立案方法を説明する．

3.2.1 生産スケジューリング問題の特徴

(1) 生産形態と計画問題の構造

生産スケジューリング問題は，生産ロットと装置の組み合わせの制約条件や生産効率の評価指標に従い，生産ロットの製造時間や段取時間を装置の時間軸上に割り付ける生産計画問題である．生産ロットの生産工程や製造設備の装置構成によって，フローショップとジョブショップの生産形態があり，探索空間の大きさが異なる．生産工程と装置構成の例を図 3.1，計画結果の例を図 3.2 に示す．

フローショップ（図 3.1（a））は「すべての生産ロットの製造装置を同一」とする流れ作業の生産形態のことである．装置間で生産ロットの入れ替えがなく，各装置での生産ロットの製造順序は変化しない．

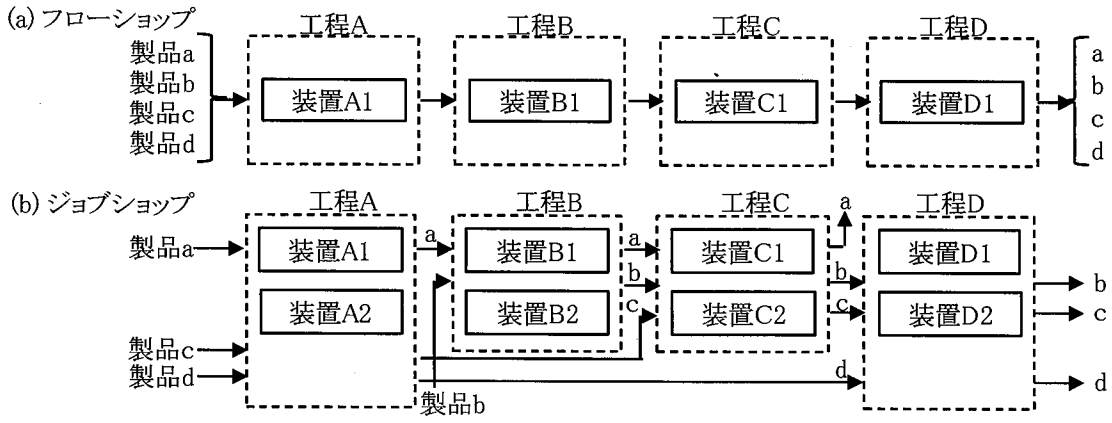

図 3.1　フローショップとジョブショップの生産形態

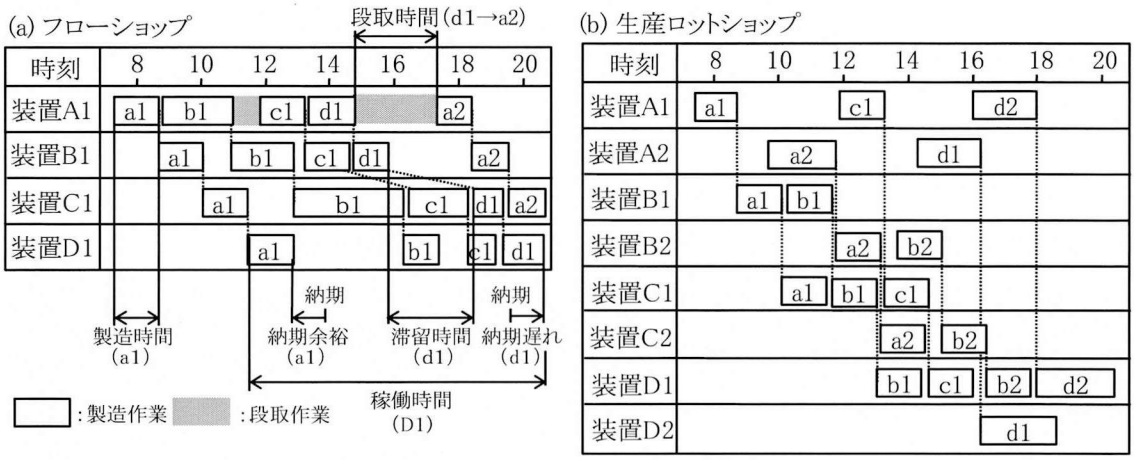

図 3.2　生産スケジュールの例

このフローショップ問題は１つの製造順序（投入順序）を決定する問題である．生産スケジュール（図 3.2（a））はどの装置上でも製造順序が同一の順序（生産ロット a1, b1, c1, d1, a2 の順序）となる．このスケジュールに，長い製造時間の作業（装置 C1 の b1 や c1）が存在すると，その後の作業（装置 C1 の c1 や d1）の開始時刻が遅れ，装置間での滞留時間（d1 の装置 B1 の終了時刻と装置 C1 の開始時刻の差）が長くなる．その結果，最終工程の終了時刻が遅くなり，納期の厳しい生産ロットの場合は納期遅れが発生する．

一方，ジョブショップ（図 3.1（b））は，製造装置を同一とする制約がない生産形態のことで，生産ロットを製造する装置を選択することが可能である．この計画立案では，空き時間のある製造可能な装置を検索し，滞留時間や納期遅れ等が小さくなる装置に製造時間を割り付けることが可能である．この問題は生産ロット・装置の組み合わせと装置ごとの製造順序を決定する必要がある問題である．

これらの生産形態は，製品の生産工程や装置構成に依存するだけでなく，生産量の増減によっても変化する．フローショップの生産形態であっても生産量が増加し，ボトルネックとなる生産工程があれば装置や作業員を増やし，生産形態をジョブショップに変更する．逆に，生産量が減少すればジョブショップからフローショップに変更する．生産スケジューリングは対象とする生産形態が変化し，決定する計画内容（組み合わせと順序）さえも変化する問題である．

(2) 探索空間の大きさ

１つの製造順序を決定するフローショップと装置ごとの製造順序を決定する

ジョブショップでは候補解の数（以下，探索空間の大きさと呼ぶ）が異なる．N 件の生産ロットのフローショップスケジューリングの探索空間の大きさは N！である．これに対し，N 件の生産ロットを M 台の装置に割り付けるジョブショップスケジューリングの探索空間の大きさは$(N!)^M$となる．これらの生産形態の違いによる探査空間の大きさ（解候補の数）を表 3.1 に示す．

生産ロットが数件，装置が数台の小規模な問題の探索空間は10^7～10^{10}程度であり，すべての候補解の条件評価を行う汎用的な列挙法で最適解を得ることが可能である．しかしながら，生産ロットが数十件を超える実用規模の問題では探索空間の大きさが10^{64}以上となり，どんな高性能な計算機を使用しても列挙法による探索処理では多大に時間がかかり，実用的な探索時間とはならない．

表 3.1　生産スケジューリング問題の探索空間（列挙法）

生産ロット数 N	フローショップ (N!)	生産ロットショップ $(N!)^M$				
		装置数 M＝2	装置数 M＝5	装置数 M＝10	装置数 M＝20	装置数 M＝50
1	1	1	1	1	1	1
2	2	4	32	1,024	10^{6}	10^{15}
5	120	14,400	10^{10}	10^{21}	10^{42}	10^{104}
10	10^{7}	10^{13}	10^{33}	10^{66}	10^{131}	10^{328}
50	10^{64}	10^{129}	10^{322}	10^{645}	10^{1290}	10^{3224}
100	10^{158}	10^{316}	10^{790}	10^{1580}	10^{3159}	10^{7899}
500	10^{1134}	10^{2268}	10^{5670}	10^{11341}	10^{22682}	10^{56704}

(3) 生産スケジューリングの制約条件と評価指標

生産スケジューリング問題は，このような膨大な探索空間を有するにもかかわらず，単に生産ロットの製造時間を確保すればよいという問題ではない．製品特性や生産量に応じた多種多様の制約条件と評価指標を考慮することが要求される問題である．生産スケジューリングの制約条件や評価指標に用いられる条件評価の変数と評価尺度の例を表 3.2 に示す．

これらの条件評価の変数や評価尺度は，対象とする製品の特性や与えられる生産量によって異なる．例えば，後述する食品加工では滞留時間が長くなると製品品質（味）に影響を及ぼすため，最大滞留時間が重視される．一方，製品が小さな電子部品では，滞留より納期が重視される．また，納期遅れが多発する

生産量では平均納期遅れが重視されるが，少ない生産量では稼働時間や空き時間の平準化，偏差を小さくすることが重視される．しかも，すべての生産ロットの作業を装置に割り付けなければ，それらの評価尺度（平均納期遅れ，最大滞留時間，装置稼働時間偏差等）の値を計算することができない．

表 3.2　制約条件と評価指標の変数と評価尺度

条件評価の変数	割付内容の評価尺度
製造時間	製造時間(全体, 生産ロット別, 装置別)の最大, 最小, 平均, 合計
段取時間	段取時間(全体, 生産ロット別, 装置別)の最大, 最小, 平均, 合計
稼働時間	稼働時間(全体, 装置別)の最大, 最小, 平均, 合計. 偏差
空き時間	空き時間(全体, 装置別)の最大, 最小, 平均, 合計. 偏差
滞留時間	滞留時間(全体, 生産ロット別, 装置別)の最大, 最小, 平均, 合計
着手可能時刻	開始時刻(全体, 生産ロット別, 装置別)の最大, 最小, 平均, 合計
納期	納期遅れと納期余裕(全体, 生産ロット別)の最大, 最小, 平均, 合計

3.2.2 生産スケジュールの立案方法

すべての生産形態や評価尺度に対して大規模な計画問題の最適解を実用的な処理時間で得る立案方法はないが，特定の生産形態や評価尺度に対する立案方法は数多く提案されている．

(1) Johnson の方法

Johnson の方法は，2 台の装置と N 件の生産ロットのフローショップ問題で総稼働時間を最小とする方法である[84]．この方法は，最も短い製造時間を有する生産ロットを割付対象として選び出し，その製造時間が第 1 工程の製造時間であれば製造順序の前方に，第 2 工程であれば後方に割り付ける方法である．この方法は，2 工程の製造時間（2N の製造時間）の中から最小製造時間を探索して未割付の生産ロットがなくなるまで繰り返す方法であり，その探索回数は 2N(N+1)である．生産ロット数 N が 500 件であれば，探索回数は約 10^6 であり，列挙法の探索空間（表 3.1）の 10^{1134} と比べてはるかに小さな数である．2 章の計画問題定義の条件評価マトリックス（N×2 の製造時間）が計算されていれば，生産ロット数 N の 2 乗オーダーの計算量で立案可能であり，計算機を使用すれば短時間に計算機処理することが可能である．

しかしながら，Johnson の方法は 2 台の装置の割付方法であり，装置が 3 台

となる問題には適用できない．しかも製造時間以外の条件評価の変数（表 3.2）が考慮されておらず，例えば，短納期の生産ロットが増えると納期遅れとなる生産ロットが多発することとなる．

(2) ディスパッチングルール

Johnsonの方法のように装置数や評価指標が変わるだけ適用不可となる方法は計画問題定義が変化する計画問題に適さない．一方，計画者は，制約条件や評価指標が変化する計画問題であっても，過去の経験から割付方法を切り替えて計画を立案する．この切り替えて使用する割付方法（ディスパッチングルール）は，着目する変数（以下，着目変数と呼ぶ）が異なり，影響を及ぼす評価指標が変化する．製造時間と納期に関する割付方法[87]を表 3.3 に示す．

表 3.3　製造時間と納期に関する割付方法（ディスパッチングルール）

割付方法	着目変数	割付手順
最小製造時間優先	製造時間	製造時間が最も短い生産ロットを割付対象として前方から割り付ける.
最大製造時間優先	製造時間	製造時間が最も長い生産ロットを割付対象として前方から割り付ける.
最早納期優先	納期	納期が最も短い生産ロットを割付対象として前方から割り付ける.
納期余裕優先	納期 製造時間	納期余裕が最も短い生産ロットを割付対象として前方から割り付ける. （納期余裕＝納期－製造時間）

最小製造時間優先割付は製造時間が短い生産ロットに着目し，早く作業完了する生産ロットを前方に割り付ける方法で，滞留する生産ロットの数が減少し，平均滞留時間等の評価指標には優位に作用する．また，納期余裕優先割付では納期と製造時間の差（納期余裕）に着目し，納期余裕が小さい生産ロットを前方に割り付ける方法で，製造時間が長く納期が短い生産ロットでも納期遅れが少なくなる解を立案することが可能である．

割付方法の選択は評価指標に依存するだけでなく，生産ロットのデータによっても異なる．たとえば，製造時間の長い生産ロットは納期も長い（納期余裕のある）という正の相関を有するデータでは最小製造時間優先割付でも納期優先割付でも同じ生産ロットを優先して割り付け，同等の結果が得られる可能性が高い．これに対して製造時間が長い生産ロットにもかかわらず納期が短い（納期余裕のない）という負の相関を有するデータでは，最小製造時間優先割付と納期

優先割付では異なる生産ロットが優先され，まったく異なる計画結果が得られる．

しかも選択する割付方法は計画立案中の割付状態によっても変化する．未割付の生産ロットに「納期余裕の少ない生産ロット」がなければ，どの割付方法（最小時間優先割付と納期余裕優先割付）を選択しても，同等の結果が得られる可能性が高い．しかしながら，割付を進め，「稼働率の高い装置」が多くなると割付可能な作業開始時刻が遅くなり，「納期余裕の少ない生産ロット」が現れる．納期余裕優先割付を選択して，「納期余裕の少ない生産ロット」を割り付けなければ，納期遅れが発生する．「稼働率の高い装置」や「納期余裕の少ない生産ロット」である等の特徴から割付状態を把握し，その割付状態に適した割付方法を選択することも必要である．

3.3 柔軟な計画立案プロクラムの実現方式

3.3.1 実現の考え方

ディスパッチングルールは最適解を得る立案方法ではないが，計画者の知識や経験を用いて，適切な割付方法を選択することにより，変化する計画問題に対応した生産スケジュールを立案することが可能な方法である．

しかも，この方法は着目変数の値によって割付優先対象を選んで，割付可能な被割付対象に割付優先対象の製造時間を割り付けることを繰り返す方法で，その探索回数（N 件の生産ロット，M 台の装置）は MN（N－１）／２である．表 3.1 の列挙法の探索空間と比べてはるかに小さく，計算機を使用すれば実用的な時間内に計画を立案することが可能である．

そこで計画者の知識や経験を用いて「特徴抽出，戦略決定，割付更新」の計画立案プロセスを繰り返し，適切なディスパッチングルール（割付方法）を選択して割付を進める知識型スケジューリング方式を導入する．

本方式では，

①特徴の定義は対象固有の条件評価であり計画問題定義言語を用いて記述する，

②ディスパッチングルールを選択する条件は特徴量を用いて記述する，

③代表的なディスパッチングルールの割付方法を実装した割付更新プログラムを用意する．本方式の概要を図 3.3 に示す．

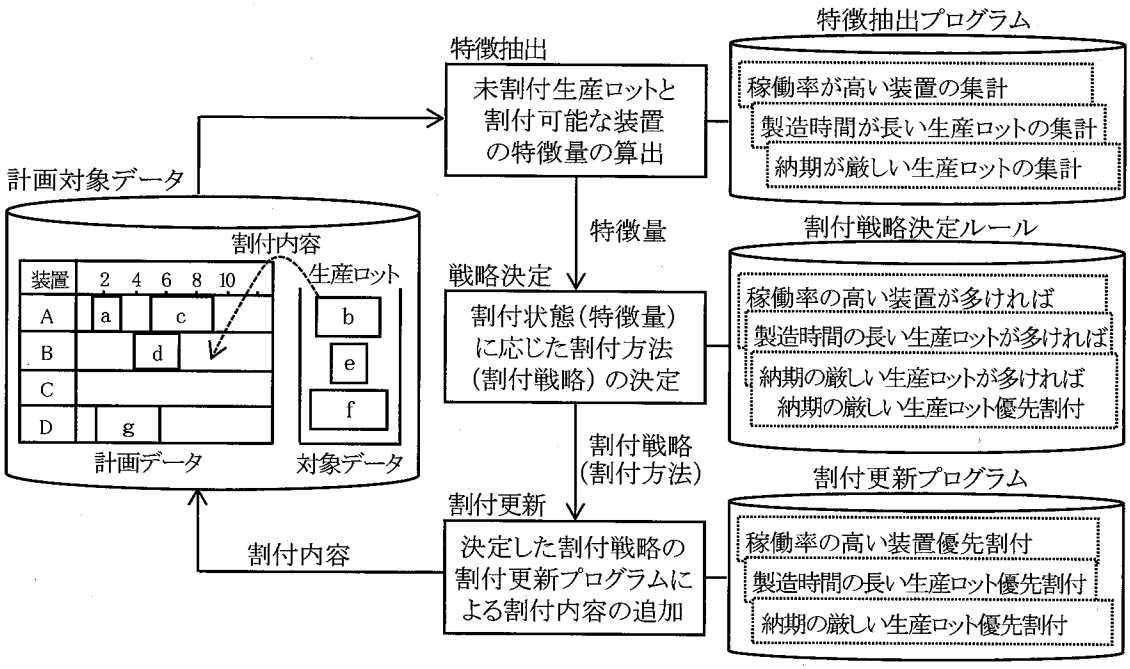

図 3.3　知識型スケジューリング方式

本方式では，まず，計画対象データ（対象データと計画データ）から「納期余裕の少ない生産ロット」や「稼働率の高い装置」等の特徴を示す対象（生産ロットや装置）の数（以下，特徴量と呼ぶ）を算出する．次に，抽出した特徴量に基づき割付方法（以下，割付戦略と呼ぶ）を決定する．そして，割付戦略に対応した割付方法により割付内容（生産ロット，装置，作業開始時刻，作業終了時刻）を計画データに追加する．未割付生産ロットがなくなるまで，「特徴抽出，戦略決定，割付更新」を繰り返し，最終的な計画データを得る．

本方式を実装した計画立案プログラムには，割付戦略に対応した割付更新プログラムの他に特徴量を算出する特徴抽出プログラムと割付戦略を決定する割付戦略決定ルールが必要である．この中には「納期余裕の少ない生産ロット」や「稼働率の高い装置」等の抽出基準の定義（以下，特徴抽出定義と呼ぶ）と「納期余裕の少ない生産ロットが多ければ，納期余裕優先割付」等の割付戦略決定の選択基準が含まれる．これらの定義や基準は計画者の経験知識であり，評価指標や対象データの変化に対応して変更することを必要とする部分である．

そこで，計画者が変更する処理内容を分離記述し，その変更箇所には変更容易な記述形式を提供する．他の処理内容はシステム開発者が処理効率の良い手続き型プログラムを開発する．これにより，計画者の変更容易性と計画立案の処理性能を確保した柔軟な計画立案プログラムを実現する．

特に，割付更新プログラムは割付を優先する割付対象を選択し，製造や段取の作業時間が確保可能な被割付対象に，作業の開始と終了の時刻の計画データを登録するプログラムである．このプログラムの割付優先対象の定義（以下，割付優先定義と呼ぶ）と被割付対象の選択は計画問題定義や割付方法によって異なる．しかしながら，それらの処理は生産ロットを割付優先対象とする処理と装置を割付優先対象とする処理にパターン化が可能である．割付方法の処理パターンを図 3.4 に示す．

（ｉ）生産ロット着目割付
Step1：割付優先対象となる生産ロットを選ぶ．
Step2：選んだ生産ロットの作業時間を確保可能な割付候補（装置）を探す．
Step3：割付候補の中で条件評価値の高い装置に割付優先対象を割り付ける．

（ii）装置着目割付
Step1：割付優先対象となる装置を選ぶ．
Step2：選んだ装置の空き時間で作業時間を確保可能な割付候補（生産ロット）を探す．
Step3：割付候補の中で条件評価値の高い割付優先対象に装置に割り付ける．

図 3.4　割付方法の処理パターン

図 3.4 の生産ロット着目割付は，割付優先対象を生産ロット，装置を被割付対象とする割付処理であり，装置着目割付は，割付優先対象を装置，生産ロットを被割付対象とする処理である．この処理パターンのプログラム（以下，割付基盤プログラムと呼ぶ）をライブラリ化し，割付対象の選択基準（割付優先定義）や被割付対象の選択基準（計画問題定義の条件評価）と組み合わせることによって割付更新プログラムを実現する．割付方法ごとに異なる割付優先対象の定義記述に理解容易な記述形式を与え，割付更新プログラムの変更容易性を確保し，複数の割付更新プログラムの開発・保守工数を削減する．

さらに，特徴抽出プログラムと割付更新プログラムには空き時間を集計する処理や作業時間を集計する処理等の計画データに対する共通処理が含まれる．計画データに対する共通処理を関数（以下，スケジューリング関数と呼ぶ）として提供することにより，スケジューリング問題の計画データに対するアクセスを共通化する．まず，すべての生産スケジューリングに使用可能となる関数（以下，基本関数と呼ぶ）をライブラリ化する．次に，基本関数と対象固有の計画問題定義を組み合わせた関数（以下，拡張関数と呼ぶ）を作成し，特徴抽出プログラムと割付更新プログラムの開発箇所や変更箇所を局所化する．

3.3.2 計画立案プログラムの構成方式と記述形式

(1) 計画立案プログラムの構成方式

計画立案プログラムの繰り返し処理を構造化し，計画者の変更箇所を分離した知識型スケジューリング用計画立案プログラムのモジュール構成を提案する．そのモジュール構成を図 3.5 に示す．

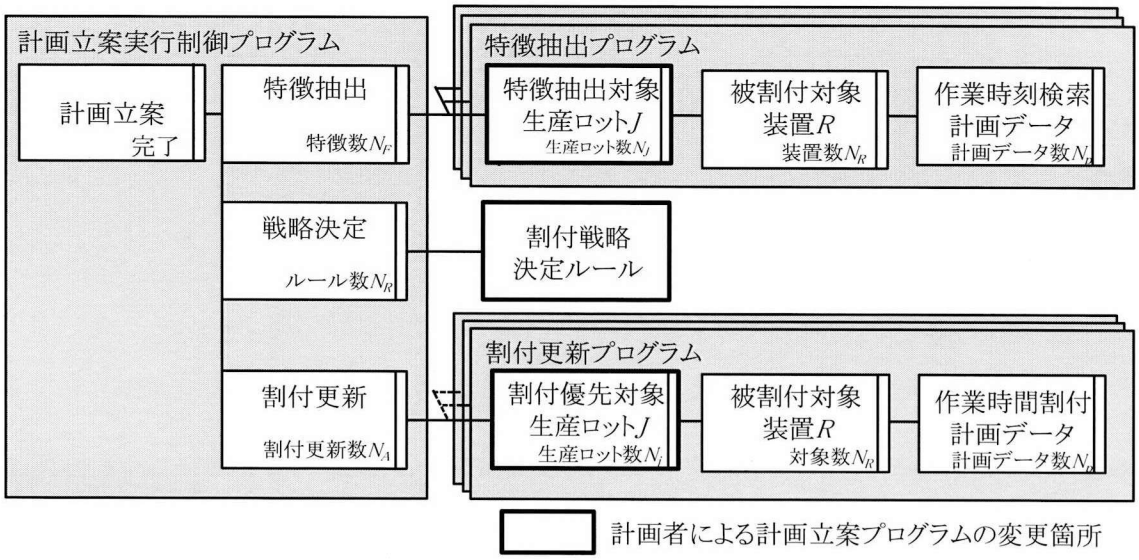

図 3.5　計画立案プログラムのモジュール構成

本構成では，計画立案プログラムを計画立案実行制御プログラム，特徴抽出プログラム群，割付更新プログラム群から構成する．まず，計画立案実行制御プログラムを，特徴抽出プログラムの実行制御処理，割付戦略決定ルールによる割付戦略の戦略決定処理，決定戦略に対応した割付更新プログラムの実行制御処理から構成する．また，特徴抽出プログラムを，特徴抽出定義によって任意の基準を超える割付対象（生産ロット）を抽出する処理，被割付対象（装置）の検索処理，計画データから開始可能な作業時刻を検索する処理から構成する．さらに，割付更新プログラムを，割付戦略に対応した割付優先対象を選択する処理，被割付対象の検索処理，被割付対象に割付優先対象の作業時間を登録する処理から構成する．

これにより，計画立案実行制御プログラムは対象固有の処理を含まず，すべての生産スケジューリング問題で利用可能である．さらに，製造や段取の作業時間をあらかじめ計画問題定義言語で定義し，その条件評価マトリックスを

計算しておくことで，計画者の変更箇所を特徴抽出定義，割付戦略決定ルール，割付更新プログラムの割付優先定義に局所化することが可能となる．

(2) 特徴抽出定義・割付戦略決定ルール・割付更新プログラムの記述形式

特徴抽出定義，割付戦略決定ルール，割付更新プログラムの記述形式として，計画問題定義言語，デシジョンテーブル形式，手続き型記述言語を提案する．その記述例を図 3.6 に示す．

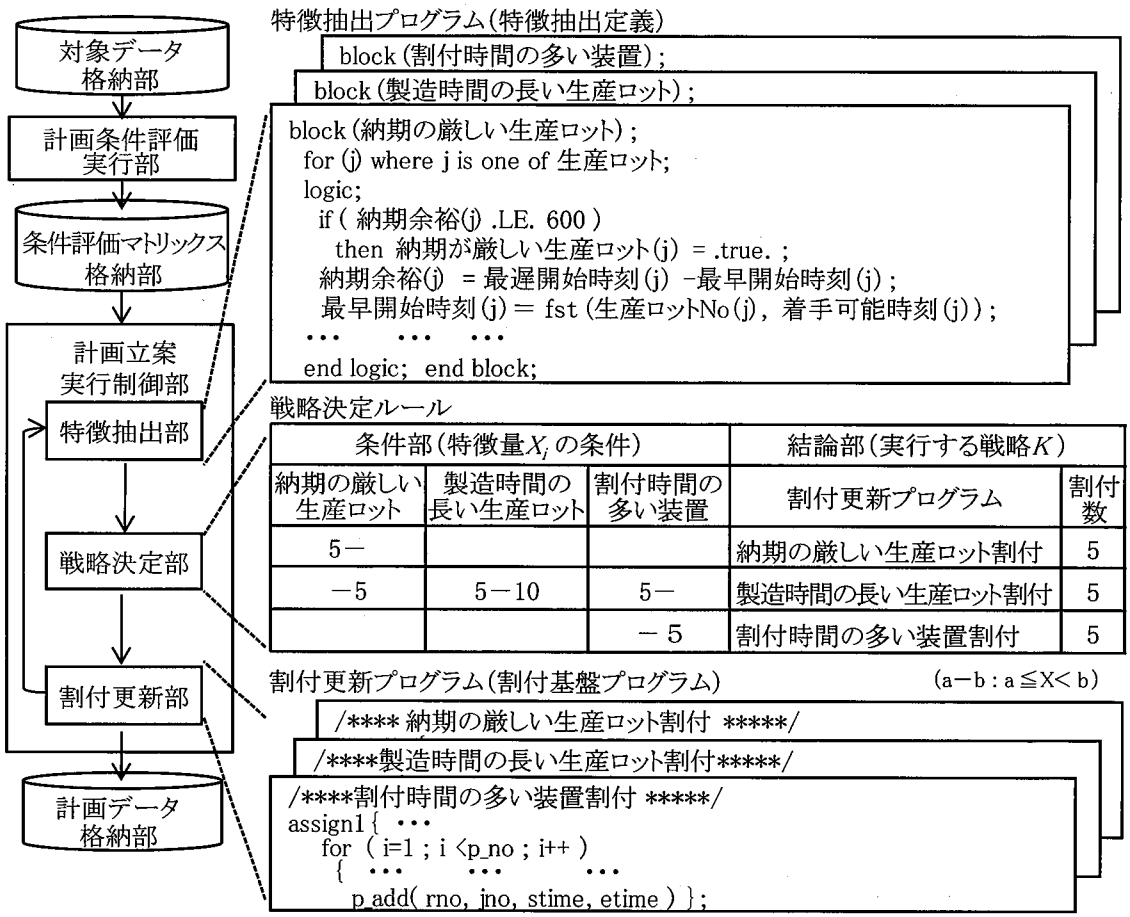

図 3.6 計画立案プログラムの記述形式

(a) 特徴抽出定義の記述

特徴抽出定義は計画問題定義言語を用いて記述する．納期余裕や稼働時間等の計画データを参照する特徴抽出定義の記述では，後述するスケジューリング関数（最早開始時間の検索関数や稼働時間の集計関数等）を用いる．なお，計画問題定義言語で記述した特徴抽出定義は第 2 章で提案したプリ

コンパイラで手続き型プログラムに変換し，計画立案プロクラムのモジュールとして取り込む．これにより，特徴抽出プログラムの変更容易性と処理効率を確保する．

(b) 割付戦略決定ルールの記述形式

割付戦略決定ルール記述は，特徴量の条件部と割付方法（割付更新プログラム）と割付数（1 回の割付更新で追加する割付内容の数）を結論部とするデシジョンテーブル形式で記述する．

割付戦略決定ルール記述に汎用性のある IF-THEN 形式のルール記述を適用すると，条件記述が羅列となり競合や排他となるルールが分かりにくくなる．デシジョンテーブル形式を採用することにより，競合するルールや排他となるルールをわかりやすくし，変更容易性を高める．また，条件部の記述を特徴量の閾値に限ることによって，割付戦略決定の処理内容の記述に限定し，計画問題の制約条件が混在することを防ぐ．さらに，デシジョンテーブルを読み込んで戦略決定を行う処理を手続き型言語で開発し，割付戦略決定ルールの変更容易性と処理効率を確保する．

(c) 割付更新プログラムの記述形式

割付更新プログラムの処理は，2 パターン（図 3.4 の生産ロット着目割付，装置着目割付）に分かれる．あらかじめ，割付優先定義や製造時間を参照する２パターンの割付基盤プログラムを手続き型プログラムで開発しておくことによって，多種多様な割付更新プログラムの効率的な開発保守を実現する．

割付更新プログラムの割付優先定義は計画問題定義言語を用いて記述し，その他の処理内容を手続き型言語で記述する．これにより，割付更新プログラムに含まれる変更部分（割付優先定義）の変更容易性を確保する．その記述内容はプリコンパイラを用いて手続き型言語に変換することにより，割付更新プログラムの処理効率も確保する．

なお，本方式の構成方式と記述形式は生産スケジューリング用のプログラムライブラリ（スケジューリング関数と割付基盤プログラム）を具備することにより，計画者による変更箇所を局所化し，計画立案プログラムの変更容易性と計画立案時の処理性能を確保することが可能となる．

3.3.3 スケジューリング関数と割付基盤プログラム

装置の時間軸上に生産ロットの製造や段取の作業時間を割り付けるスケジューリング問題に対して，汎用的に利用可能なプログラムライブラリを提案する．

(1) スケジューリング関数

生産スケジューリング問題は，装置の時間軸上に生産ロットの製造や段取の作業時間を割り付ける計画問題として取り扱われる．この問題の計画データは，縦軸を装置，横軸を時刻とする計画内容表示画面に出力されることが多い．これは，縦軸を装置，横軸を時刻とした平面上に生産ロットの作業時刻と作業時間を表現することで「納期の厳しい生産ロット」や「稼働率の高い装置」の把握が容易となるからである．計画内容表示画面を用い，特徴抽出や割付更新の実行に必要となるスケジューリング関数の機能要件を図 3.7 に示す．

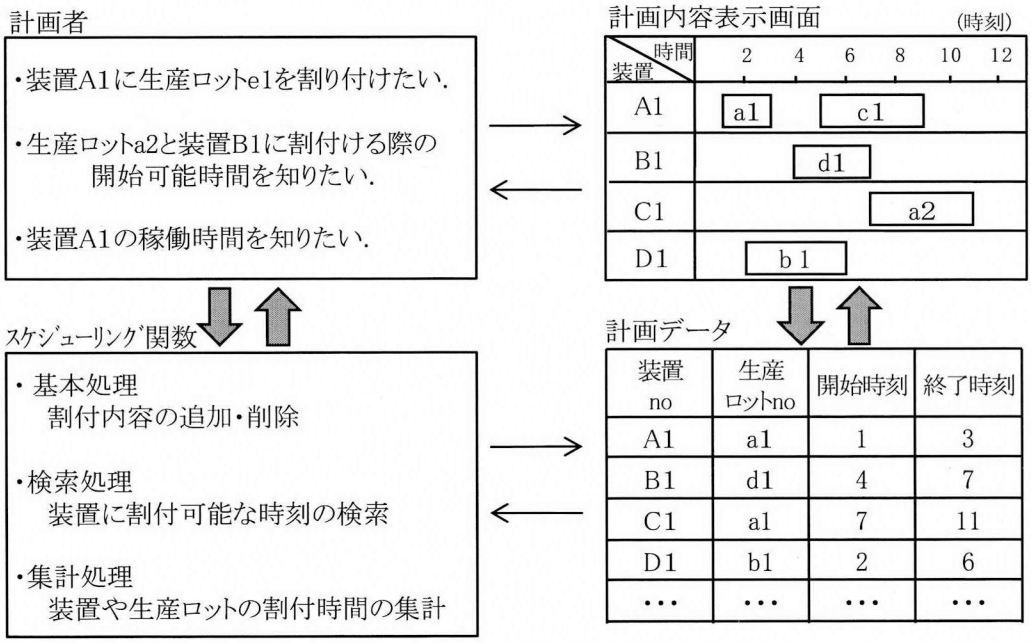

図 3.7　スケジューリング関数の機能要件

計画者は，計画内容表示画面で「稼働率の高い装置」や「納期の厳しい生産ロット」等の特徴を示す計画対象（装置や生産ロット）を抽出し，それらの数に適した割付方法によって計画更新（計画データの更新）を行う．

この特徴抽出では，任意の生産ロットを任意の装置に割り付ける際の作業時間から開始可能時刻や装置に割り付けられた作業時間を参照し，「納期の厳しい

生産ロット」,「稼働率の高い装置」等の特徴を抽出する.この処理を行うには,計画データから開始可能時刻の検索や稼働時間の集計等のデータ処理が必要である.

(a) 基本関数

開始可能時刻の検索や稼働時間の集計等のデータ処理は,計画対象,作業時間,検索や集計の対象期間の指定を除けば,計画対象に依存しない.その基本的な処理は生産スケジューリング問題のすべてで共通化が可能である.そこで,計画データ(生産ロットと装置の組み合わせ,作業時間の開始時刻と終了時刻)を格納するテーブルを用意し,計画対象固有の計画問題定義に依存しない時間軸上の検索や割付時間の集計を行う基本関数を提案する.スケジューリング関数の基本関数を表 3.4 に示す.

表 3.4 スケジューリング関数の基本関数

種別	関数名	機　　能	引　　数
基本処理	p_int	計画テーブルの初期化	—
	p_add	割付データの登録	生産ﾛｯﾄNo, 装置No, 作業開始時刻, 作業終了時刻
	p_del	割付データの削除	生産ﾛｯﾄNo, 装置No
	p_ref	割付データの参照	参照開始時刻・参照終了時刻
検索関数	FSTIME	最早開始時刻の検索	装置No, 検索開始時刻, 検索終了時刻, 作業時間
	LSTIME	最遅開始時刻の検索	装置No, 検索開始時刻, 検索終了時刻, 作業時間
	FETIME	最早終了時刻の検索	装置No, 検索開始時刻, 検索終了時刻, 作業時間
	LETIME	最遅終了時刻の検索	装置No, 検索開始時刻, 検索終了時刻, 作業時間
集計関数	TATIME	期間の割付時間の集計	集計開始時刻, 集計開始時刻
	JATIME	生産ﾛｯﾄの割付時間の集計	生産ﾛｯﾄNo
	RATIME	装置の割付時間の集計	装置No
	JRTIME	生産ﾛｯﾄ・装置の割付時間の集計	生産ﾛｯﾄNo, 装置No
	JTTIME	生産ﾛｯﾄ・期間の割付時間の集計	生産ﾛｯﾄNo, 集計開始時刻, 集計終了時刻
	RJTIME	装置・期間の割付時間の集計	装置No, 集計開始時刻, 集計終了時刻

※ 関数の戻り値:基本処理(完了コード),検索関数(時刻),集計関数(時間)

基本関数(表 3.4)は,基本処理関数(計画データの初期化,更新内容の追加,割付内容の削除と参照の関数群),検索関数(指定期間内で任意装置の作業時間を確保可能な時刻の検索関数群),集計関数(指定期間内の任意の装置や生産ロットの割付時間の集計関数群)とする.

この基本関数に，計画データに対する登録・検索・集計の基本的なデータ処理を包含する．これにより，特徴抽出プログラムと割付更新プログラムの記述内容を削減し，対象固有の変更箇所の局所化を図る．

なお，スケジューリング関数の時間軸は，計画対象によって様々な単位（日，時，分等）で取り扱われるため，0 を起点とした半直線（整数）とし，基本関数の引数は計画対象の識別子，作業時間，指定期間とする．

(b) 拡張関数

基本関数は計画対象に依存しない処理である．すべてのスケジューリング問題に適用可能であるが，この関数内に包含する処理内容は限定的で工数の削減効果が少ない．そこで，製造や段取の作業時間の条件評価マトリックスの値を参照する拡張関数を提案する．拡張関数の例（最早開始時刻 fst）を図 3.8 に示す．

```
/* 最早開始時刻 fst (生産ロットNo, 着手可能時刻)*/
fst (j, stime) { f_stime=999999; etime=999999
  for ( r=1 ; r <r_no ; y++ ) {                /*装置rの探索*/
    f_time= FSTIME( r, stime, etime, operation_time(j. r) );
    if (min_fstime > f_stime) {
        min_fstime = f_stime;/***最早開始時刻更新***/
}; }; };
```

※ FSTIME：基本関数，operation_time：条件評価マトリックス

図 3.8　最早開始時刻 fst の拡張関数

図 3.8 の最早開始時刻 fst は，条件評価マトリックス（operation_time）の値を参照して，装置の中から最も早く作業可能な時刻を検索する関数である．

基本関数（表 3.4）の最早開始時刻の検索関数 FSTIME の引数が装置 No，検索開始時刻，検索終了時刻，作業時間であるのに対して，拡張関数 fst は引数の生産ロット No，着手可能時刻である．基本関数を拡張することで引数を削減し，任意の装置 r の探索を行う繰り返し処理を関数に包含する．これにより，特徴抽出定義や割付更新プログラムの記述内容の削減を可能とする．

(2) 割付基盤プログラム

割付更新プログラムは，割付候補（生産ロット j もしくは装置 r）を選んで，条件評価マトリックスの値を参照し，被割付対象（装置 r もしくは生産ロット j）

に作業時間を割り付ける処理である．割付候補を選ぶ割付優先定義と被割付対象を選ぶ制約条件や評価指標の処理は計画対象ごとに異なるが，その他の処理は図3.4の2パターン(生産ロット着目割付と装置着目割付)の処理となる．

そこで，割付優先定義や条件評価マトリクスを参照して割付更新処理を行う割付基盤プログラムを提案する．割付基盤プログラムの例を図3.9に示す．

```
/**** 生産ロット着目割付プログラム *****/
lot_Assign(p_no) { p=0; f_stime=999999 };                    /* p_no: 割付数 */
  for ( x=0 ; x<jno ; x++ ) {
   if ( assign_lot(x) ) {                                    /* assign_lot(x):割付生産ロット */
    for ( y=0 ; y<rno ; y++ ) {                              /* 割付可能な装置の探索 */
      f_time=FSTIME(y, start_time(x), due_date(x), operation_time(x, y) );
                                  /* 基本関数(最早開始時刻検索), 計画問題定義(作業時間) */
      if ( stime > f_stime ) { stime = f_stime; r = y; };
    };
    p_add(x, r, stime, stime+operation_time(x, r));
                                  /* 基本関数(計画データの登録), 計画問題定義(作業時間) */
    p++; if ( p>p_no ) { x=jno };
 }; }; };
```

図3.9　割付基盤プログラムの例(生産ロット着目割付)

図3.9の割付基盤プログラム（生産ロット着目割付）は，割付優先定義で指定する生産ロットassign_lot(x)に対して，operation_time(x, y)の作業時間が確保可能な装置 y の中で最早可能時刻となる装置に，指定された生産ロットの作業時間を割り付けるプログラムである．このプログラム自身には計画対象固有の定義を含まない．このため，生産スケジューリング問題で汎用的に利用することが可能である．このような割付基盤プログラムをライブラリ化することによって，計画対象ごとの割付更新プログラムの開発保守工数の削減と変更箇所の局所化を図る．

3.3.4　知識型スケジューリングシステムのシステム構成

提案した計画立案プログラムの構成方式と記述形式，プログラムライブラリを実装した知識型スケジューリングシステムの構成を図3.10に示す．

本システムは，全体制御部，計画問題定義や計画立案手順の計画知識編集部，プリコンパイラ，計画問題条件評価部，計画立案手順実行部，計画立案支援部

／開発支援部から構成する．なお，計画立案プログラムの構成要素は，割付戦略決定ルール，特徴抽出プログラム，割付優先定義プログラム，スケジューリング関数，割付更新プログラムである．本システムでは，システム開発者がスケジューリング関数（拡張関数）と割付更新プログラムを手続き型言語で開発し，計画者が対象固有の計画問題定義プログラム，特徴抽出プログラム，割付優先定義プログラムを開発する．

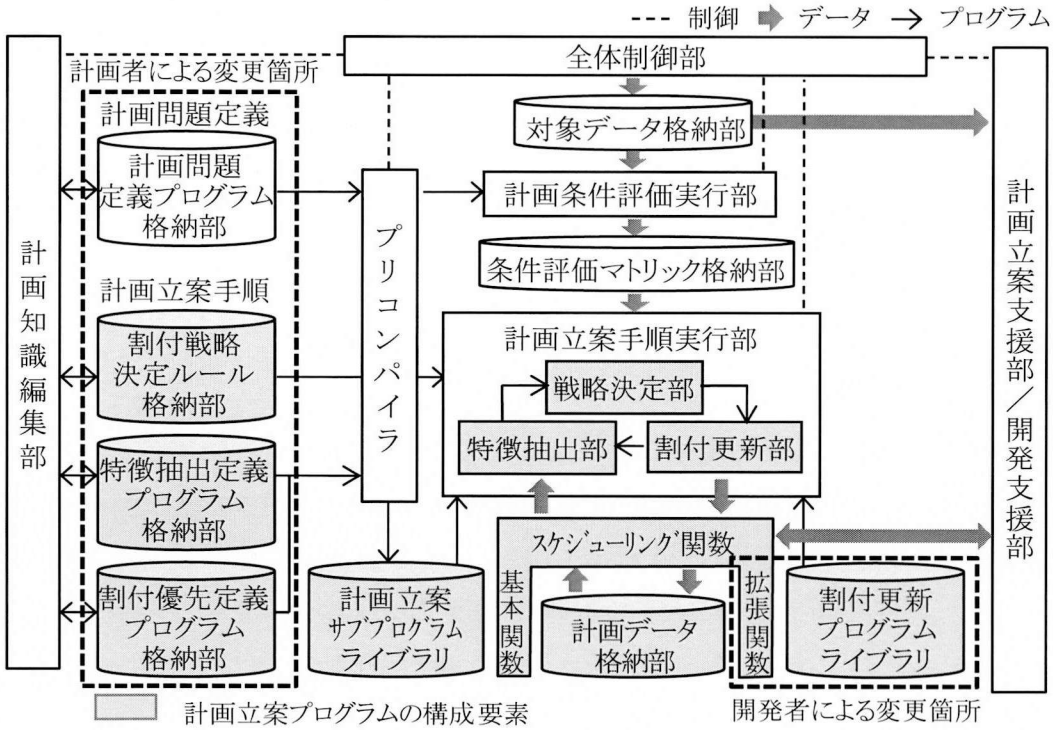

図 3.10　知識型スケジューリングシステムのシステム構成

(1) 全体制御部

計画知識編集部，計画問題条件評価部，計画立案手順実行部，計画立案支援部，開発支援部の実行を制御する．外部システムから対象データを収集・格納する．

(2) 計画知識編集部

計画問題定義言語記述の計画問題定義，特徴抽出定義，割付優先定義のプログラムとデシジョンテーブル記述の割付戦略決定ルールの表示・修正を行う．

(3) プリコンパイラ

計画問題定義言語記述のプログラムを手続き型プログラムに変換する．なお，

特徴抽出定義や割付優先定義のプログラムは計画立案手順実行部から呼び出されるサブプログラムとして計画立案サブプログラムライブラリに格納する.

(4) 計画問題条件評価部

プリコンパイラが生成した計画問題定義の手続き型プログラムにより対象データを入力し，条件評価マトリックスを出力する.

(5) 計画立案手順実行部

条件評価マトリックスのデータを入力し，まず，計画立案サブプログラムライブラリのプログラムを呼び出し，特徴抽出を行う．次に，割付戦略決定ルールにより戦略決定を行う．そして，割付更新プログラムライブラリから決定戦略の割付更新プログラムを呼び出し，割付実行を行い，割付内容をスケジューリング関数によって計画データ格納部に格納する.

(6) 計画立案支援部／開発支援部

計画立案支援では，計画データの表示，特急注文や作業中止等の例外処理による割付内容の追加，削除，移動，交換を行う．開発支援では，「特徴抽出，戦略決定，割付実行」のトレース（特徴量と割付戦略の表示)，割付更新を任意の時点まで戻すアンドゥ，計画者が割付戦略を選択して割付更新を進めるステップ実行等を行う.

3.4 食品加工スケジューリング問題への適用と評価

食品加工は，市場ニーズや季節変動による製品や生産量の変化が起こり，生産工程，加工装置，生産形態，作業人員，製造能力等の変更を必要とする計画対象である．このため，計画対象の制約条件や評価指標の変化とともに，計画立案手順の変更を必要とし，生産スケジューリングのシステム化が困難な生産計画である．そこで，食品加工のスケジューリング問題に，提案方式の構成方式と記述形式を適用し，知識型スケジューリングシステムを用いて，計画者による計画立案手順の変更容易性と，システム開発者による計画立案プログラムの開発保守性を評価する.

3.4.1 計画問題定義と計画立案手順

食品加工スケジューリングは，製品品種ごとに生産工程が異なり，材料到着日と製品納入日の時刻制約を考慮し，生産ロットの加工作業を実施する作業時刻を立案するジョブショップ型（図 3.1(b)）のスケジューリング問題である．

食品加工の生産形態は加工装置ごとに異なり，その製造時間はラインスピード，投入人員，製造数から算出する．材料到着日と食品納入日の間で，その製造時間を確保できる装置を見つけ出し，加工作業の開始時刻と終了時刻を決定する．計画立案は，各装置の空き時間が少なく，生産効率が高くなるように生産ロットの割付を進めるとともに，第 1 工程の開始から最終工程の終了までの時間が長くならないように割付を進める．

本計画対象の計画問題定義，計画立案プログラムの特徴抽出プログラム（特徴抽出定義），割付戦略決定ルール，割付更新プログラムを以下に示す．

(1) 計画問題定義

食品加工スケジューリングの計画問題定義の例を図 3.11 に示す．加工装置の生産形態には，自動化された無人のライン型の作業形態と複数の作業員で食品加工を行うショップ型の生産形態がある．

製造能力（図 3.11）は生産ライン型では製品品種と装置種別によって異なるラインスピードによって定義され，ショップ型では製品の標準加工時間と作業人数よって定義される．なお，作業時間は製造能力と生産ロットの製造数から算出される．

```
block (製造能力) ;
  for (x,y) where x is one of 生産ロット，y is one of 加工装置;
  logic;
    if ( 生産形態(y) .EQ. 'FS' ) /* FS:ライン型，JS:ショップ型*/
      then 製造能力(x,y) = ラインスピード(x,y),
      else 製造能力(x,y) = 標準加工時間(x) / 作業人数(y) ;
    ラインスピード(x,y)
    = ライン能力テーブル
      {製品品種(x) , 装置種別(y)} ;
end logic;  end block;
```

ライン能力テーブル

製品品種	装置種別	ラインスピード
a1	A1	2
b1	A1	3
…	…	…

図 3. 11　食品加工の計画問題定義（製造能力）

(2) 特徴抽出プログラム

「納期の厳しい生産ロット」,「残工程の多い生産ロット」,「割付時間の多い装置」,「加工時間の長い生産ロット」等の 6 つの特徴抽出定義を計画問題定義言語で記述した.「納期の厳しい生産ロット」の特徴抽出定義を図 3.12 に示す.

図 3.12 の特徴抽出プログラムでは，すべての加工装置の時間軸上を検索する拡張関数の最早開始時刻 fst と最遅開始時刻 lst を用いて，最遅開始時刻と最早開始時刻の差を納期余裕として定義している．その納期余裕が 600（分）以下となる生産ロットを「納期の厳しい生産ロット」として特徴抽出するプログラムである.

このように，すべての加工装置（装置）の時間軸上を検索する処理を拡張関数に包含することによって，特徴抽出プログラムの記述内容が簡潔となる．これにより，時間軸上の制約条件（材料到着時刻や出荷時刻）や特徴抽出の閾値（600 分）の変更も容易に行うことが可能である.

```
block (納期の厳しい生産ロット) ;
  for (j) where j is one of  生産ロット;
  logic;
    if ( 納期余裕(j) .LE. 600 )
      then 納期の厳しい生産ロット(j) = .true. ,
      else  納期の厳しい生産ロット(j) = .false. ;
    納期余裕(j)          = 最遅開始時刻(j) - 最早開始時刻(j);
    最早開始時刻(j) = fst(生産ロットNO(j), 材料到着時刻(j));
    最遅開始時刻(j) = lst(生産ロットNO(j), 出荷時刻(j));
end logic;  end block;
```

図 3.12　特徴抽出定義「納期の厳しい生産ロット」

(3) 割付戦略決定ルール

割付戦略決定ルールを表 3.5 に示す．表 3.5 の第 1 行目は,「納期の厳しい生産ロット」が 10 以上かつ「残工程の多い生産ロット」の数が 5 つ以上ある場合,「納期の厳しい生産ロット割付プログラム」を実行し，割付数 5 の更新内容を追加することを定義している.

このようなデシジョンテーブル形式で記述することにより, ルール間の競合や排他がわかりやすくなる．また，生産量（ロット数）の増減にも，閾値を変更するだけで対応することが可能となる.

表 3.5　割付戦略決定ルール

納期の厳しい生産ロット	残工程の多い生産ロット	割付時間の長い加工装置	・・・	割付戦略（割付更新プログラム）	割付数
10－	5－			納期の厳しい生産ロット割付	5
－5	5－	5－		割付時間の多い装置割付	5
－5	5－			残工程の多い生産ロット割付	5
－10				納期優先割付	10

(4) 割付更新プログラム

このスケジューリング問題では，生産ロット着目割付と装置着目割付の割付基盤プログラムから7つの割付更新プログラムを開発した．一覧を表3.6に示す．表 3.6 の割付更新プロクラムは割付優先対象が異なるプログラムである．生産ロット着目割付を基盤とする割付更新プログラムの納期の厳しい生産ロット割付プログラムを図 3.13 に示す．

表 3.6　開発した割付更新プログラム群

割付更新プログラム	割付基盤プログラム
納期の厳しい生産ロット割付	生産ロット着目割付
残工程の多い生産ロット割付	生産ロット着目割付
割当可能装置の少ない生産ロット割付	生産ロット着目割付
加工時間の長い生産ロット割付	生産ロット着目割付
納期着目割付	生産ロット着目割付
割付時間の多い装置割付	装置着目割付
割当可能生産ロットの少ない装置割付	装置着目割付

```
/**** 納期の厳しい生産ロット割付プログラム *****/
ShortSlack Lot Assign(p_no) { p=0; f_stime=999999 };                /* p_no: 割付数*/
 for ( x=0 ; x<jno ; x++ ) {
   if ( ShortSlack Lot (x) ) {                    /* 特徴定義(納期の厳しい生産ロット)*/
    for ( y=0 ; y<rno ; y++ ) {                                        /*  装置の探索  */
     f_time=FSTIME(y, start_time(x), due_date(x), operation time(x,y) );
                                   /* operation_time:計画問題問題定義(作業時間)*/
     if ( stime > f_stime ) { stime = f_stime; r=y; }; };
     p_add(x, r, stime, stime+operation time(x, r));      /*基本関数(更新内容登録)*/
 p++; if ( p>p_no ) { x=jno };
}; }; };
```

図 3.13　納期の厳しい生産ロット割付プログラム

図 3.13 の割付更新プログラム（ShortSlack_Lot_Assign）は，割付優先対象の選択に，特徴抽出定義（図 3. 12）の「納期の厳しい生産ロット」（ShortSlack_Lot）を用い，計画問題定義の条件評価マトリックスの作業時間（operation_time）を参照し，基本関数 p_add により最早可能な加工装置に開始時刻と終了時刻を登録するプログラムである．この割付更新プログラムは，割付基盤プログラム（図 3.9）をベースとして，プリコンパイラが生成したサブプログラム（ShortSlack_ Lot）を呼び出すように書き換えるだけで開発することが可能である．その他の割付更新プログラムもプリコンパイラ生成のサブプロプログラムを利用することにより，開発者によるプログラム開発保守工数を削減するとともに，計画者による割付優先定義の変更容易性を確保することが可能となる．

3.4.2 計画立案手順の変更と計画結果の変化

知識型スケジューリングシステムに計画問題定義（図 3.11 の製造能力）や計画立案手順（図 3.12 の特徴抽出定義，表 3.5 の割付戦略決定ルール，表 3.6 の割付更新プログラム）を登録し，食品加工スケジューリングシステムを構築した．食品加工スケジューリングの計画内容表示画面を図 3.14 に示す．

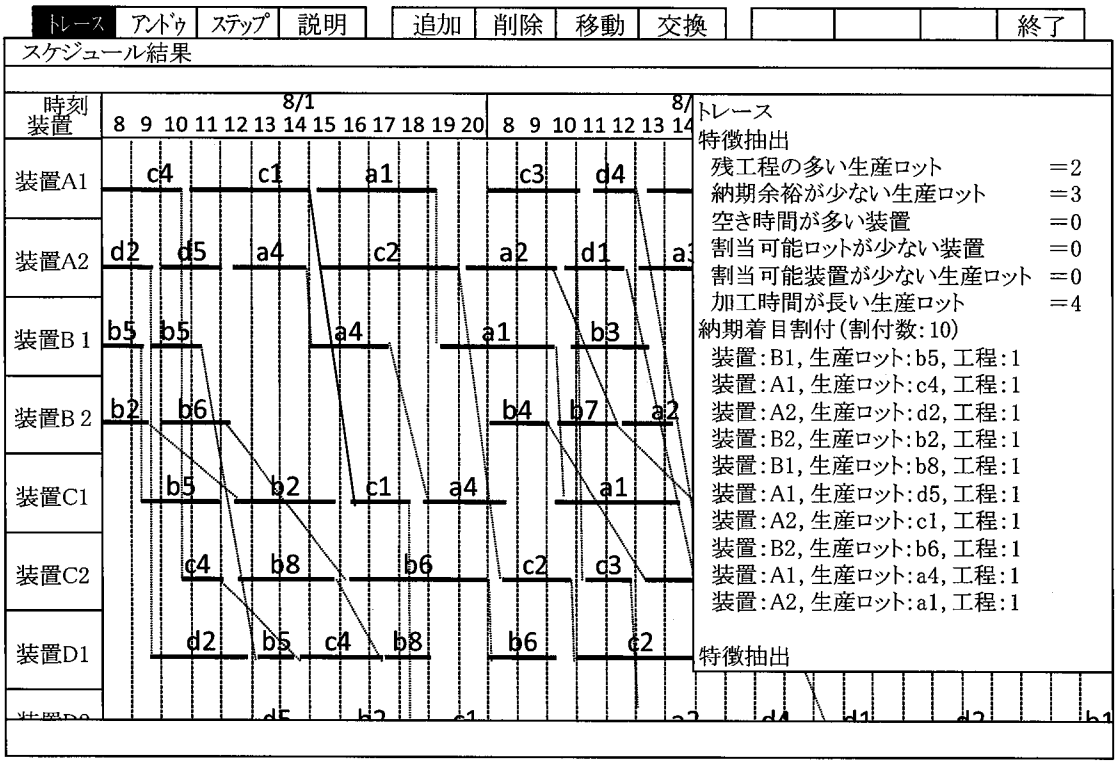

図 3. 14　計画内容の表示例

図 3.14 は，縦軸を装置，横軸を時刻とする平面上に計画した生産ロットの作業時間を示す計画内容表示と，特徴抽出定義で計算した特徴量と割付戦略決定ルールで選択した割付戦略を示すトレース表示の例である．なお，本システムでは，任意の割付状態に戻すこと（アンドゥ）や，計画者自身が割付戦略を選択して実行すること（ステップ実行）も可能である．

既存の特徴抽出定義や割付戦略決定ルールで所望の計画が立案できなくなると，計画者は割付状態に合わせて，特徴抽出プログラムや割付戦略決定ルールを変更する．その例を図 3.15 に示す．図 3.15 の変更前の特徴抽出定義では納期余裕を最遅開始時刻と最早開始可能時刻の差から求めていた．しかし，作業工程の多い注文が増え，残工程の作業時間を考慮しない納期余裕では，納期の厳しい生産ロットが抽出できなくなり，残作業時間を考慮した納期余裕による「納期の厳しい生産ロット」の定義へと変更した．

変　更　前

＜特徴抽出定義＞

```
block(納期の厳しい生産ロット);
 for ( j ) where j is one of 生産ロット;
 logic;
  if( 納期余裕(j) .LE. 600 )
    then    納期の厳しい生産ロット(j) = .true. ,
    else    納期の厳しい生産ロット(j) = .false. ;
  納期余裕(j) = 最遅開始時刻(j) -最早開始時刻(j);

  最遅開始時刻(j)=fst(生産ロットNO(j), 着手可能時刻(j))
  最早開始時刻(j)=lst(生産ロットNO(j); 納期(j));
```

＜戦略定義＞

納期の厳しい生産ロット	残工程の多い生産ロット	割付時間の多い生産ロット	…	割付戦略(割付更新プログラム)	割付数
10−	5−			納期優先割付	5
−5	5−	5−		装置優先割付	5
−5	5−			生産ロット優先割付	5
−10		−5		生産ロット優先割付	10

変　更　後

＜特徴抽出定義＞

```
block(納期の厳しい生産ロット);
 for ( j ) where j is one of 生産ロット;
 logic;
  if( 納期余裕(j) .LE. 400 )
   then 納期の厳しい生産ロット(j) = .true. ,
   else 納期の厳しい生産ロット(j) = .false. ;
  納期余裕(j) = 最遅開始時刻(j) - 最早開始時刻(j)
                                  - 残製造時間(j);
  最遅開始時刻(j)=fst(生産ロットNO(j), 着手可能時刻(j));
  最早開始時刻(j)=lst(生産ロットNO(j); 納期(j));
```

＜戦略定義＞

納期の厳しい生産ロット	残工程の多い生産ロット	割付時間の多い生産ロット	…	割付戦略(割付更新プログラム)	割付数
10−	5−			納期優先割付	5
−5	10−	5−		装置優先割付	5
−5	5−			生産ロット優先割付	5
−10		−5		生産ロット優先割付	10

図 3.15　特徴抽出定義と割付戦略決定ルールの変更

一方，割付戦略決定ルールは納期の厳しい生産ロットに残作業の要因が入ったため，納期の厳しい生産ロット割付（納期優先割付）が優先して選択されるように変更した．図 3.15 の変更前と変更後の計画結果を図 3.16 に示す．

図 3.16 の変更前のスケジュール結果は，工程数の少ない，製品 d が優先して計画されているのに対して，変更後のスケジュール結果は，工程数の多い製品 a，製品 b，製品 c が優先して計画され，その結果として工程数の多い製品の納期遅れが削減される．

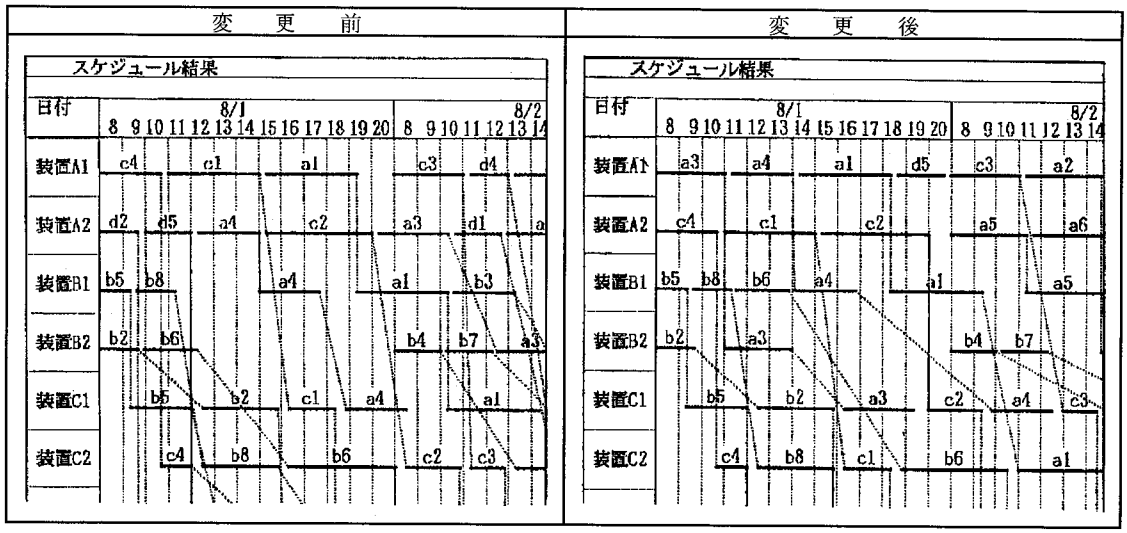

図 3.16　特徴抽出定義と割付戦略決定ルールの変更による計画結果の変化

従来，このような変更は特徴抽出の手続き型プログラムをシステム開発者が変更する必要があったが，本方式では計画者が特徴抽出定義と割付戦略決定ルールを変更するだけで計画者の計画立案手順を反映した計画立案プログラムに変更することが可能である．

3.4.3 考察

提案方式では，計画者が特徴抽出プログラムや割付戦略決定ルールを変更することによって，制約条件や対象データの変化に対応した計画立案手順に変更することが可能である．図 3.15 の例では，納期の厳しさに残作業時間を加味した特徴抽出プログラムに変更することによって，作業工程が多く，残作業時間の長い生産ロットの納期を考慮した計画を立案することが可能となった．

戦略決定だけをルール表現で記述した計画型エキスパートシステムの場合，このような変更はシステム開発者へ依頼する必要があるが，本提案のプログラム構成では，計画者が特徴抽出プログラムや割付戦略決定ルールを直接修正することで頻繁に発生していたシステム開発者による計画立案プログラムの変更作業を削減することが可能となった．

割付更新プログラム（表 3.6）も特徴抽出プログラムで抽出した対象を優先して割り付ける単純なプログラム群となり，割付更新プログラムの開発保守性が向上した．また，割付更新プログラム（図 3.13）の開発も割付基盤プログラムを利用することによって，参照する特徴抽出定義のサブルーチンを変更する

だけで開発することが可能である．本事例で，システム開発者による個別開発部分は，割付基盤プログラム用いた割付更新プログラムと対象固有の制約条件を取り込んだスケジューリング関数の拡張関数（最早開始可能時刻 fst と最遅開始時刻 lst の検索関数）とだけである．

本事例の計画立案プログラム全体（但し，マンマシンインターフェースは除く）のステップ数（C 言語）は約 30Ksteps であった．汎用的に利用可能な計画立案制御部，割付基盤プログラム，スケジューリング関数が 15Ksteps，提案言語記述部分（プリコンパイラの生成コード）が 12Ksteps であり，個別開発部分（割付更新プログラムの改造と拡張関数の開発）は 3Ksteps であった．なお，提案言語記述部分のソースコード（条件評価式と業務テーブル）は約 500 行であった．

本事例では，計画立案プログラムの開発保守を手続き型言語（C 言語）で実施した場合，システム開発者が 30Ksteps を開発保守する必要があるが，提案方式のプログラム構成や記述形式を適用することよって，システム開発者による開発保守部分は 3Ksteps となり，システム開発者による開発保守部分を 10 分の 1 に削減した．

本方式では，拡張関数をスケジューリング用のプログラムライブラリとして蓄積していくことによって，同等の枠組みを持つ計画問題（例えば，作業時間が装置によって異なるスケジューリング問題）で再利用することが可能である．特に，蓄積されたライブラリは，その内部に計画問題特有の多重な繰返し処理を包含し，提案言語による特徴抽出定義の簡易な記述を実現する．これにより，計画者だけで特徴抽出プログラム等の計画立案手順を開発保守していくことが可能となる．

3.5 結言

生産計画支援システムにおいて，計画立案手順の変更容易性を確保するため，生産スケジューリング問題を対象として，計画立案プログラム構成方式と，その記述形式を提案した．

提案の構成方式では汎用的に利用可能な部分（計画立案制御部，割付基盤プログラム，スケジューリング関数）と対象固有部分（特徴定義，戦略決定ルール）

に分け，変更容易性のあるプログラム構成を提案した．また，対象固有部分には計画問題定義言語とデシジョンテーブルの記述形式を与え，汎用的に利用可能な部分は効率的な計算処理の記述が可能な手続き型言語で記述することを提案した．

これにより，対象固有部分は計画者が直接記述変更できるようになり，計画者による計画立案手順の変更容易性を実現した．また，スケジューリング関数や割付基盤プログラムのライブラリ化によって，システム開発者の開発保守工数の削減が可能であることを示した．

提案方式を食品加工のスケジューリング問題に適用し[91]，計画者による計画立案手順の変更容易性と，システム開発者による計画立案プログラムの開発保守工数が削減されることを確認した．提案方式は，第 2 章のプリコンパイラを備えたワークステーション版の知識型計画支援システム HPGS/W とともに，汎用性のある知識型スケジューリングシステム HPGS/W-SCH として，(株) 日立製作所より製品化された．本方式を用いた開発手順[92]も整理され，食品，鉄鋼，衣料，電子部品，半導体，金属加工等の様々な分野で適用された[91, 93]．製品開発後 20 年経過した現在でも，本方式により生産スケジューリングシステムが開発保守されている．

なお，本方式のスケジューリングシステムは，割付戦略決定ルールにより，割付更新プログラムを選択して計画を立案するシステムである．計画結果の良し悪しは割付戦略決定ルールに依存する．このため，そのルール獲得が重要であり，次章で割付戦略決定ルールの獲得方法を提案する．

第4章

統計的集約型知識獲得方式

4.1 緒言

本章では計画者の割付戦略決定ルールを獲得する統計的集約型知識獲得方式を提案する.

第 3 章では，割付戦略決定ルールを用いる計画立案プログラムの構成方式と記述形式を提案し，計画立案手順の変更容易性を図った．この計画立案プログラムでは,割付戦略決定ルールによって適切な割付戦略が選択されなければ,厳しい納期の生産ロットが未割付となり,稼働率の高い装置ばかりに生産ロットを割り付け，装置稼働率が偏り，滞留時間が長く，納期遅れが多い計画結果となる．計画結果の良し悪しは割付戦略決定ルールに依存する.

しかしながら，割付戦略決定ルール自身は，計画問題の解ではなく，ヒューリスティクスな計画立案の経験によって得られる知識である．論理的な根拠があるわけではなく，計画者に何度ヒアリングを実施しても「納期の厳しい生産ロットが多ければ，納期の厳しい生産ロット割付」というあいまいな経験知識しか得られない.

このような経験知識では割付戦略を決定する定量的な特徴量の閾値が得られない．定量的な閾値を得るには，ヒアリングを繰り返すよりも，計画者の割付戦略選択履歴を教師データとして機械学習することが有効である. しかしながら,計画者の教師データには誤教師データやあいまい教師データが含まれ, それらを同等に取り扱うと数多くのルールに分割され，獲得ルール数の増大によって割付戦略決定ルールの変更容易性が失われる.

本知識獲得の目的は，数多くのルールを獲得することではなく，適切な割付

戦略の選択が可能で，変更が容易なルール表現を獲得するところにある．そこで，統計解析手法[94, 95]を用いて，割付戦略決定に対する教師データの有効性を評価し，教師データから誤教師データを削除し，あいまい教師データではルールを細分化しない知識獲得方式を検討する．

以下，第 4.2 節では知識獲得の基本構成と計画者の選択に含まれる誤りやあいまいさを示し，知識獲得の問題点を示す．第 4.3 節では統計的手法とバージョン空間法の長所を利用した提案方式の考え方を示す．第 4.4 節では MAKAM (Multivariate Analytical Knowledge Acquisition Method) のアルゴリズムを提案し，その動作例を説明する．第 4.5 節ではルール獲得実験結果を示し，提案方式により，最適化に有効な解探査空間を有し，計画者が修正しやすい簡潔な割付戦略決定ルールが獲得可能であることを示す．

4.2 知識獲得のアプローチと課題

4.2.1 知識獲得の基本構成

知識獲得処理は計画者の教師データを収集する部分と収集データから割付戦略決定ルールを獲得する部分から構成される．その基本構成を図 4.1 に示す．教師データの収集には第 3 章で説明した知識型スケジューリングシステム（図 3.10）の計画立案支援部のステップ実行機能，計画立案手順実行部の特徴抽出部と割付更新部を用いる．計画者に知識獲得用の対象データを与え，割付戦略決定ルールを使用せず，計画者が割付戦略を選択して計画立案し，所望の計画結果が得られた過程の特徴量と割付戦略を教師データとして収集する．

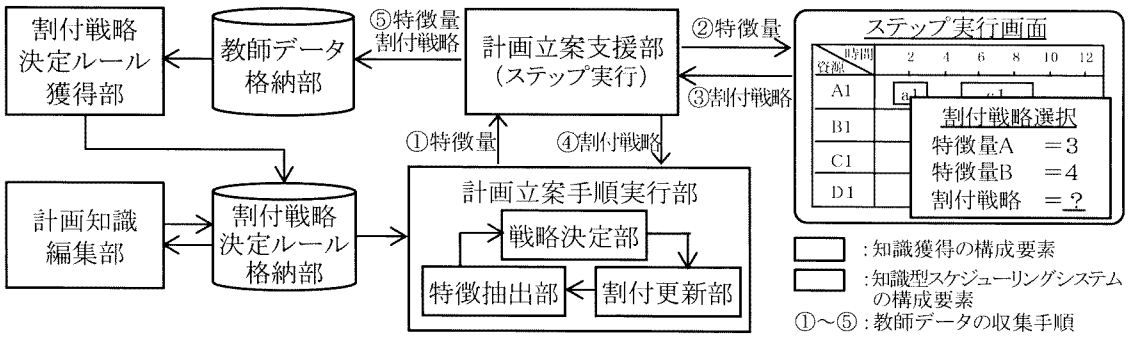

図 4.1　知識獲得の基本構成

割付戦略の教示画面を図 4.2 に示す．計画者が割付戦略を選択する際に「納期の厳しい生産ロット」や「空き時間の少ない装置」を見落とさないように教示画面には，割付内容を示す計画内容表示画面上に特徴抽出部が計算した特徴量を表示する．特徴量の算出と表示，計画者による割付戦略の選択，割付更新の実行，教師データの格納（図 4.1 の①～⑤）を繰返し，教師データを収集する．

収集された教師データの例を表 4.1 に示す．教師データは，所望の計画結果が得られた立案過程（実行ステップ i ）における特徴抽出定義 j の特徴量 X_{ij} と割付戦略 K_i のデータである．この教師データを分析し，割付戦略 K_i を選択する特徴量 X_{ij} の条件を導き出すことで，具体的な特徴量の閾値を持つ割付戦略決定ルールを獲得することが可能となる．

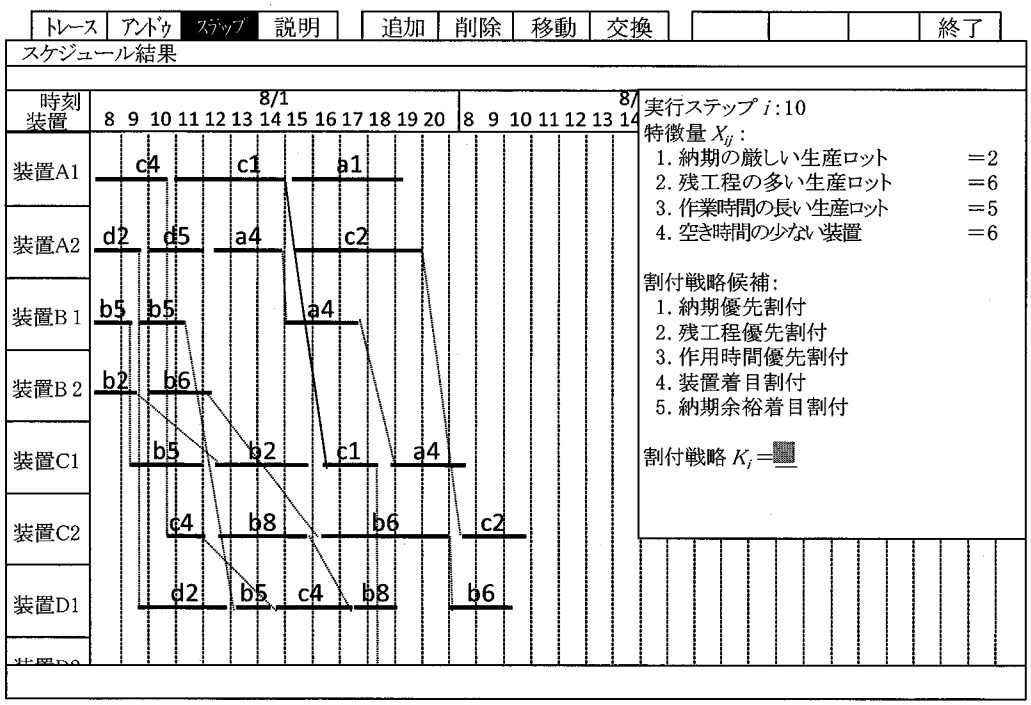

図 4.2　教師データ収集のステップ実行画面

表 4.1　教師データの例

特徴量 X_{ij}				割付戦略
納期の厳しい生産ロット	残工程の多い生産ロット	作業時間の長い生産ロット	空き時間の少ない加工装置	K_i
0	7	7	2	残工程優先
2	6	6	5	作業時間優先
2	6	5	6	装置優先割付
5	3	3	7	納期優先割付
…	…	…	…	…

4.2.2 実現上の課題

計画者による割付戦略選択には計画立案段階ごとに戦略選択の誤りやあいまいさを発生させる以下の要因がある．

①計画立案の初期段階：割付済みの生産ロットの数が少なく，装置の空き時間が多い．割付優先対象となる納期の厳しい生産ロット，作業時間の長い生産ロット，稼働率の高い装置等の特徴を示す計画対象がなければ，割付戦略選択の自由度が大きく，特徴量が同じでも異なった割付戦略を選択する．

②計画立案の中盤：割付済みの生産ロットの数が多くなると，納期の厳しい生産ロットや空き時間の少ない装置等の特徴を示す複数の計画対象が現れる．それらの特徴に応じた割付戦略を選択するが，選択すべき割付戦略が競合し，誤った割付戦略を選択する場合がある．

③計画立案の終盤：未割付の生産ロットが少なくなり，どの割付戦略を選択しても同じ生産ロットが同じ装置に割り付けられる．どの割付戦略を選択しても割付内容が同じとなる．

このような要因があるにもかかわらず，計画結果の評価指標（平均納期遅れや最大滞留時間等）の値は，すべての生産ロットを装置に割りつけてみなければ，得られない．このため，図 4.2 の画面を用いて特徴量や割付戦略候補を表示しても，教師データには誤教師データやあいまい教師データが含まれる．

このような教師データに対して，バージョン空間法等を用いてあいまい教師データ（例えば表 4.1 の 2 行目と 3 行目）を厳密に反映すると，詳細にルールが分割され，ルール数が増加する．詳細に分割された割付戦略決定ルールの例を表 4.2 に示す．

表 4.2　詳細に分割された割付戦略決定ルールの例

戦略選択条件 C_{rj}				割付戦略
納期の厳しい生産ロット	残工程の多い生産ロット	作業時間の長い生産ロット	空き時間の少ない加工装置	K_r
5－8	5－10	5－8	0－5	納期優先割付
8－10	0－5	8－10	5－10	納期優先割付
3－5	0－3	5－8	5－8	装置優先割付
0－3	3－5	8－10	8－10	装置優先割付
…	…	…	…	…
0－5	5－8	0－5	0－5	作業時間優先
5－10	8－10	5－10	5－10	残工程優先

（X－Y: X ≦ C_{rj} < Y）

表 4.2 のように細分化すれば，収集した教師データの再現性は確保される．しかしながら，ルール数が増加すると修正すべきルールが不明確となる．計画問題定義（制約条件や評価指標の定義）の変更に対する割付戦略決定ルールの変更容易性が失われる．

一方，統計解析手法の多変量解析を教師データに用いれば，誤教師データやあいまい教師データを含んでいても割付戦略ごとの判別方程式に集約可能である．判別方程式の例を表 4.3 に示す．

表 4.3　判別方程式の例

判別方程式		割付戦略
$0.500-1.000\ C$	<0	残工程優先
$-1.850-0.555\ A$	<0	作業時間優先
$0.203+0.616\ B-0.367\ A$	<0	装置優先割付
・・・	・・・	・・・

A：納期の厳しい未割付生産ロットの数，　B：残工程の多い未割付生産ロットの数，
C：製造時間の長い未割付生産ロットの数

統計解析手法ではデータ数を増加させると解析結果（判別方程式）の信頼性が向上する．しかしながら，表 4.3 の表現では，どんな優秀な計画者であっても判別方程式を直接修正することはできない．

割付戦略決定ルールは計画問題定義の変更に対して計画者の直接変更を可能とすることも必要である．このルール獲得の課題は，教師データを厳密に反映させることとでも，計画者が変更できない表現形式で獲得することでもない．計画者が変更可能で簡潔なルール表現を獲得し，そのルールによって所望の計画を立案することが課題である．簡潔な割付戦略決定ルールの例を表 4.4 に示す．

表 4.4　簡潔な割付戦略決定ルールの例

戦略選択条件 C_{rj}				割付戦略
納期の厳しい生産ロット	残工程の多い生産ロット	作業時間の長い生産ロット	空き時間の少ない加工装置	K_r
5－				納期優先割付
－5			5－	装置優先割付
	3－			作業時間優先
		8－		残工程優先

（X－Y：$X \leqq C_{rj} < Y$）

戦略決定ルール（表 4.4）は戦略選択条件 C_{rj}（特徴量の閾値 $X \leqq C_{rj} < Y$）と実行する割付戦略 K_rからなり，行数が獲得ルールの大きさ(ルール数)である．獲得ルールの簡潔さの評価指標は，ルール数だけでなく，戦略選択条件 C_{rj} の閾値（表 4.4 の X，Y）の数も対象とする．

また，割付戦略決定ルール（表 4.4）を用いた自動割付ではルールの登録順に（1 行目から）戦略選択条件 C_{rj}をチェックして割付戦略 K_rを選択し，割付状態を更新する．制約条件により割付状態が更新されない場合，次行以降の戦略選択条件 $C_{r'j}$を満足する割付戦略 $K_{r'}$を実行して割付状態を更新する．戦略選択条件 C_{rj} と $C_{r'j}$の重なりは探索空間として利用することが可能である．このルール獲得の課題を以下に示す．

① 戦略選択条件 C_{rj}の閾値が少ない簡潔なルールを獲得すること．

② 獲得ルールをそのまま適用しても準最適な解を導き出すこと．

③ 適切な大きさの探索空間を有するルールを獲得すること．

4.3 提案方式 MAKAM の概要

統計解析手法の長所は誤りやあいまいなデータを含んでいてもデータの集約が可能な点にある．また，データ数を多くすれば分析結果の信頼性が向上する．特に，多変量解析の自由度調整済みの判別効率による変数選択の考え方は判別に寄与しない変数を除外するというものである．これにより，統計的に優位な変数のみによってルールを作成することが可能である．一方，バージョン空間法のルール獲得方法の長所は，「納期が厳しい生産ロットがある」「空き時間の少ない装置が 2 台以上ある」等の単純な判断基準（閾値）からなる簡潔なルールを獲得可能な点にある．

そこで，これらの統計解析手法とバージョン空間法の利点を生かし，判別分析の変数選択によるルール生成空間（判別効率の良い変数の組み合わせ）の絞り込みと，マハラノビスの距離による教師データ（誤教師データ，あいまい教師データ）の区分けを行う．さらに，Suspended Region（戦略選択条件 C_{rj} と $C_{r'j}$の重なり）を導入し，誤教師データ以外の教師データを用い，バージョン空間法により，単純な判断基準からなる簡潔なルールを生成することとする．

(1) Suspended Region の導入

教師データには誤教師データのような誤差が含まれる．誤差を含んでいても解析可能な多変量解析の一手法である判別分析は，任意の特徴量を組み合わせた空間上で判別関数 Z により任意の戦略 $\boldsymbol{K}$ とそれ以外の戦略 $\boldsymbol{notK}$ の領域に分けることが可能である．その例を図 4.3 に示す．

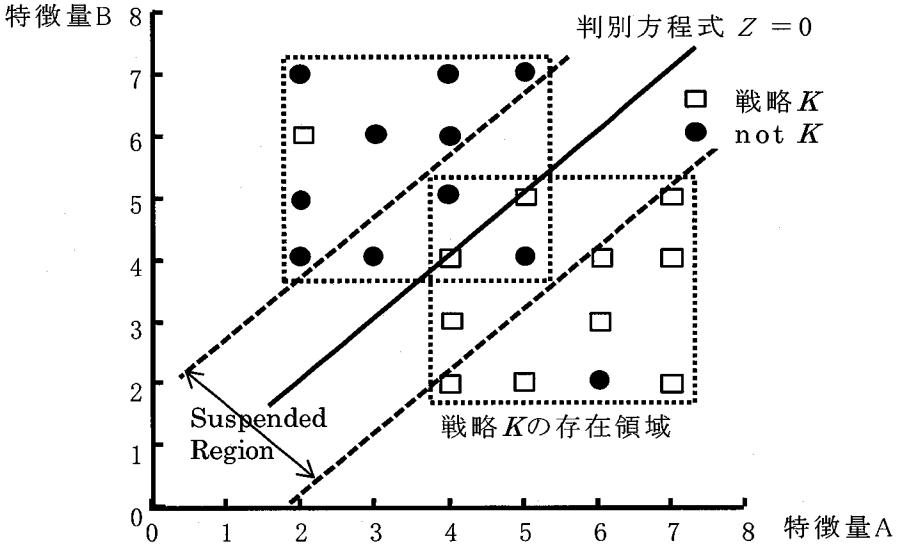

図 4.3　教師データの分布と Suspended Region

この空間（AB 平面）で判別方程式（Z=0）は戦略 $\boldsymbol{K}$ とそれ以外との境界線であり，この境界線に近い教師データはあいまい教師データである可能性が高い．また境界線を超えて，逆の領域に深く入っているデータは誤教師データである可能性が高い．すべての教師データを同等に取り扱い，誤判別率が 0 となるようにルールを生成すると，領域が細分され，生成されるルール数が増大する．

そこで，判別方程式（Z=0）の近傍で複数の戦略が重なる Suspended Region を設ける．その領域では複数の戦略が選択可能な領域とし，計画立案の分岐点（探索空間）とする．計画立案時はこの分岐点で異なる戦略を選択することで最適な解の探索に利用するとともに，細分化されない簡潔なルールを獲得する．

(2) 確信度による Suspended Region の設定

Suspended Region は，計画者が戦略の選択を迷う領域であり，解の探査において分岐点となる探索空間である．しかしながら，計画問題の規模に応じた探索空間としなければ，大規模な問題では探索空間が広がり，解が発散する．

逆に，分岐点をもたなければ，探索空間がなくなり，最適な解を導き出す術がなくなる．そこで，教師データ i が Suspended Region に含まれるか否かを判定する評価尺度として確信度 P_i を導入する．教師データ i の確信度 P_i は，教師データの数（$\boldsymbol{K}$ と $not\boldsymbol{K}$ のデータ数）の違いを考慮し，教師データ i の判別関数 Z の値と各群の判別関数 Z の平均値との割合とする．戦略 $\boldsymbol{K}$ の確信度を P_{1i}，戦略 $\boldsymbol{K}$ 以外（$not\boldsymbol{K}$）の確信度を P_{2i} とし，下記の式で計算する．

$$P_{1i} = \frac{Z_i}{Z_1}, \qquad P_{2i} = \frac{Z_i}{Z_2}$$

Z_i ：教師データ i の判別関数の値

Z_1 : 戦略 $\boldsymbol{K}$ のデータの判別関数の平均値

Z_2 : 戦略 $\boldsymbol{K}$ 以外の戦略 $not K$ のデータの判別関数の平均値

この確信度 P を用いて削除基準 D_e（確信度 P 上の Suspended Region の下限値），あいまい基準 D_a（確信度 P 上の Suspended Region の上限値）を導入する．これにより，削除基準 D_e 以下の教師データはルール生成のデータとして使用しない．削除基準 D_e からあいまい基準 D_a までの教師データをあいまい教師データとしてルール生成に使用する．

(3) 生成空間と生成順序の選択

ルール生成に用いる特徴の数を増やせば判別精度は高くなるが，判別空間が広がり，ルールが獲得しづらく，戦略の判別に不要な条件まで含む．そこで，変数増加による分散を考慮した自由度調整済みの判別効率 ϕ_k[95]により，戦略 $\boldsymbol{K}$ に対してルールが獲得しやすい生成空間 V_k（特徴量の組み合わせ）を決定する．また，判別効率の高い戦略からルールを生成することにより，判定済みの条件に対する排他条件を削除し，より簡潔なルールを獲得する．

(4) ルールの生成

ルール獲得のアルゴリズムとして，複数概念の選言表現の逐次的学習が可能な複合多重集約アルゴリズム[96]を用い，あいまい教師データの場合，ルールの分割（特殊化）を行わないようにアルゴリズムを拡張する．これにより，Suspended Region で複数の戦略が重複した探索空間を持つルールを生成し，獲得されるルールに最適解の探査における戦略決定の分岐点を持たせる．

4.4 提案方式 MAKAM のアルゴリズムと動作例

(1) アルゴリズム

判別分析とバージョン空間法を用いる提案アルゴリズムを図 4.4 に示す.

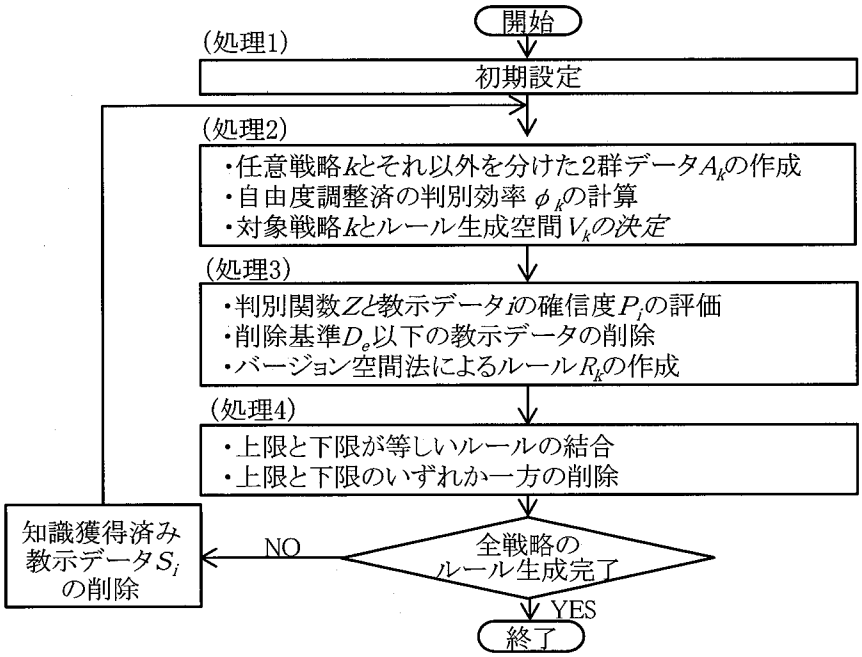

図 4.4 MAKAM のルール生成アルゴリズム

(処理 1) Suspended Region 設定用のあいまい基準 D_a, 削除基準 D_eを設定し, ルール生成用に任意の区間で分割した概念木のデータと教師データ Sを入力し, 割付戦略決定ルール群 Rを初期化（$R_K = \{\phi\}$）する.

(処理 2) 教師データ Sを, 任意の戦略 Kとそれ以外を分けた 2 群データ A_kを作成し, それぞれの 2 群データにより変数選択を行い, 自由度調整済みの判別効率 ϕ_kが最も大きい戦略 Kを選択し, ルール生成戦略とする. 戦略 Kの判別に必要な変数（特徴量）を戦略 Kのルール生成空間 V_kとする.

(処理 3) 戦略 Kのルール R_kを初期化（$R_k=\{\phi\}$）する. 2 群データ A_kにより判別関数 Zを求め, 教師データ iの確信度 P_iを評価, 削除基準 D_e以下の教師データを誤教示データとして削除する. バージョン空間法により, あいまい基準 D_a 以下の教示データをあいまい教示データとして, ルールを分割する細分化を行わず, 上限と下限を持つルール R_kを作成する.

(処理４)　生成された戦略 $\boldsymbol{K}$ のルール R_K の中で，上限と下限が等しいルールを結合する．教師データの誤判別を増やさない上限か下限のいずれか一方を削除し，戦略 $\boldsymbol{K}$ の R_K を戦略決定ルール群 R に追加する．

(処理５)　ルールが生成された戦略の教師データを削除する．

処理2から処理5を繰り返し，すべての戦略の割付戦略決定ルールを生成する．

(2) 動作例

2 特徴量（特徴量 A，B)と 4 戦略（戦略 1, 2, 3, 4）の例を用い，アルゴリズムの動作を説明する．概念木（特徴量の値域を 8 等分した 2 分木）を図 4.5，教師データの分布を図 4.6 に示す．また，各戦略のルール生成ステップにおける変数選択による変数の組み合わせの空間 V（1：必要変数，0：不要変数）と，その判別効率 ϕ_k を表 4.5 に示す．それぞれのステップで生成されたルールを表 4.6 に，最終的に追加される割付戦略決定ルールを表 4.7 に示す．

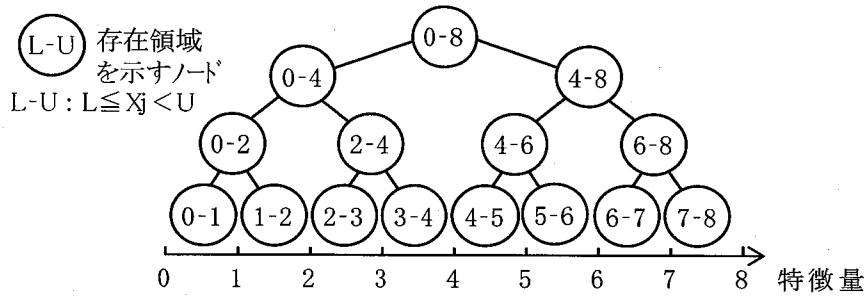

図 4.5　特徴量の概念木の例

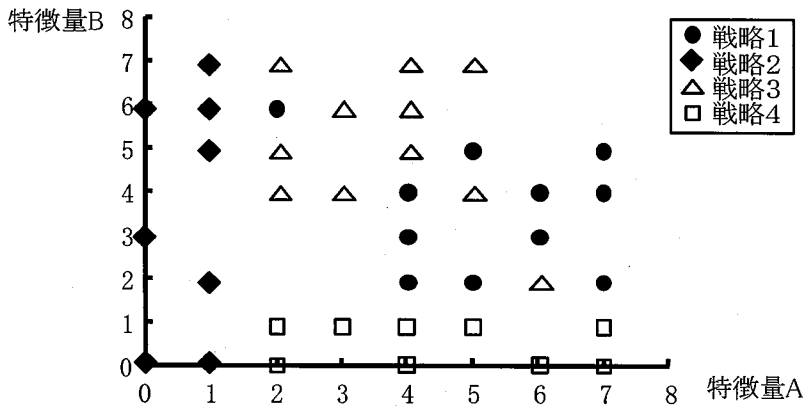

図 4.6　教師データの例

(ステップ 0)　あいまい基準 D_a (0.5)，削除基準 D_e (−0.5)とし，ルール生成で用いる概念木（図 4.5）のデータと教師データ（図 4.6）を入力する（処理 1）.

(ステップ 1)　4 つの戦略 $\boldsymbol{K}$ ごとに変数選択を行い，戦略ごとの判別効率 ϕ_k を求める（表 4.5）. 判別効率 ϕ_k が最も高い戦略 2（表 4.5 の 2 行目）をルール生成戦略として選び，特徴量 A をルール生成空間とする（処理 2）. バージョン空間法により戦略 2 のルール R_k（表 4.6 の 1 行目）が生成される（処理 3）. 生成されたルールの下限の条件が削除され，割付戦略決定ルール $\boldsymbol{R}$（表 4.7 の 1 行目）に追加され(処理 4)，戦略 2 の教師データが削除される（処理 5）.

(ステップ 2)　戦略 2 の教師データ削除により，3 つの戦略 $\boldsymbol{K}$ で変数選択を行い，戦略 4 が 1 つの特徴量で判別でき（表 4.5），戦略 4 がルール生成戦略として選ばれる（処理 2）. 次に，特徴量 B を用いてルール R_k（表 4.6 の 2 行目）が生成される（処理 3）. ステップ 1 と同様に，割付戦略決定ルール $\boldsymbol{R}$（表 4.7 の 2 行目）に追加され（処理 4），戦略 4 の教師データが削除される（処理 5）.

(ステップ 3)　教師データ $\boldsymbol{S}$ に残された戦略が 2 つとなり，変数選択により特徴量 AB 空間上で，戦略 1 と戦略 3 のルールが生成される（処理 2）. 教師データの確信度 P_i 評価し，削除基準 D_e 以下の誤教師データを削除し，あいまい基準 D_a 以下のあいまい教師データでは細分化を行わず，戦略 1 と 3 が重複するルールを持つルール $\boldsymbol{R}$（表 4.6 の 3〜6 行目）が生成される(処理 3). 下限と上限の値が等しく，その他の条件が同じルールが結合され，不要な下限と上限が削除され，戦略決定ルール（表 4.7 の 3〜4 行目）に追加される.

表 4.5　ルール生成空間と生成戦略の選択パラメータ

ステップ	戦略 k	空間 V (A,B)	判別効率 ϕ_k
1	1	(1, 1)	1.1
	2	(1, 0)	6.0
	3	(0, 1)	1.4
	4	(1, 1)	3.6
2	1	(1, 1)	0.4
	3	(1, 1)	2.7
	4	(0, 1)	7.4
3	1	(1, 1)	1.2
	3	(1, 1)	1.2

表 4.6　バージョン空間法によって得られるルール

ステップ	戦略選択条件 C_{rj} 特徴A	特徴B	戦略 K_r
1	0－2		2
2		0－2	4
3	4－8	2－4	1
	4－8	4－6	1
	2－4	4－8	3
	4－6	4－8	3

表 4.7　最終的に得られる割付戦略決定ルール

ステップ	戦略選択条件 C_{rj} 特徴A	特徴B	戦略 K_r
1	－2		2
2		－2	4
3	4－	－6	1
	－6	4－	3

4.5 提案方式の評価

提案方式により獲得されるルールの探索空間と計画性能を評価する２つの実験を行った. 実験１では, 任意の対象データから割付戦略決定ルールを獲得し, 解分布から獲得ルールの探索空間を評価する. さらに, 実験２では, 複数の対象データからルールを獲得し, 教示時と異なる対象データを複数与え, 獲得ルールによる計画性能を評価する.

獲得ルールの質は教示者によって異なるため, ランダムに割付戦略を選択して計画立案を行い, その中で評価指標の高い値が得られた計画の戦略選択を教師データとする. また, 装置ごとに異なる作業順序を決定する汎用的なジョブショップスケジューリング問題は探索空間が広く, 評価値の高い計画が見つからず, 教師データの収集が困難である. このため, 本実験では, 投入順序だけを決定する探索空間が狭いフローショップスケジューリングの計画問題で評価を行う. なお, 提案方式は, Suspended Region 設定により探査空間の大きさに影響を及ぼすため, 一定の Suspended Region(D_e=0.5, D_a=－0.5)と設定し, 生成ルール数や解の分布を評価した.

4.5.1 フローショップスケジューリング問題

評価実験に用いる計画問題は，各生産ロットが２つの作業工程を有し，①各機械は１つの生産ロットしか処理できない，②第一工程が終了してから第２工程の作業を開始する，③２機械で処理される生産ロットの順序は等しいという制約を持ち，計画目的を平均納期遅れの最小化とする典型的なフローショップ型の問題である．２機械での生産ロットの順序は等しいという制約からＮ個の生産ロットの投入順序を決定する問題であり，解の探査空間はＮ！である．Johnson の解法により総稼働時間が最小となる作業順序が立案可能である．計画目的が最大滞留時間の最小化とした場合，最大滞留時間の最小化に優位に働くが，平均納期遅れが最小になるとは限らならない．

割付戦略（割付方法）として，①Johnson の解法に準じて割付を進める Johnson 割付(Johnson)，②納期優先割付(due)，③納期余裕優先割付(slack)，④最小作業時間優先割付(spt)，⑤最大作業時間優先割付(lpt)を用意した．また，特徴量として，未割付生産ロットの中で，Ⓐ納期とⒷ納期余裕が少ない生産ロットの数，Ⓒ総作業時間，Ⓓ第１工程作業時間，Ⓔ第２工程作業時間が大きい生産ロットの数，Ⓕ工程間の作業時間差がある生産ロットの数を抽出した．

4.5.2 探索空間と計画性能の評価

(1) 獲得ルールの探索空間

探索空間の評価は対象データ（作業時間と納期）を変化させるとその影響を大きく受けるため，ルール獲得と同じ対象データを用いて行う．

まず，評価に使用する対象データは，作業時間が異なる 12 件の生産ロットの平均作業時間 250）とし，3 つの生産ロットの納期は 500，残りの生産ロットの納期は 1,000 とする． 次に，ルール獲得はランダムに割付戦略を選択して計画を 100 回立案し，その中で平均納期遅れが小さい計画を立案した 10 回分の割付戦略の選択履歴（10 回×12 選択履歴）を教師データとし，提案方式を用いて平均納期遅れの小さい解を求める割付戦略決定ルールを獲得する．

探索空間の評価は，獲得ルールの Suspended Region で選択する割付戦略をランダムに変更し，出現する解分布を平均納期遅れと最大滞留時間で評価する．さらにランダムに生産ロットや割付戦略を選択するランダム生産ロット選択（rand）とランダム戦略選択（rs）の解分布と比較評価する．

ルール獲得結果（平均納期遅れの小さい教師データを用いた割付戦略決定

ルール）を表 4.8 に，獲得ルール(DMAT1)，rand，rs によって出現する平均納期遅れの統計量を表 4.9 に示す．

表 4.8　獲得ルール(DMAT1)

戦略選択条件 C_{rj}			割付戦略
納期	納期余裕	作業時間差	K_r
−2			納期余裕優先割付(slack)
		−2	納期優先割付(due)
	−3		Johnson割付(Johnson)
		2−	最小作業時間優先割付(spt)
		−3	最大作業時間優先割付(lpt)

表 4.9　出現解の平均納期遅れ（計画目的）

平均納期遅れ	最小(最良)	最大(最悪)	平　均	標準偏差
ランダム仕事選択(rand)	332	724	510	93
ランダム戦略選択(rs)	293	563	393	61
獲得ルール (DMAT1)	275	499	339	38

平均納期遅れ（表 4.9）の標準偏差は，ランダム生産ロット選択(rand)が一番大きく，獲得ルール（DMAT1）が一番小さい．ランダム生産ロット選択は平均納期遅れの最良，最悪，平均の値も悪い．これに対して，獲得ルールは平均納期遅れの最良，最悪，平均の値も良く，平均納期遅れの小さな解を出す可能性が高い．

割付戦略①~⑤の解も含めて平均納期遅れと最大滞留時間の分布を図 4.7 に示す．ランダム生産ロット選択の解が一番広く分布し，獲得ルール（DMAT1）の平均納期遅れと最大滞留時間は比較的小さい領域にまとまって分布する．

Johnson 割付（Johnson）は総作業時間を最小とし，最大滞留時間に関して良好な結果が得られているが，平均納期遅れを最小とする解ではない．最大作業時間優先割付(lpt)は最大滞留時間の比較的小さく，最小作業時間優先割付(spt)は平均納期遅れが小さいところに位置し，納期余裕優先割付（slack），納期優先割付（due）はその中間に位置する．

この解分布は 1 組の対象データ（生産ロットのデータ）によるものである．評価指標の優位性は与えるデータの作業時間と納期の関係によっても変化する．その違いについては次項の実験で示す．

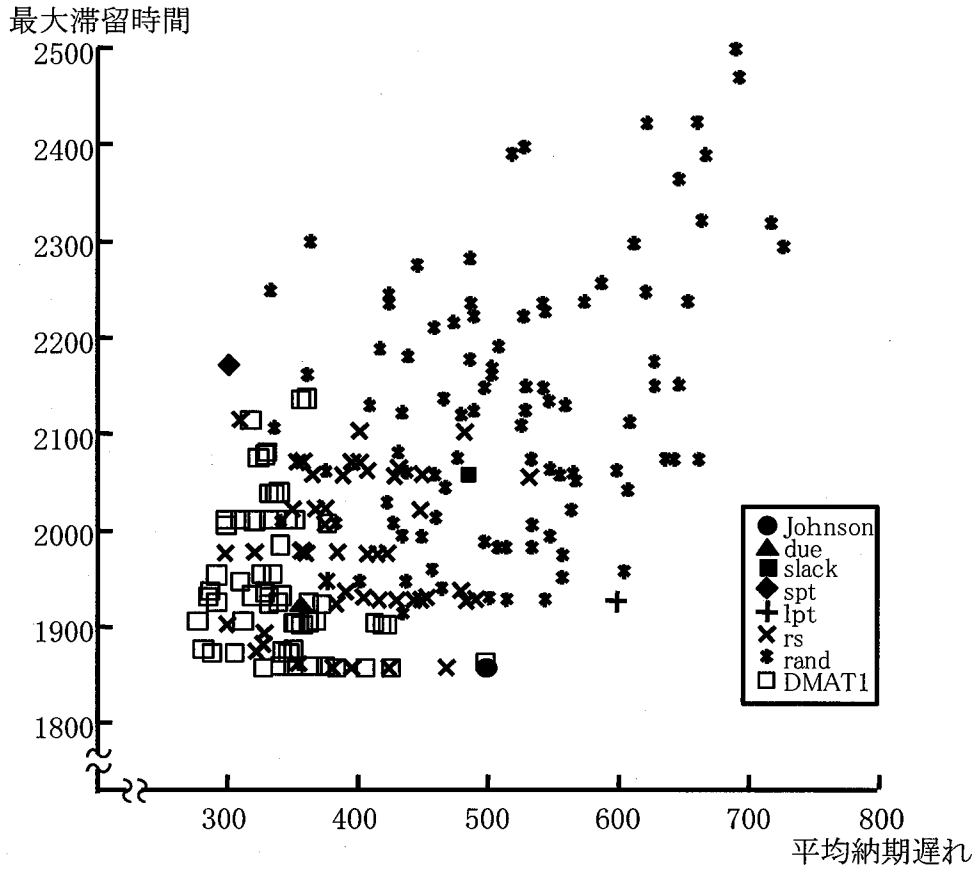

図 4.7　獲得ルールの探索空間（計画結果の分布）

(2) 獲得ルールの計画性能

探索空間の評価では，同じ対象データ（生産ロット）を用いて，ルール獲得と解の分布の評価を実施したが，日々の計画業務では対象データは常に変化する．獲得ルールの性能評価では，ルール獲得と計画立案で異なる対象データを与え，獲得ルールの計画性能性能（平均納期遅れの最小化）を評価する．

ルール獲得用の対象データは，生産ロットの作業時間を 0 から 500 の一様乱数により変化させた 10 組のデータを用意する．そして，探索空間の評価と同様に平均納期遅れの小さい計画結果の割付戦略の選択履歴（10 組×10 回×12 選択履歴）のデータを用いて割付戦略決定ルールを獲得する．

計画性能の評価は，ルール獲得と同様にを一様乱数により生産ロットの作業時間が異なる計画立案用の対象データ（100 件）を用意する．それぞれの割付方法で計画立案を行い，平均納期遅れの値を評価する．なお，本実験では他の割付方法との性能比較を行うため，Suspended Region で複数の割付戦略が選択

可能な場合でも異なる割付戦略を選択せず，ルール登録順で最初に選択される割付戦略を用いて初期解を立案して評価する．

提案方式による獲得ルール（DMAT2）を表 4.10 に示す．DMAT2 は，最大作業時間割付（lpt），納期優先割付（due），Johnson 割付（Johnson），最小作業時間優先割付（spt）の順で割付戦略を適用するルールである．一般的に最大作業時間割付(lpt)が優先されると．作業時間の短い生産ロットが後回しとなり，生産ロットの平均納期遅れを増加させる傾向がある．そこで，最大作業時間割付(lpt)を削除したルール DMAT3（修正ルール）を作成した．その修正ルールを表 4.11 に示す．

表 4.10　10 計画対象データからの知識獲得（DMAT2）

戦略選択条件 C_{rj}			割付戦略
納期	納期余裕	作業時間差	K_r
		−2	最大作業時間優先割付(lpt)
		4−	納期優先割付(due)
	−2		Johnson割付(Johnson)
2−			納期余裕優先割付(slack)
−3			最小作業時間優先割付(spt)

表 4.11　修正ルール（DMAT3）

戦略選択条件 C_{rj}			割付戦略
納期	納期余裕	作業時間差	K_r
		4−	納期優先割付(due)
	−2		Johnson割付(Johnson)
2−			納期余裕優先割付(slack)
−3			最小作業時間優先割付(spt)

作業時間の異なる対象データ（100 件）の計画結果の平均納期遅れの値を図 4.8 に示す．解のばらつき（平均納期遅れの標準偏差及び最大値と最小値の差）は，単一割付戦略では最大作業時間優先割付（lpt）が一番大きく，最小作業時間優先割付（spt）と納期優先割付（due）が比較的小さい．獲得ルールの中では 120 件の教師データから得た DMAT1 が一番大きく，1,200 件の教師データの DMAT 2 が一番小さい．獲得ルールの計画性能（平均納期遅れ）は，その他の割付方法より，最良，平均，最悪とも良い値を得た．

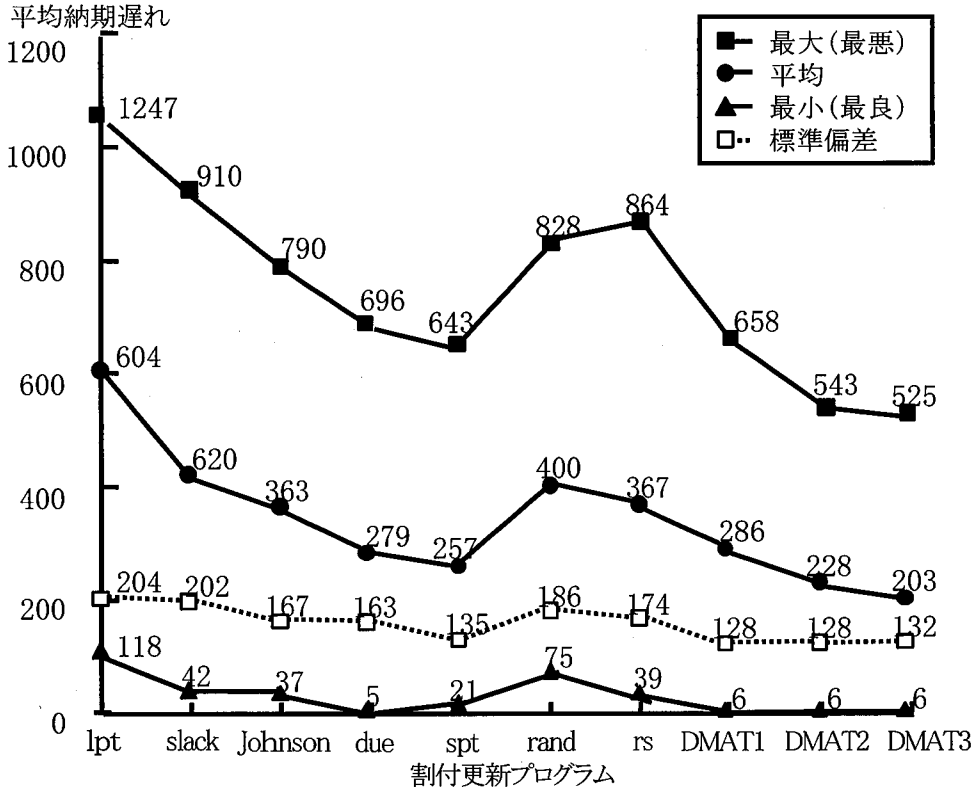

図 4.8　獲得ルールの計画性能（100 対象データに対する計画結果）

4.5.3 考察

２つの実験で収集した教師データからバージョン空間により厳密に出現領域を分割するルールを獲得すると，そのルール数は 50 となる．提案方式での獲得ルール数は 5，判別に必要となる特徴量は 3 であった．提案方式では統計的手法の性質から教師データを増加させることにより，戦略の判定に必要な特徴量が選択され，不要な特徴量は削除される．ルール数だけでなく，割付戦略決定ルールの条件となる特徴量の閾値の数も削減され，獲得ルールは簡潔になる．

実験では教師データ収集に一様乱数を用いたため，誤教師データやあいまい教師データが数多く存在したが，提案方式では簡潔な割付戦略決定ルールを獲得でき，計画者による修正も容易となる．数多くの教師データから信憑性の高い割付戦略決定ルールを獲得することが可能である．また，計画者が獲得ルールを確認し，計画者の経験知識により獲得ルールを修正することで，より最適な割付戦略決定ルールを獲得することが可能になる．

提案方式により獲得された割付戦略決定ルールには解探索に有効な分岐点

（探査空間）を有し，日々変化する対象データに対しても準最適な初期解を導き出す可能性が高い．

生産計画支援システムに Suspended Region で異なる戦略を選択して評価値の高い解を探す最適化機能を設け，Suspended Region（あいまい基準 D_a と削除基準 D_e）を調整することによって，計画問題の規模に応じた探査空間（分岐点）を持たせることも可能となる．

4.6 結言

計画問題用の割付戦略決定ルール獲得方式として統計的集約型知識獲得方式を提案した．本方式では，教誤教師データやあいまいな教師データを評価する教師データの確信度を導入し，Suspended Region を設け，あいまい基準 D_a と削除基準 D_e によって，教師データの取扱いを変えることにより，有効な探索空間を有し，しかも，修正可能で簡潔な割付戦略決定ルールを獲得することが可能である．

これにより，割付戦略決定ルール獲得の容易化が図れる．特に，新しい特徴抽出プログラムや割付更新プログラムを開発した場合に，それらのプログラムが必要であるかを評価可能である．また，獲得するルールの準最適な解を複数導き出す有効な探査空間を有し，削除基準やあいまい基準を調整し，複数の割付戦略が重なってもよい Suspended Region をコントロールすることによって，日々変化するデータにも対応可能な探索空間を有した実用的な割付戦略決定ルールが獲得可能である．

本方式では，あいまい基準 D_a と削除基準 D_e を設定することによって，獲得されるルールの数がコントロールすることが可能であるが，あいまい基準 D_a と削除基準 D_e の設定値が，計画性能やルールの変更容易性に対してどのような影響を与えるかを検討する必要がある．また，本方式では割付戦略決定ルールの獲得を対象としたが，戦略決定には，特徴量の抽出基準にも依存しており，特徴量の抽出基準も含めた戦略決定の知識獲得を検討することも必要である．

第5章 結論

5.1 本研究のまとめ

本研究では，計画問題定義，計画立案手順，計画立案知識の３つの知識情報処理の変更容易性に対して，生産計画問題向きの記述言語，プログラム構成方式，知識獲得方式の提案を行った．

本論文では，研究成果を以下の４章に分けて述べた．

第１章では，計画問題定義と計画立案手順の変更容易性，計画立案の知識獲得の問題の背景と必要性について述べた後，それぞれの問題に対する関連研究の説明を行った．これら３つの問題を解くための方法として，計画問題定義言語とプリコンパイル方式，計画立案プログラム構成方式と記述形式，統計的集約型知識獲得方式の研究方針を述べた．

第２章では，計画問題定義を対象とし，計画問題の制約条件や生産効率の評価指標の値を算定する数式や数表を記述するための計画問題定義言語と，その記述内容を計算機用の手続き型プログラムに変換するプリコンパイル方式の提案を行った．提案言語については専門用語による計画問題定義プログラムの記述仕様と入出力データの定義方法を説明し，変換方式については計算処理性能を確保するためのメモリ割付方法，計算順序と繰返順序の作成方法を説明した．これら提案内容により，計画問題特有の組み合わせ計算の計算処理性能を意識せず，理解容易な順序で計画問題定義プログラムの記述が可能であることを示した．提案内容を実問題に適用し，計画問題定義の変更容易性を確認した．

第３章では，計画立案手順を対象とし，計画問題定義の変化に対し計画立案手順の変更容易性を確保する計画立案プログラム構成方式とその記述形式を提案した．提案内容として，特徴抽出，戦略決定，割付更新を繰り返す計画立案手順を導入し，その計画立案プログラムを計画者が記述内容を変更する

部分とシステム開発者が計算処理性能の良いプログラムを開発保守する部分に分けて取り扱うプログラム構成方式と記述形式を説明した．これらの提案内容により，システム開発者による計画立案プログラムの変更箇所を局所化し，計画者による計画立案手順の変更容易性の確保が可能であることを示した．提案内容を生産スケジューリングシステムに適用し，計画立案手順の変更容易性を確認した．

第 4 章では，計画立案知識を対象とし，計画者の教示データから計画立案手順を選択する割付戦略決定ルールを獲得するための統計的集約型知識獲得方式 MAKAM を提案した．提案内容として，統計解析手法を用い教師データに含まれる誤教師データやあいまい教師データの確信度を評価する方法と，その評価結果を用いてルール表現を獲得する計画立案知識の獲得方式を説明した．これらの提案内容により，計画立案に有効な探索空間を有し，計画者が直接変更可能なルール数に集約されたルール表現が獲得可能であることを示した．提案内容を，フローショップ問題に適用し，ルール数や探索空間の大きさを評価し，提案方式の有効性を確認した．

以上，本研究では，計画対象の制約条件や評価指標の変化に対して，計画問題定義プログラムと計画立案プログラムの変更容易性を確保し，計画者の計画立案ノウハウを用いて生産計画を作成する柔軟な生産計画支援システムを実現した．

5.2 今後の研究課題

本研究では，計画対象の製品や装置の制約条件と生産効率の評価指標の変化に対して，生産計画支援システムの計画問題定義と計画立案手順の変更容易性を確保した．本研究は，日々の生産計画業務を行っている計画者に理解容易な記述形式を与え，計画問題定義や計画立案手順を計画者が変更して生産計画を立案する生産計画支援システムに関する知識情報処理の研究である．

生産計画支援システムの柔軟性を確保しても，そのシステムを取り扱う計画者に計画問題定義や計画立案手順の知識がなければ，生産工程や生産量の変化対応した生産計画の立案はできない．第 1 の研究課題としては，計画者自身の知識獲得問題が挙げられる．特に，計画立案手順の知識は計画立案を繰り返す中で発見的に得られる経験的な知識に基づくものである．計画者が変更した計画立案

手順に従うだけの生産計画支援システムであれば，計画者自身の知識獲得には寄与しないし，計画立案手順の知識を持たない計画者は利用できない．生産計画支援システムが計画立案手順を変更することによって，計画対象の変化に対応できることを計画者自身に気付かせることも必要である．計画支援システム自身が計画立案手順の一部を変更することを繰り返し，その変更によってより良い計画が得られることを計画者に提示し，計画者自身に発見的に得られる経験的な知識の蓄積を可能とすることが今後の研究課題の１つである．

本研究では，生産計画支援システムに取り込んだ計画問題定義と計画立案手順を用いて生産計画を作成するシステムを研究してきた．生産現場では，計画立案後に，特急注文の割り込み，納期短縮，生産量の増減等の変更要求に応じて，変更計画を作成することが頻繁に発生する．この計画変更では，開始作業の手戻りや作業順序の変更等が発生し，関係部署との変更条件や変更タイミングの調整も必要である．その変更条件は多種多様となり，自動的な計画変更のシステム化が困難な問題の１つである．第２の研究課題としては，この計画変更の問題が挙げられる．生産計画支援システムの計画編集機能（追加，削除，移動，交換等）を用いた計画者の手作業では，局所的な変更にとどまり，生産効率低下の原因にもなる．この問題点を解決するには関係部署との変更条件を生産計画支援システムに取り込み，その変更条件に従って，生産効率低下の少ない計画を立案することが必要である．計画内容の変更条件評価に対する記述形式や変更範囲が局所化する計画変更手順を研究することが必要である．

第３の研究課題としては，生産計画以外の計画支援システムへの展開が挙げられる．生産計画支援システムの柔軟性を確保しても，生産を実施するための調達計画や生産した製品の物流計画の柔軟性が欠如していれば，生産工程や生産量の変化にあわせた生産計画を立案することができない．柔軟な生産計画支援システムを効果的に運用していくには，生産計画以外の計画支援システムの柔軟性も確保することが必要である．しかしながら，調達計画や物流計画等の計画支援システムでは計画対象や計画内容のデータの取り扱い方が異なるとともに，計画問題定義や計画立案手順も異なる．本論文で述べた計画問題定義や計画立案手順に変更容易な記述形式を与え，基本的なデータ処理や計画立案プログラムの変更容易な構成を実現するという解決のアプローチは適用可能である．それぞれの計画問題にあわせた計画問題定義や計画立案手順のプログラム構成，それぞれの計画者の知識や経験の記述形式を研究することも必要である．

謝辞

本研究の全過程を通じて，終始懇切丁寧なる御指導と格別の御鞭撻を賜わりました大阪大学大学院情報科学研究科マルチメディア工学専攻 薦田憲久教授に深く感謝申し上げます.

本研究をまとめるにあたり，貴重なお時間を割いて頂き，丁寧なる御教示を賜わりました大阪大学大学院情報科学研究科マルチメディア工学専攻 藤原融教授，原隆浩准教授，秋吉政徳准教授（現 広島工業大学 教授）に深く感謝申し上げます.

大学院博士後期課程において，情報工学全般に関して親切なる御指導と御助言を賜わりました大阪大学大学院情報科学研究科マルチメディア工学専攻 西尾章治郎教授，岸野文郎名誉教授（現 関西学院大学 教授）に深く感謝申し上げます.

本研究の機会を与えて頂くとともに，高所よりご指導とご鞭撻を賜りました(株)日立製作所 システム開発研究所 元主管研究員 三森定道博士，元部長 春名公一博士，元主任研究員 栗原謙三博士（現 神奈川大学 教授）に心より御礼申し上げます.

本研究を進めるにあたり，暖かいご指導とご鞭撻を頂きました(株)日立製作所 システム開発研究所 元主任研究員 田代勤氏（現 (株)日立セキュリティサービス 常務），元主任研究員 村田智洋博士（現 早稲田大学 教授）に心から御礼申し上げます.

また，本研究を進めるにあたり，上司として本論文をまとめる動機付けと機会を与えていただいた，（財）情報処理開発協会 元常務理事 竹田原昇司氏，元常務理事 兼谷明男氏，電子商取引推進センタ 元センタ長 片岡幸一氏に心から御礼申し上げます.

本研究の全般にわたり，製造業の情報システムの有識者として本研究対象とした生産計画支援システムの特徴に関して熱心にご指導とご鞭撻を頂きました(株)日立製作所 セキュリティトレーサビリティ事業部 元副事業部長 中島洋氏（現 CTO, Hitachi Europe Ltd.），産業情報システム事業部 元技師 谷口浩一氏（現 産業・流通システム事業部 部長）に心から御礼申し上げます.

本研究の対象とした知識型生産計画支援システムの最初のユーザとして，熱心にご指導とご鞭撻を頂きました新日本製鐵(株) 広畑製鐵所システム部 山崎朝昭氏，冲中一郎氏（(株)日本ディベロップメント 元会長），森久博博士（現 (株)新日鐵住金ソリューションズ 部長）に心から御礼申し上げます.

本研究の共同研究者として，本研究の全般に亘り熱心にご指導と暖かい励ましを頂きました(株)日立製作所 システム開発研究所 元研究員 大場みち子博士（現 公立はこだて未来大学 教授），(株)日立マイコン 元技師 原敬市氏（現 (株)日立超 LSI システムズ 部長）に心からお礼申し上げます.

本論文の執筆にあたり，何かと便宜を図って頂き，有益なるご助言，暖かいお心遣いを頂きました筆者の所属する薦田研究室の皆様，(株)日立製作所 セキュリティトレーサビリティ事業部 事業部長 中井幸一氏，元主管技師 金野千里博士に厚く御礼申し上げます.

最後に，本論文の執筆にあたり，常に体調を気遣いながら，暖かく励まし支えてくれた妻 真理，息子 宏太郎，娘 真奈ならびに亡き母 節子に心から感謝します.

参考文献

[1] 松島克守：CIM 製造業の情報戦略，工業調査会 (1987).

[2] 黒田充，中根甚一郎，圓川隆夫，田部勉：生産管理，朝倉書房 (1998).

[3] 人見勝人：生産システム工学（第 5 版），共立出版 (2009).

[4] N. Komoda, K. Kera, and T. Kubo: An Autonomous, Decentralized Control System for Factory Automation, *IEEE Computer*, Vol.17, No.12, pp.73-83 (1984).

[5] T. Murata, N. Komoda, K. Matsumoto, and K. Haruna: A Petri-net Based FA(Factory Automation) Controller for Flexible and Maintainable Sequence Control and its Applications in Factory Automation, *IEEE Trans. on Industrial Electronics*, Vol.IE-33, No.1, pp.362-366 (1986).

[6] 渡辺茂，秋山穣：生産システムと最近自動化技術，日本工業新聞社 (1986).

[7] 関根智明：スケジューリングの理論，日刊工業新聞社(1971).

[8] 人見勝人：生産の計画理論，有斐閣双書 (1975).

[9] S. C. Graves: A Review of Production Scheduling, *Operations Research*, Vol. 29, No.4, pp.647-657 (1981).

[10] F. A. Rodammer and K. P. White: A Recent Survey of Scheduling, *IEEE Trans. on Systems, Man and Cybernetics*, Vol.18, No.6, pp.841-850 (1988).

[11] 黒田充，村松健児：生産スケジューリング，朝倉書店(2002).

[12] 小島保郎，奥出聡，斎礼：ポリエステル樹脂工程スケジューリングエキスパートシステム，日立評論，Vol.72, No.11, pp.83-88 (1990).

[13] 福澤英司，森脇英博，仲田正信，長沼正成，品川基：システムキッチン塗装工程計画エキスパートシステム，日立評論，Vol.72, No.11, pp.95-102 (1990).

[14] 大場みち子，薦田憲久，川嶋一宏，原敬市：プロジェクト管理支援エキスパートシステム，第16回SICEシステムシンポジウム，pp.161-166 (1990).

[15] 森久博，馬場伴匠，薦田憲久：ロット分割方式による鉄鋼スケジューリングシステムの開発，電気学会論文誌C，Vol.119, No.10, pp.1289-1296 (1999).

[16] 野村百合子，岩本雅彦，山之内徹，渡辺正信：プロセス産業を対象とした生産計画エキスパートシステム構築ツール，第32回日本OR学会シンポジウム，生産スケジューリング・シンポジウム'94講演論文集, pp.25-30 (1994).

[17] 今井太一，淺田克暢，中川義之，熊本和浩，野平正樹：汎用的アプローチを用いた製鋼工程スケジューリングシステム，システム制御情報学会論文誌，Vol.10, No.4, pp.204-210 (1997).

[18] アディティア・ワルマン，松村健児：段取時間のある多目的多段工程動的ロットサイズスケジューリング：ラグランジュ分解調整法，日本経営工学会論文誌，Vol.53, No.5, pp.385-396 (2002).

[19] 岩村幸治，桑原慎也，谷水義隆，杉村延広：人にやさしい自律分散型生産システムに関する研究－作業者の希望を考慮したリアルタイムスケジューリング，システム制御情報学会論文誌，Vol.22, No.3, pp89-95 (2009).

[20] 関根泰次：数理計画法，岩波書店 (1976).

[21] 古林隆：線形計画法入門，産業図書 (1980).

[22] A. Schrijver: *Theory of Linear and Integer Programming*, John Willey & Sons (1986).

[23] 今野浩，山下浩：非線形計画法，日科技連出版社 (1978).

[24] M. S. Bazaraa and C.M.Shetty: *Nonlinear Programing Theory and Algorithms*, John Willey & Sons (1979).

[25] 今野浩，鈴木久敏：整数計画法と組合せ最適化，日科技連出版社 (1991).

[26] 久保幹雄，田村明久，松井知己：応用数理計画ハンドブック，朝倉書店 (2002).

[27] 森雅夫，松井知己：オペレーションズリサーチ，朝倉書店(2004).

[28] 宮代隆平，松井知己：ここまで解ける整数計画，システム／制御／情報，Vol.50, No.9, pp.363-368 (2006).

[29] 木下順隆：数理計画モデル記述システム OMNI，ソフトウェア流通，No.9, pp.98-103 (1981).

[30] 井上斉:石油精製LPとOMNIの利用, bit, Vol.13, No.13, pp.1646-1653 (1981).

[31] E. F. D. Ellison and G. Mitra: UIMP : User Interface for Mathematical Programming, *ACM Transaction on Mathematical Software*, Vol.8, No.3, pp.229-255 (1982).

[32] 藤井実, 斉藤博一, 横川三津夫, 佐藤治, 安川茂 : 数理計画問題記述簡易言語(PDL/MP)とその処理システム, 情報処理学会論文誌, Vol.27, No.9, pp.880-891 (1986).

[33] A. Tate: *Planning in Expert Systems,* Research Paper 221, Artificial Intelligence Applications Institute, University of Edinburgh (1984).

[34] 石塚満, 小林重信編 : エキスパートシステム, 丸善 (1991).

[35] 入澤毅 : エキスパートシステムを用いた生産スケジューリング, オペレーションズリサーチ, Vol.36, No.3, pp.141-145 (1991).

[36] 関根泰次:プラント分野における計画問題の理論と実際, 人工知能学会誌, Vol.6, No.6, pp.817-822 (1991).

[37] 宮辺裕 : 鉄鋼分野における計画問題の理論と実際, 人工知能学会誌, Vol.6, No.6, pp.811-816 (1991).

[38] 浜崎孝志, 亀田達也, 奥出聡, 小六正彦, 小塚潔 : 計画・スケジューリング形エキスパートシステムへのアプローチ, 日立評論, Vol.72, No.11, pp.41-46 (1990).

[39] 田部勉 : 知識ベースシステム構築のための知識獲得法について, 日本経営工学会誌, Vol.40, No.6, pp.98-104 (1990).

[40] T. R. Grober: Automated Knowledge Acquisition for Strategic Knowledge, *Machine Learning*, Vol. 4, No.3-4, pp.293-339 (1989).

[41] 中須賀真一 : 概念学習による知識獲得を利用した製造ラインのダイナミック・スケジューリング, 計測自動制御学会第15回システムシンポジウム, pp.367-372 (1989).

[42] 淵一博, 溝口文雄, 古川康一 : 制約論理プログラミング, 共立出版(1989).

[43] 丸山哲也, 永田守男 : 制約, ルール, 解に優先度を導入した制約論理型言語の提案と実現, 第46回情報処理学会全国大会, 7F-1, pp.5-11-5-12, (1994).

[44] 坂口隆 : 制約論理プログラミングの探索手法と対話スケジューリング, オペレーションズリサーチ, Vol.47, No.1, pp.16-21 (2002).

[45] 田野俊一，増位庄一，坂口聖二，舩橋誠壽：知識ベースシステム構築ツール EUREKA における高速処理方式，情報処理学会論文誌，Vol.28, No.12, pp.1255-1268 (1987).

[46] 栗原謙三，原敬市，小林隆，汐見龍徳：線形計画法の支援によるルールベース型作業スケジュール方式，情報処理学会論文誌，Vol.30, No.8, pp.976-989 (1989).

[47] M. E. Thomas and P. W. Devid: *Heuristic Scheduling Systems with Applications to Production Systems and Project Management*, John Willy & Sons (1993).

[48] M. Zweben and M. S. Fox: *Intelligent Scheduling*, Morgan Kaufman (1994).

[49] T. Ohkawa and N. Komoda: Current Status of Knowledge Acquisition Research in Japan, in *Proc. of Emerging Technologies for Factory Automation on IEEE/SICE International Workshop*, pp.81-110 (1992).

[50] 黄東明，園川隆夫，秋庭雅夫：成功例に基づく平均納期遅れ基準のスケジューリング問題におけるディスパッチング・ルール生成に関する研究，日本経営工学会誌，Vol.41, No.6, pp.383-389 (1991).

[51] 諏訪晴彦，森田浩，藤井進：帰納的学習法によるスケジューリング・ルールの自動獲得とルールに基づく近傍探索スケジューリングージョブショップ総所要時間最小化問題への適用，システム制御情報学会論文誌 Vol.12, No.3, pp.152-160 (1999).

[52] T. M. Mitchell: Version Spaces, A Candidate Elimination Approach to Rule Learning, in *Proc. of International Joint Conference on Artificial Intelligence (IJCAT '88)*, pp.305-310 (1988).

[53] J. R. Quinlan: Introduction of Decision Trees, *Machine Learning*, Vol.1, No.1, pp.81-106 (1986).

[54] 川嶋一宏，薦田憲久，原田俊一，福田正廣：柔軟な計画を可能とする計画システム開発支援システム用業務論理記述言語とプリコンパイル方式，情報処理学会第 32 回全国大会，4W-6, pp.1897-1898 (1986).

[55] 川嶋一宏，薦田憲久，原田俊一，福田正廣：知識型計画支援システム向業務論理記述言語用プリコンパイラ，情報処理学会第 33 回全国大会，3W-5, pp.2065-2066 (1986).

[56] 川嶋一宏，薦田憲久，原田俊一：知識型計画支援システム向業務論理記述言語用プリコンパイラの最適化機能，情報処理学会第 34 回全国大会，1T-8, pp.1089-1090 (1987).

[57] 川嶋一宏，薦田憲久，原田俊一，三森定道：知識型計画支援システム向業務論理記述言語用プリコンパイラ，情報処理学会論文誌，Vol.28, No.9, pp.975-986 (1987).

[58] K. Kawashima, N. Komoda, S. Harada, and S. Mitsumori: Precompiler for Process Logic Description Language in Intelligent Planning Support System, in *Proc. of 1987 Annual International Computer Software & Applications Conference (COMPSAC'87)*, pp.649-655 (1987).

[59] 川嶋一宏，薦田憲久，原田俊一：知識型計画支援システム向業務論理記述言語用プリコンパイラの拡張機能，情報処理学会第 35 回全国大会，6D-5, pp.2637-2638 (1987).

[60] 川嶋一宏，薦田憲久，原田俊一：知識型計画支援システムの多段処理機能，情報処理学会第 36 回全国大会，7Q-1, pp.1585-1586 (1988).

[61] 川嶋一宏，原敬市，薦田憲久：知識型計画支援システム HPGS によるスケジューリング問題記述方式，情報処理学会第 39 回全国大会，2B-4, pp.155-156 (1989).

[62] 原敬市，薦田憲久，川嶋一宏：知識型計画支援システムHPGS によるスケジューリングシステムの開発，情報処理学会第 39 回全国大会，2B-5, pp.157-158 (1989).

[63] 原敬市，川嶋一宏，大場みち子，薦田憲久：知識型計画システムの開発手順，情報処理学会第 40 回全国大会，2D-5, pp.238-239 (1990).

[64] 川嶋一宏，原敬市，薦田憲久，大場みち子：知識型計画システム用データ基本操作機能，情報処理学会第 40 回全国大会，2D-6, pp.240-241 (1990).

[65] 大場みち子，薦田憲久，川嶋一宏：知識型計画システムにおける計画結果説明機能，情報処理学会第 40 回全国大会，2D-7, pp.242-243 (1990).

[66] 大場みち子，薦田憲久，川嶋一宏：計画向制約指向型説明機能の説明文選択方式，情報処理学会第 41 回全国大会，5K-2, pp.2-45-2-46 (1990).

[67] 大場みち子，薦田憲久，川嶋一宏：知識型計画システムにおける制約指向型説明機能, 情報処理学会論文誌, Vol.31, No.11, pp.1688-1695 (1990).

[68] 川嶋一宏，薦田憲久，原敬市，大場みち子：知識型計画システム，第 7 回

SICE 産業システムシンポジウム，pp.5-8 (1990).

[69] 大場みち子，薦田憲久，原敬市，川嶋一宏：対話型スケジューリングシステムにおける改善知識型最適化方式，情報処理学会第 42 回全国大会，2N-2, pp.1-289 -1-290 (1991).

[70] 原敬市，大場みち子，薦田憲久，川嶋一宏：最適化機能を有する知識型スケジューリングシステムの開発，情報処理学会第 42 回全国大会，2N-3, pp.1-291-1-292 (1991).

[71] 川嶋一宏，薦田憲久：計画システムにおける統計的集約型知識獲得方式，情報処理学会第 41 回全国大会，5K-1, pp.2-43 - 2-44 (1990).

[72] 川嶋一宏，薦田憲久：知識型計画システムにおける統計的集約型知識獲得方式の提案，第 12 回 SICE 知能工学シンポジウム，pp.39-44 (1990).

[73] 川嶋一宏，薦田憲久：知識型計画システムにおける知識獲得機能の拡張，情報処理学会第 42 回全国大会，2N-4, pp.1-293 -1-294 (1991).

[74] K. Kawashima and N. Komoda: Multivariate Analytical Knowledge Acquisition Method for Knowledge based Planning System, in *Proc. of IEEE International Conference on Industrial Electronics Control and Instrumentation (IECON'91)*, pp.1622-1627 (1991).

[75] 川嶋一宏，原敬市：学習機能を有する知識型スケジューリングシステムの開発，電気学会 電子・情報・システム部門 第 1 回全国大会，D-4-6, pp.362-363 (1991).

[76] 川嶋一宏，薦田憲久：知識型計画システムにおける統計的集約型知識獲得方式，情報処理学会論文誌，Vol.34, No.5, pp.864-872 (1993).

[77] 司馬正次，阿部 昌信，吉川経教：PL/I 入門，倍風館 (1974).

[78] 石田晴久：プログラミング言語 C，共立出版 (1988).

[79] 中田育夫：コンパイラ，産業図書 (1981).

[80] 山崎朝昭，冲中一郎：鉄鋼業におけるＡＩの応用ー生産計画システムに対する知識工学の応用ー，日本機械学会誌，Vol.89, No.815, pp.1194-1197 (1986).

[81] (株)日立製作所：HITAC APP 知識型計画支援システム HPGS 文法，APP-A-424-10 (1988).

[82] (株)日立製作所：HITAC APP 知識型計画支援システム HPGS 概説・利用の手引き，APP-A-424-10 (1988).

[83] 冲中一郎，山崎朝昭：鉄鋼業における生産計画への知識工学適用事例，第5回知識工学シンポジウム, pp.45-50 (1987).

[84] R. W. Conway, W. L. Maxwell, and L. W. Miller: *Theory of Scheduling*, Addison-Wesley (1967).

[85] S. S. Panwalkar and W. Iskander: A Survey of Scheduling Rules, *Operations Research*, Vol.25, No.1, pp.45-61 (1977).

[86] M. Pinedo: *Scheduling: Theory, Algorithms and Systems*, Prentice Hall (1995).

[87] 田中克己，石井信明：スケジューリングとシミュレーション，計測自動制御学会 (1995).

[88] 福井哉，橋本文雄：納期遅れ最小化のためのディスパッチング・ルールに関する研究，日本経営工学会誌, Vol.46, N0.4, pp.248-253 (1995).

[89] 冬木正彦，井上一郎：バックワード／フォワード・ハイブリッドシミュレーション法に基づく個別受注生産における納期重視型生産スケジューリング，日本経営工学会誌，Vol.46, No.2, pp.144-151 (1995).

[90] 柏原秀明，宮崎茂次：自律分散型製造装置モデルによる総利益最大化バックワード・フォワード・スケジューリング，日本生産管理学会論文誌, Vol.16, No.2, pp.25-26 (2010).

[91] 原敬市，薦田憲久，川嶋一宏，大場みち子：知識型スケジューリングシステムの開発，日立マイコン技報，Vol.4, No.1, pp.56-61 (1990).

[92] 川嶋一宏：スケジューリングシステムの導入と支援ツール，次世代生産管理システムの構築と運用，pp.687-709，アーバンプロデュース (1992).

[93] 橋ヶ谷俊男，山崎昌哉，永井聡，松本晃幸：EP8000 による生産管理システム構築事例：コーテック株式会社，日立評論，Vol.72, No.11, pp.43-46 (2003).

[94] 永田靖，棟近雅彦：多変量解析法入門，サイエンス社 (2001).

[95] 奥野忠一，芳賀敏郎，矢島敬二，奥野千代子，橋本茂司，古川陽子：続・多変量解析法，日科技連出版社 (1976).

[96] 田代勤，薦田憲久：複数概念の選言表現の逐次的学習のための複合多重集約アルゴリズム，情報処理学会論文誌，Vol.38, No.9, pp.1073-1082 (1989).