博 士 論 文

人間の持つ解決戦略を利用した問題解決システムの構築及びその設計に関する研究

指導教官　　下平博　助教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

飯田 栄治

平成 11 **年** 7 **月** 22 **日**

目 次

1 序論 **1**

1.1 はじめに 1
1.2 “問題解決” とは 1
1.3 問題解決システム 2
1.3.1 解決すべき対象問題の分類 2
1.3.2 問題の記述性と解決アプローチ 4
1.4 従来の問題解決システム 5
1.5 問題解決システムの構成要素 7
1.6 従来の問題解決システムの問題点 7
1.7 社会の動向 8
1.8 研究の背景 9
1.8.1 本研究の背景 9
1.8.2 近年の研究の動向 10
1.8.3 本論文における研究の立場 10
1.9 研究の目的と効果 11
1.9.1 本研究の目的 11
1.9.2 研究対象となるシステムの特徴 12
1.9.3 本研究効果の応用可能性 13
1.10 本研究における重点検討項目 14
1.11 本論文の構成 14

2 人間の問題解決に関する一考察 **17**

2.1 はじめに 17
2.2 使用する用語の定義 17
2.2.1 問題解決のコストと最適性 17
2.2.2 解の質 18
2.2.3 解決戦略 18
2.2.4 エキスパート 18

2.3 人間の問題解決の事例 . . . 18
2.3.1 １５パズル . . . 19
2.3.2 追跡ゲーム . . . 22
2.4 人間の問題解決に対する仮説 . . . 25
2.5 まとめ . . . 26

3 提案する問題解決方式 27
3.1 はじめに . . . 27
3.2 問題解決方式の提案 . . . 27
3.2.1 提案する方式の問題解決方法 . . . 30
3.3 提案する問題解決方式の数学的定式化 . . . 32
3.3.1 問題に関する定義 . . . 33
3.3.2 問題のグラフおよびパス . . . 34
3.3.3 パスが問題の解である条件 . . . 34
3.3.4 スマートな解 . . . 34
3.3.5 部分問題 . . . 35
3.3.6 状態評価関数 . . . 35
3.3.7 オペレータの影響 . . . 36
3.3.8 単調な状態評価関数に基づく問題 . . . 41
3.3.9 特定の状態変数の解決 . . . 42
3.3.10 フェーズ・フロー . . . 43
3.3.11 フェーズフローの可解性 . . . 44
3.3.12 フェーズ木 . . . 46
3.4 可解性および解決不能状態に関する考察 . . . 48
3.5 制約を伴う問題解決 . . . 49
3.5.1 問題空間に制約を伴う問題 . . . 49
3.5.2 実時間制約 . . . 50
3.6 まとめ . . . 51

4 提案方式による問題解決システムとその設計 53
4.1 はじめに . . . 53
4.2 システムの枠組み . . . 53
4.3 システム構成 . . . 54
4.3.1 入力インターフェース . . . 56
4.3.2 解決戦略データベース . . . 57
4.3.3 問題発見モジュール . . . 58

4.3.4 問題選択モジュール . . . 59
4.3.5 問題解決モジュール . . . 60
4.3.6 出力インターフェース . . . 61
4.4 システムの問題解決プロセスの分析 . . . 62
4.4.1 入力インターフェース . . . 62
4.4.2 問題発見処理 . . . 63
4.4.3 問題選択処理 . . . 64
4.4.4 問題解決処理 . . . 66
4.4.5 出力インターフェース処理 . . . 68
4.5 提案システムと従来システムの構成の違い . . . 69
4.6 提案システムの設計法 . . . 69
4.6.1 従来のルールベースシステム構築法 . . . 69
4.6.2 提案システムの設計法の概要 . . . 72
4.6.3 提案システムの具体的な設計ステップ . . . 73
4.7 まとめ . . . 77

5 実験及び評価 79
5.1 N^2-1 パズル . . . 81
5.1.1 はじめに . . . 81
5.1.2 問題の定義 . . . 81
5.1.3 問題解決システムの設計（その１） . . . 82
5.1.4 問題解決システムの設計（その２） . . . 88
5.1.5 実験及び結果 . . . 91
5.1.6 評価 . . . 94
5.1.7 まとめ . . . 94
5.2 追跡ゲーム (Pursuit Game) . . . 95
5.2.1 はじめに . . . 95
5.2.2 問題の定義 . . . 96
5.2.3 問題解決システムの設計（その１） . . . 97
5.2.4 問題解決システムの設計（その２） . . . 103
5.2.5 実験及び結果 . . . 105
5.2.6 評価 . . . 107
5.2.7 まとめ . . . 107
5.3 航空機制御 . . . 108
5.3.1 はじめに . . . 108
5.3.2 問題の設定 . . . 108

5.3.3 問題解決システムの設計（その１） . . . 109
5.3.4 実験及び結果 . . . 116
5.3.5 評価 . . . 121
5.3.6 まとめ . . . 121

6 実験結果に対するディスカッション 123
6.1 人間の持つ問題解決戦略について . . . 123
6.1.1 被験者の妥当性 . . . 123
6.1.2 問題の妥当性 . . . 123
6.1.3 未経験者の解決戦略 . . . 123
6.2 計算機実験の結果に対する個別ディスカッション . . . 124
6.2.1 可解性 . . . 124
6.2.2 問題解決のための知識内容とその記述量 . . . 124
6.2.3 処理時間（実時間性） . . . 125
6.2.4 開発及び改修の容易性 . . . 125
6.3 計算機実験の結果に対する総括 . . . 125

7 結論 129
7.1 はじめに . . . 129
7.2 本研究の主題 . . . 129
7.3 本研究にて明らかとなった事実 . . . 130
7.3.1 人間の持つ解決戦略の有効性 . . . 130
7.3.2 人間の解決戦略を用いる問題解決システムの定式化 . . . 130
7.3.3 人間的な解決戦略の抽出及び実装 . . . 130
7.3.4 人間的な解決戦略を用いるシステムの問題解決能力 . . . 131
7.3.5 提案方式が有効に機能したことに対する理由 . . . 131
7.4 今後の課題 . . . 131
7.4.1 現状の技術で解決可能な課題 . . . 132
7.4.2 論理中心の問題と合成問題への応用 . . . 132
7.4.3 今後の展望 . . . 132

謝辞 133

参考文献 135

本研究に関する発表論文 143

Appendix **145**

A スマートな解の証明 **147**

B N^2-1 パズルの計算量分析 **149**
B.1 アルゴリズムとその分析 149
B.1.1 アルゴリズムの比較 149
B.1.2 解の長さに関する分析 153

C N^2-1 パズルの追加実験 **155**
C.1 実験及び結果 155
C.1.1 アルゴリズムの比較 155
C.1.2 評価 156
C.1.3 まとめ 156

D 追跡ゲームの実験に関する初期データ **161**
D.1 データの定義 161
D.2 データ 162

E 航空機操縦問題の初期設定データ **167**
E.1 データの定義 167
E.2 実験における初期データ 167

F 駒落ち将棋システムへの応用 **169**
F.1 はじめに 169
F.1.1 将棋プログラムとその研究 169
F.1.2 駒落ち将棋 170
F.2 問題の定義 171
F.3 フェーズフロー構築のための知識表現 172
F.3.1 メインフェーズフロー 173
F.3.2 サブのフェーズフロー 174
F.3.3 フェーズフローチャート 176

図 目 次

1.1 問題の存在 2
1.2 問題の分類構成 2
1.3 問題の性質と解決アプローチ 4
1.4 問題解決システムの構成要素 7
1.5 解決対象問題の変化 8
1.6 システム性能 9
1.7 本研究の対象領域 11
1.8 新たなアプローチ 11
1.9 部分問題空間の結合 12
1.10 論文構成 15

2.1 １５パズルの例 19
2.2 ルールベースシステムの処理概要 20
2.3 １５パズルの結果 21
2.4 追跡ゲームの例 22
2.5 追跡ゲームの結果 24

3.1 問題解決モデル 28
3.2 部分問題のグラフ 29
3.3 フェーズ木 29
3.4 フェーズ木の展開 30
3.5 問題解決プロセス 31
3.6 定式化の体系 32
3.7 relevant 及び irrelevant 36

4.1 システム構成 55
4.2 入力インターフェース 56
4.3 解決戦略データベース 57
4.4 問題発見モジュール 58

4.5 問題選択モジュール . . . 59
4.6 問題解決モジュール . . . 60
4.7 出力インターフェース . . . 61
4.8 ルールベースシステムの構築手順 . . . 69
4.9 フェーズフロー図解例 . . . 76

5.1 N^2-1 パズルの例 . . . 81
5.2 N^2-1 パズルの解決過程 . . . 81
5.3 N^2-1 パズルの知識表現 . . . 83
5.4 オペレータの基本パターン . . . 84
5.5 可解性（指定タイルの移動） . . . 85
5.6 可解性（後戻りしないための戦略） . . . 86
5.7 N^2-1 フェーズフロー . . . 87
5.8 サブゴールの選択 . . . 89
5.9 サブゴールの定義 . . . 89
5.10 N^2-1 パズルのフェーズフロー（その２） . . . 90
5.11 解の長さと平均計算時間 . . . 91
5.12 解の長さと計算時間の分布 . . . 92
5.13 解の長さの分布 . . . 93
5.14 search horizon の違いにおける解の長さと計算時間の分布 . . . 93
5.15 追跡ゲーム必勝パターン . . . 95
5.16 追跡ゲーム解決例 . . . 96
5.17 追跡ゲームの知識表現 . . . 98
5.18 エージェント移動パターン (8 方位) . . . 99
5.19 L 字フォーメーションの構築過程 . . . 100
5.20 追い詰め戦略 . . . 101
5.21 追跡ゲームのフェーズフロー . . . 102
5.22 解決ストーリー . . . 103
5.23 追跡ゲームのフェーズフロー（その２） . . . 104
5.24 解の長さと計算時間の平均値 . . . 105
5.25 解の分布状況 . . . 106
5.26 Y 軸回りの変化 . . . 110
5.27 X 軸回りの変化 . . . 110
5.28 Z 軸回りの変化 . . . 111
5.29 速度変化 . . . 111
5.30 航空機モデル . . . 112

5.31 オペレータ . . . 113
5.32 航空機操縦における問題空間 . . . 114
5.33 航空機操縦における制御の流れ . . . 114
5.34 操縦フェーズフロー . . . 115
5.35 TEST1 および TEST2 の解決性能の比較 . . . 117
5.36 実験２の設定（その１） . . . 118
5.37 航空機の実時間探索制御における航跡の変化 (先読み深さ 1,3,5) . . . 119
5.38 提案手法による着陸までの航跡の変化 . . . 120

6.1 解の質と計算時間 . . . 126
6.2 適用対象 . . . 127

B.1 実時間探索アルゴリズム . . . 150
B.2 提案手法のアルゴリズム . . . 152

C.1 RTA^*と提案手法 . . . 157
C.2 上限値、提案手法 および 最適解の解長さ . . . 157

D.1 追跡ゲームデータ定義 . . . 161

F.1 １０枚落ち将棋の初期配置 . . . 172
F.2 １０枚落ち将棋の知識表現 . . . 173
F.3 解決プロセス概要 . . . 174
F.4 解決プロセス概要 . . . 176

表 目 次

1.1 問題の分類 3
1.2 研究対象システムの特徴とそのメリット／デメリット 12
1.3 問題の難易度と応用の可能性 13

2.1 人間の問題解決に対する仮説 25

4.1 解決戦略データベース 54
4.2 従来のシステムと提案手法の開発の流れの違い 72
4.3 フェーズフローのための関係記述記号 75

5.1 例題の特徴 79

6.1 各例題に対する解決知識と記述量 124

C.1 平均の解長さ 155
C.2 8 パズルに関する実験例題及び結果 158
C.3 15 パズル実験例題 159
C.4 24 パズル実験結果（SD 解法） 160

E.1 実験における初期データ 168

F.1 駒落ちのパターン 171

第 1 章

序論

1.1 はじめに

本論文は、人間の持つ解決戦略を有効利用し、ユーザーの要求を満足する問題解決システムを、比較的短時間に構築するための方法論に関する研究をまとめたものである。以下のステップに従って議論を進める。

- 人間の持つ問題解決戦略の有効性について考察する。
- 人間の持つ解決戦略を利用した問題解決方式を提案し、数学的に定式化する。
- 人間の持つ解決戦略を利用した問題解決システムの設計法を提案する。
- 提案するシステムの性能を例題をとおして評価する。

本章では、その準備として研究の背景等について述べる。

1.2 “問題解決” とは

まず、本文中、頻繁に出現する用語 “問題解決” の概念について述べる。

人工知能の研究対象は、人間の認識、判断、推論、問題解決、発話や行動の司令、さらに学習などがあり、究極的には、「頭脳の機能を機械によって実現することを目指す」学問分野である。それは、1950 年のシャノンのチェスマシンの論文 [Shannon 1950] にさかのぼる。“問題解決” は、ゴールと利用可能な手段が明確に定義されている問題に対し与えられた状態を探索または何らかの推論を用いながらゴールに導く（または、ゴール状態に変換する）ことである。別の見方をすれば、問題解決とは、概念世界にある要求と現実世界の状態との間に生じているギャップを埋める行為のこと [Nishioka 1998] と見ることができる。また、逆に、このようなギャップが存在している場合は、解決すべき “問題”

が存在していることになる。たとえば、図 1.1では、現実世界の積木の状態は、概念世界の要求を満足していない (ギャップが存在する) ので “問題” が存在していることになる。

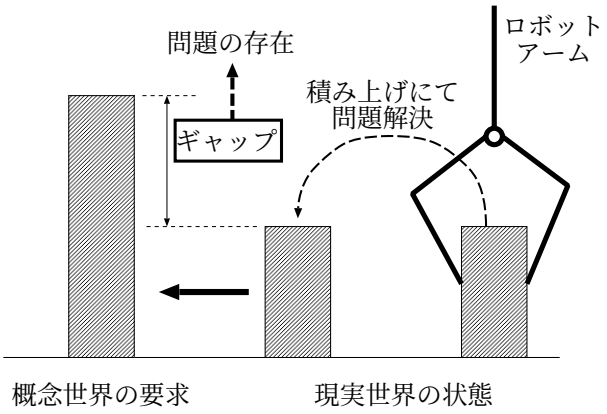


図 1.1: 問題の存在

1.3 問題解決システム

「問題解決」を行なうシステムを問題解決システムという。

1.3.1 解決すべき対象問題の分類

一般に、問題解決システムの解決対象は、図 1.2 および、表 1.1 に示すように、現在、大きく、解析問題と合成問題に分類され、さらに、解析問題は、解釈問題、診断問題、制御問題に分類され、また、合成問題は、計画問題と設計問題に分類できる。しかし、実際には、複合的な問題も存在する。

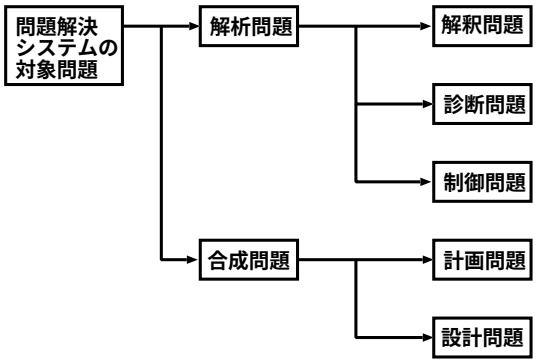


図 1.2: 問題の分類構成

表 1.1: 問題の分類

	解釈問題	診断問題	制御問題
定義	解釈問題とは、計測器やセンサーをとおして観測された連続的なデータを分析して、システムの構造や状態を推定し、これに物理的な意味づけを与えることである	診断問題とは、システムに異常または故障が生じたときに観測されたデータ及びシステムに関する知識を利用して原因の箇所を同定することである。	制御問題とは、システムの状態を監視していて、予め予定されたとおりにシステムの状態が遷移するように操作を加えることである。
特徴	1. 時空間データの高次利用 2. 雑音の混入 3. 誤りの存在 4. 必要なデータの欠落 5. 信号処理技術との接続 6. 対象に関するモデルの利用 7. 感覚データの扱い	1. 設計レベルの知識の利用 2. 経験的知識の利用 3. 経験的知識の不確実性 4. 操業データの利用 5. 計測のための時間とコスト 6. 知識抽象の必要性 7. 対話的診断の推論制御	1. 時間遅れによる履歴依存性 2. 非線系性による局所性 3. 安定化の応答性 4. 制御精度の実現 5. 実プロセスとの接続 6. 実時間性と信頼性の確保 7. 離散系における解析問題
基本タスク	1. 特徴抽出モデル 2. モデルとの不完全な整合 3. システム構造の同定 4. システム状態の推定 5. 曖昧さの処理 6. 不完全性の処理 7. 解釈の多様性	1. 異常事態の分類階層的表現 2. システム構造の階層的表現 3. 解釈問題のタスクを内包 4. 計測点の選択的決定 5. 異常原因の同定 6. 浅いモデルによる効率性 7. 深いモデルによる完全性	1. システム構造の表現 2. システム動特性 3. 診断タスクを内包 4. モデルによる状態予測 5. 安定化操業の優先的実行 6. 安定操業下での省エネ化 7. オペレータガイダンス
問題解決機能	1. 分散階層によるクラス分け 2. モデル検索 3. 部分構造の評価 4. 階層的生成検査法 5. 不確実性推論 6. 仮説推論 7. 協調推論	1. 事象駆動型推論 2. 目的駆動型推論 3. 不確実性推論 4. 仮説推論 5. 協調型推論 6. 効率性と完全性の調和 7. 費用対効果分析	1. 状態の診断機能 2. シミュレータによる予測 3. 定性推論による予測 4. 制御則の多目的評価 5. アクションガイダンス 6. デッドロックの解析 7. インタロックの回避

	計画問題	設計問題
定義	計画問題とは、与えられた目標を達成するために利用可能な資源（人、金、物）の割り当てを時系列として決定することである。	システムの入出力の要求仕様が与えられたとき、これを実現するために、構成要素を組合せ、かつ、各構成要素の内部仕様を決定すること。
特徴	1. 探索空間が非常に広い 2. 評価属性が多様である 3. 環境予測が不確実である 4. 部分計画間の相互作用 5. 既存計画の再利用 6. 費用と利益のトレードオフ 7. 対話型計画作成支援の要求	1. システム構造の解空間 2. 解析による合成 3. 構成要素の最適化 4. 検証の完全性 5. 対話形式での支援 6. 段階に応じた支援形態
基本タスク	1. 計画過程の階層化 2. 組み合せ的探索 3. 制約条件の相互作用 4. 環境の整理 5. 曖昧さの処理 6. 不完全性の処理 7. 解釈の多様性	1. 構成要素とその関連の表現 2. 階層的表現 3. 代替え案の自動生成 4. 部分システムの評価 5. 上位レベルの知的後戻り 6. 事例の検索と利用 7. 並列的問題解決
問題解決機能	1. トップダウン精密化 2. 拘束最小化 3. 網羅的探索 4. 仮説推論 5. 類推 6. 計画の多目的評価 7. リスク下での意思決定	1. トップトップダウン精密化 2. 拘束最小化 3. 不確実性推論 4. 仮説推論 5. 仮説推論による知的検索 6. 類推による知的検索 7. 協調型推論

（注意）上段の表は、解析問題、下段は合成問題に属する。(知識システムハンドブック [Kobayashi 1990])

1.3.2 問題の記述性と解決アプローチ

多くの解決すべき対象問題は、何らかのプログラミング言語にて、解決手続きを記述することが可能な問題と、事実やルールなど宣言的知識とそれらを動的に組み合わせて解を探索する推論機構部によって、解決することが可能な問題に分けて考えることができる。前者を、手続き型アプローチということとし、後者を、ルールベース型アプローチということとする。これらのアプローチのメリット／デメリットを以下に簡単に整理する。

（１）手続き型アプローチ

手続き記述型アプローチでは、システム全体の流れを容易に追えること、処理が高速であることなどの利点がある。しかし、その反面、知識が制御構造の中に渾然一体となって分散しているため、どこでどどの知識が用いられているか分かり難くなりがちである。同時に、１本の糸をたどるような幅の狭い問題解決システムとなってしまう可能性も大いにある。

（２）ルールベース型アプローチ

ルールベース型アプローチに基づくシステム（以下、ルールベースシステム）は、一般に、知識を独立したルールの形で知識ベース内にデータとして格納しているため、システム内に埋め込まれた知識を明確に認識できる。また、予め、幅広い問題空間を対象としている。ただし、一般に、システムがどのように、動作するかは実際にシステムを実行してみなければ殆ど予想がつかないこと、実行時間が長いこと、また、大規模なシステムは、デバッグの手間が掛かることなどの欠点がある。

図 3.1 は、解決手続きの記述性からみた問題の分類と各アプローチの一般的な適用の仕方を示している。実際、解決手続きを記述することが可能な問題は、ルールベース型アプローチでもその問題解決システムを構築することができるが、通常、解決効率の面で手続き記述型アプローチを用いる。

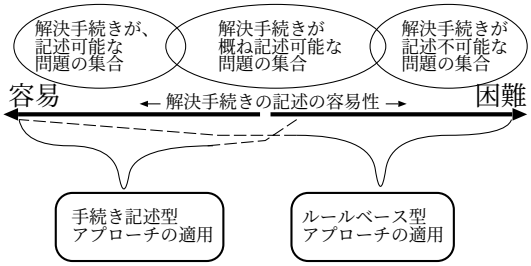


図 1.3: 問題の性質と解決アプローチ

1.4　従来の問題解決システム

一般に、問題解決システムは、ルールベースに基づく問題解決システムが大部分を占める。特に、中心的な存在はエキスパートシステムであり、チェスなどのゲームシステムは、プリミティブで少数の知識（オペレータ、評価関数）を用いながら、探索を実行していくシステムであるが、これも、一種のルールベースシステムと考えることができる。以下、これまでの問題解決システムを振り返る。

GPS

Newell らは、問題解決の過程を状態空間分析という考え方で説明しようとした。この考えは初期状態から目標状態へ至るすべての状態空間（問題解決空間）を想定し、状態から状態への移行手続きはオペレータの適用によって行なう。この研究の中で作られた、GPS(General Problem Solver) [Newell,Shaw,Simon 1963] は、状態記述 S と目標 G が与えられると、問題解決に有効な前向き推論規則を同定する。この同定プロセスは、まず、S と G の差 (difference) を計算する。この差の計算は、各ドメインに固有の関数である。この差をもとに、予め用意した、差と適用規則のテーブルから有効な規則を選択（手段ー目的分析）し、適用し、最終的に問題を解決する。本システムは、人間の問題解決過程に関する情報処理モデルであり、以後の多くの研究と類似点が多い。

STRIPS

STRIPS[Fikes 1971] は初期の代表的なロボット問題解決システムである。本システムは解決目標のスタックを持っており、スタックの一番上の目標が複合目標であったり、現段階で解決できないものであれば、要素目標を順次積んでいく。GPS に影響を受けた研究であり、スタイルは異なるが、考え方は近い。

エキスパートシステム

1970 年以降、人間の思考メカニズムをモデルとした、プロダクションシステムが出現し、これをツールとしたエキスパートシステムが注目されるようになり、現実的な問題、例えば、質量分析や核磁気共鳴の化学実験データから有機化合物の構造式を推定するエキスパートシステム DENDRAL[Buchanan 1978] や血液感染症の診断と治療に関する助言を与えるシステム MYCIN[Shortliffe 1976] など実用を目指したシステムの開発が試みられた。その他、代表的なものとして以下のようなシステムがある。

- ACE
  数百の OPS4[Forgy 1979] のルールと約５０の Lisp 関数で記述され、４０万回線分の CRAS データを VAX11/780 のＣＰＵ時間１時間で処理可能。

- R1/XCON
  R1/XCON [McDermott 1982a][Bachant and McDermott 1984] は、計算機システムの構成配置を行うシステムであり、OPS5[Forgy 1981] で、3300 ルールと 5500 の計算機構成要素の記述からなる。

- XSEL
  XSEL[McDermott 1982b] は、計算機システム販売支援システムで、OPS5 にて記述されていた。

- YES/MVS
  YES/MVS[Griesmer et al. 1984] は、オペレーティングシステムの支援システムであり、OPS5 の言語仕様、処理系を改修している。

- DAA
  DAA[Kowalski and Thomas,1983] は、集積回路の設計支援システムであり、OPS5 を用いて、130 ルールで記述されて おり、VAX11/750 の CPU 時間で MCS6502 マイクロコンピュータを専門家が満足するレベルで設計できた。

SOAR

SOAR プロジェクト [Soar 1994] は、1983 年 CMU の Alien Newell を中心に、知能一般に対するアーキテクチャの実現を目指して始まった試みである。SOAR の問題解決プロセスの概要は、

- タスクを達成するための問題空間の候補を提案し、その中から１つを選択する
- 選択された問題空間の初期状態の候補を提案し、その中から１つを選択する。
- 現在の状態が目標状態ではない間、以下のプロセスを繰り返す。
  - 現在の状態に適するオペレータの候補を提案し、その中から１つを選択する。
  - 現在の状態にオペレータを適用し新しい状態を生成する。

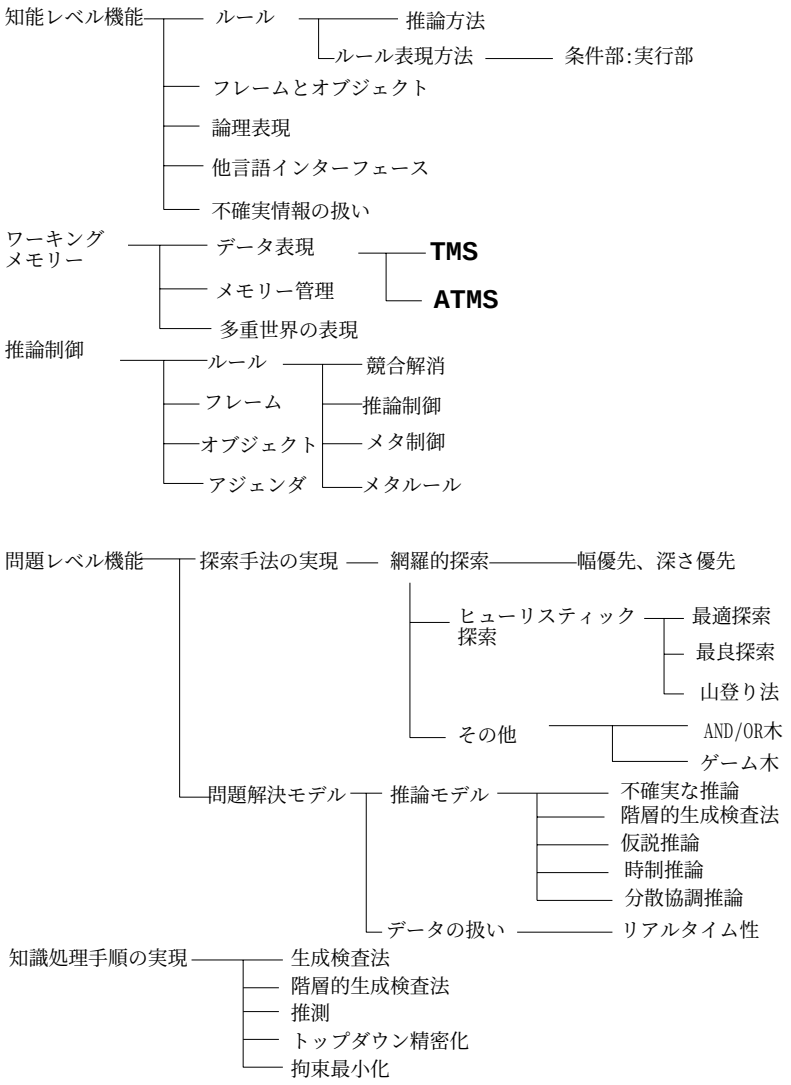
この考え方は、今後の問題解決システムの標準となっていくものと期待されているが、現実問題への応用については高い成功を収めたという報告はまだない。

ゲームシステム

特に、コンピュータチェスに関しては、計算機の性能向上と探索技術（静的評価関数、ミニマックス法及び$\alpha - \beta$法、反復深化、キラーヒューリスティック）の向上、戦略（定跡データベース、終盤データベース）の充実により、現在では、人間を凌ぐ実力となっている [H.Iida et al. 1994][Matsubara 1996][Matsubara 1994]。

1.5 問題解決システムの構成要素

前項にて示した例は、一例であるが、基本的にシステムが持つ基本構造は、図 1.4のようなものである。問題解決エンジンの中心的機能は探索である。



（知識システムハンドブック [Kobayashi 1990] より）

図 1.4: 問題解決システムの構成要素

1.6 従来の問題解決システムの問題点

前節までに述べた、これまでのルールベースシステムに共通に言える問題点は、以下のとおりである。

- ルール数の増大に伴い、ルールの参照回数、あるいは、探索木の急激な増大を招き、従って処理時間の指数関数的増大を引き起こす。このような計算量の壁に対し、いくつかの打開案が提案されている。たとえば、エキスパートシステムにおける高速化手法 [Ishida 1991] は、やSOARのような問題空間分割および探索、あるいは、多くのシステムで行なわれているように分散化、並列化などがあるが、個々に成果をあげているものの根本的な障害の解決にはなっていない。

- 適切なルールの獲得または構成、探索木の状態に対する適切な状態評価関数の構築は、一般に困難である。

- 多くのルールを使用するような、診断などのエキスパートシステムでは、必要な知識を完全に網羅し解決率を向上させることと、不要な知識を排除し処理時間を短縮することに膨大なデバッグ時間を要する。この傾向は、言うまでもなく、システムの規模が大きくなると、一層大きくなる。結果的に、システム開発コストに影響し、社会に受け入れられないこととなってしまう。

1.7 社会の動向

今後、社会のニーズとしての解決対象は、複雑化、大規模化の傾向にあると同時に、実時間性が今まで以上に要求される問題が多くなる。

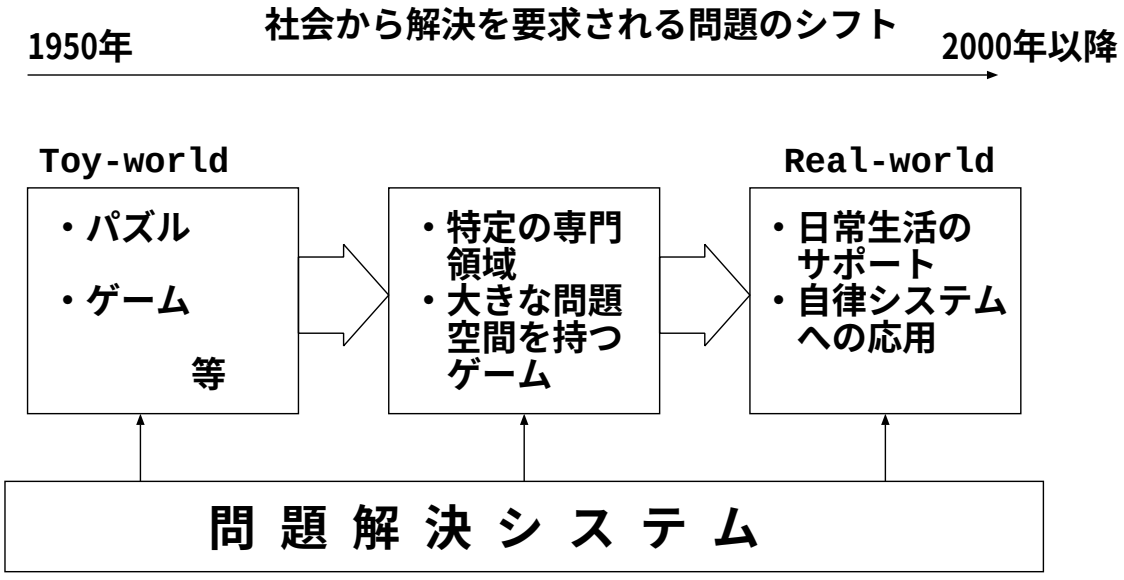


図 1.5: 解決対象問題の変化

従って、これまでのルールベースシステムの方式だけでは、知識表現、計算量、実時間性、要求される解の精度の点で問題が多い。

そのような状況において、研究者たちは、図 1.6に示すように、より現実的な問題解決に向けて、ハードウェアの性能向上、あるいは、並列化、ソフトウェアの改良などに取り組んで来たが、これらのことだけでは、対応が困難であることが分かって来た。

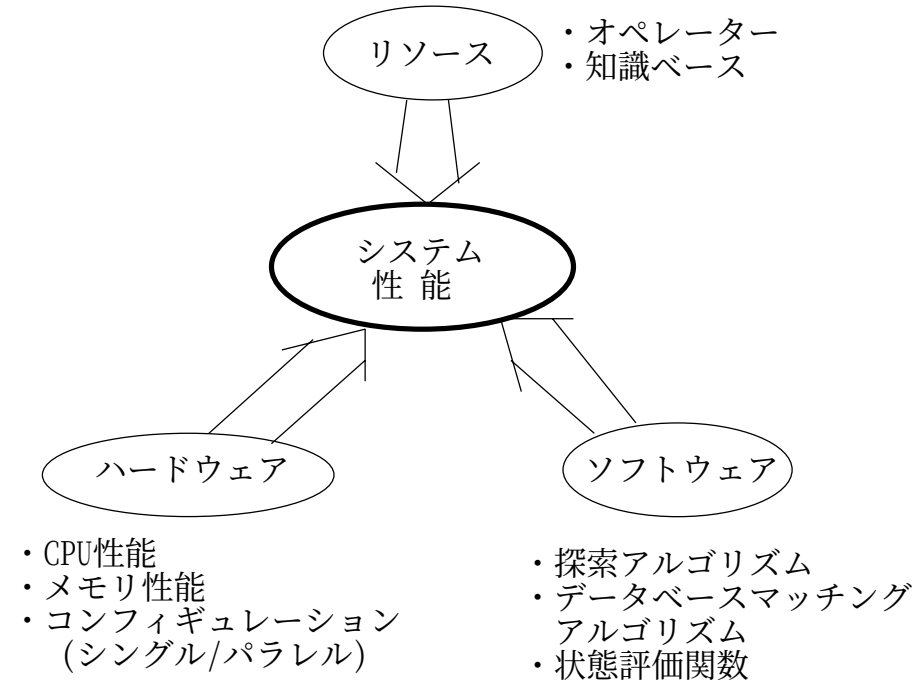


図 1.6: システム性能

1.8 研究の背景

1.8.1 本研究の背景

人間は、コンピュータのような単純な計算、データ照合、データ生成などを高速に実施することは困難である。しかしながら、日常生活を支障無く送ることができる。たとえば、極端な例かもしれないが、携帯コンピュータにエキスパートシステムを構築し、毎日の通勤や通学の際に、駅のホームにて、来た電車に乗るか否かを聞いたとする。天気や夕飯の予定や吊り広告の内容,... など、いろいろな情報を考慮した結果、５分後に “乗りなさい” という出力が出てもそこには、乗るべき電車は存在しない。従来のルールベースでは、同様のことが起こる可能性を有する。一方、人間の場合は、この電車に乗るという状況（文脈）における、局所的な問題解決を行ない瞬時に答えを出力する。

瞬時の意思決定が要求される場面において、その状況に対する判断のための洗練された

選択肢とその評価基準がある程度体系化された知識としてもっているように観察される。また、特定の法則については、多少、状況が変わっても応用が利くよう抽象化して知識構造の中に組み込まれているように見える。これは、経験的に得た、あるいは、理解によって構築した、問題解決過程、解決の方向性、利用可能な手段などといった戦略知識を巧みに利用しているためであろう。これは、物事の因果関係 [Bunge 1956][McCarthy 1969] などに着目することで、巧みにフレーム問題を回避していることに相当しているといえる [Matsubara et al. 1993] [Matsubara et al. 1990]。

そこで、問題解決システムを構築する際においても、何らかの方法で大まかな問題解決のストーリーを表現し、個々の解決知識をその体系化された問題解決ストーリーに適合した形でシステムに組み込むことができないかという視点に立って、検討を行うことは意義がある。

1.8.2 近年の研究の動向

近年、人間の認知的プロセスにウェートを置いたシステムの考え方、または、定性推論 [Kuipers 1984][Kuipers 1986] などで、既に行なわれているような、物理法則などを、ルールに含めることが重要とされるようになってきた。一方、マンマシン・インターフェースの分野で、人間を中心にしたシステムモデリング MFM(Multilevel Flow Modelling)[Lind 1990][Lind 1994] が注目されており、原子力プラントなどで、ヒューマンエラーの低減などに有効と考えられている [Chandrasekaran 1993][Fukutomi 1992][Ohga 1995] [Niwa 1996]。近年の航空機技術でも人間中心の自動化の重要性が指摘されている [JAL 1995][Sakuma 1996]。さらに、より人間の認知プロセスにウェートを置いた研究領域として、ヒューマンモデリング [Furuta 1998] が注目を集めている。知識の再利用の観点では、推論などの形式的な操作の高度化だけではなく、様々な形態で存在する知識の内容を扱う基礎研究と高度技術が不可欠であることが明らかとなってきた。そのような問題の基礎研究としてオントロジー工学 [Mizoguchi 1997] が注目されている。

これらの観点は、１つのルールが広範囲の対象に適用できる点にあり、ドメインに対し、強力な制限を与えたり、逆に、適用範囲を大幅に広げるといった効果が見込める。従って、先にあげた従来技術の問題点の打開に期待が寄せられている。しかし、前項で述べたような視点のでの検討はこれまで行なわれて来なかった。

1.8.3 本論文における研究の立場

本論文における議論も、近年の研究の立場に近い視点を有し、人間の持つ問題解決戦略を有効に活用することに着目するものである。

これまでの問題解決システムの議論では、両極端のアプローチの対比における議論が多かった。特に、エキスパートシステムに関する議論のみが注目されてきた。そこで、本研

究では、図 1.7の両極端の中間に位置する方法があるのではなかろうかと考え、人間の、特定の問題解決の側面について焦点を絞り検討をおこない、その結果に基づいて問題解決システムの構築とその方法論について検討する。

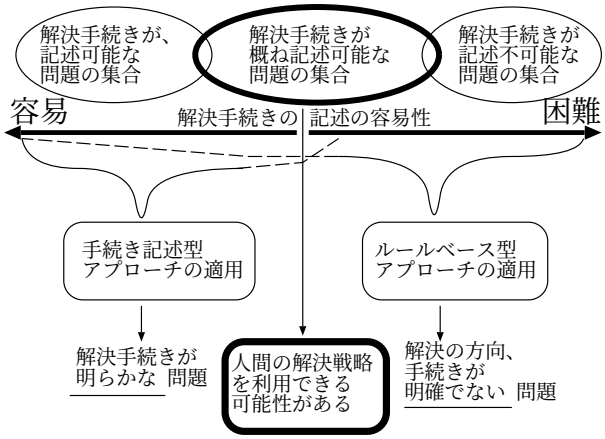


図 1.7: 本研究の対象領域

1.9 研究の目的と効果

本論文は、先の各方面での “人間を中心に置いた考え方” あるいは “内容指向” といった動向に着目し、図 1.8のように問題解決の分野に人間的解決過程を積極的に採り入れるアプローチを提案する。

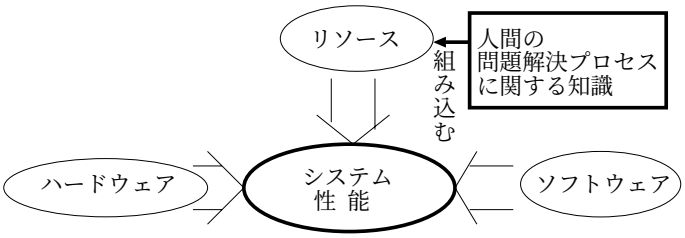


図 1.8: 新たなアプローチ

1.9.1 本研究の目的

本研究の目的は、人間の問題解決について考察し、仮説をたて、人間の持つ解決戦略を有効に活用するための問題解決方式を提案すること、および、その設計法を確立することを目指す。利用可能な解決プロセスを効果的に表現し、その表現されたプロセスに沿って

問題を解決してゆくことで、最適ではなくても、人間にとって満足できる解を確実に早く求めることができるシステムについて検討する。また、そのための条件を整理し、システム構築に役立てることとする。

1.9.2 研究対象となるシステムの特徴

本研究で検討の対象となるシステムは、大規模なデータベースや強力な探索ツールの開発ではなく、どちらかといえば、問題解決過程を特定の制約条件を満足する状態空間の系列と考え、その空間を順次移動しながら問題解決することを考える。

本研究は、人間が経験や学習によって得た問題解決のための戦略（問題分割、解決ストーリー、個々の問題解決のための効果的オペレータ、状況に対する評価能力等）自体が持つ問題解決力が極めて高いのでないかという前提に立ち、問題解決エンジン自体は、山登り法 [Athans 1974] といったシンプルな機能の利用を前提とする。一方、オペレータについても、プリミティブなものではなく高度なマクロオペレータが使われることを前提にしており、オペレータの粒度が粗くなることの弊害を吸収するために、部分問題空間の接合は、図 1.9の (a) のような１点ではなく、(b) のような空間結合を前提とするものとする。これらのメリット、デメリットは、表 1.2のとおりである。

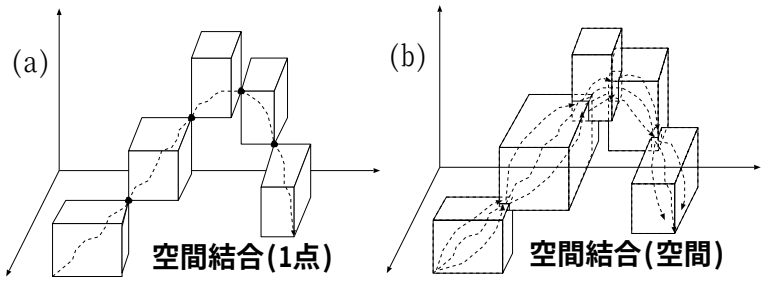


図 1.9: 部分問題空間の結合

表 1.2: 研究対象システムの特徴とそのメリット／デメリット

項目	メリット	デメリット
1. 部分問題の空間結合	1. 問題分割が容易 2. 環境変化に強い 3. サブゴールへの達成が容易	個々の部分問題の定義が曖昧
2. シンプルな評価関数	1. 計算が容易かつ高速 2. 関数を作るのが容易	解決できる問題に制約を与える

特に、項目１については、今後、応用的場面にて、様々なツール（各種探索機能、学習機能、意思決定手法など）の適用を考えることとし、本提案方式を基本形とする。さて、ここで、本提案方式の期待される効果を再度整理する。本手法は、結果的に問題が分割され、個々の部分問題について問題を定義し直すという手間が生じるが、その反面、適切な定義が可能ならば問題解決の計算量、解の最適性の観点で従来システムに比べ、原理的には、桁違いに問題解決性能が向上すると考えられる。

1.9.3 本研究効果の応用可能性

本研究は、従来データベースや探索だけでは解決が難しい問題や、最適解でなくても実時間性を確保したい問題、可解性を確保したい問題に対し有効であると考えられる。従来の問題解決における問題の難易度と本手法の応用の可能性を、表 1.3に示す。ルールやデータに信頼性があり、人間の専門家が解決の方向性を全般的、部分的に示すことができる問題であれば、比較的容易にシステム構築ができる可能性がある。

表 1.3: 問題の難易度と応用の可能性

難易度	内容	従来方式	提案方式
易	・小さな探索空間 ・信頼性の高いデータと知識	・全数探索 ・単調な推論 ・一通りの推論	・所定の条件を満たす部分問題を構成できれば応用可能
やや易	・信頼のおけないデータと知識	・多量のソースから証拠の結合 ・確率モデル ・ファジーモデル ・正確なモデル	
	・時間変化データ	・状態でトリガーされる期待	・所定の条件を満たす部分問題を構成できれば応用可能
	・巨大探索空間	・階層的生成テスト	・所定の条件を満たす部分問題を構成できれば応用可能
標準	・評価法のない部分解	・固定した順序の抽出の段階	
	・固定シーケンスを許さない領域	・抽象探索空間	・対象領域に存在するシーケンスが定義できれば応用可能
	・相互作用する部分問題	・拘束の伝播 ・最小の拘束	・相互作用関係とウェートが分かれば応用可能
	・推測の必要性	・信頼できる推論 ・依存関係に基づく推論	
やや難	・弱い単一推論のモデル	・複数の推論方式	・１つの推論方式として所定の条件を満たす部分問題を構成できれば応用可能
	・多様な専門家	・異種のモデル ・環境駆動型スケジューリング幅可変探索	・１つのモデルとして部分的に応用可能
難	・効率的でない知識ベース	・データ構造 ・コンパイレーション ・認識節約	

1.10 本研究における重点検討項目

以降議論を進めていく上で、特に、着目すべき項目は以下のとおりである。

1. 人間の持つ解決戦略はどの程度の解決能力を有するか。
2. 人間の持つ解決戦略を用いる問題解決手法が従来の枠組の延長線上において定式化可能か。
3. 人間の持つ解決戦略は獲得可能か、そして、計算機上に実装可能か。
4. 人間の持つ解決戦略を用いる問題解決システムは、十分な解決能力を有するか。

２章以降、順を追って検討していく。

1.11 本論文の構成

本論文は２章以降、以下のような構成となっている。

- ２章では、人間の問題解決について考察し、仮説を立てる。
- ３章では、提案する問題解決方式を定式化する。
- ４章では、提案方式を、更にシステムの形で提案し、システム設計法を提案する。
- ５章では、３種類の例題を用いて有効性を確認する。各例題に使用する必要データ量や処理時間などを確認し現実問題への応用の可能性を分析する。
  - 基礎例題１として N^2-1 パズルについて問題解決システムを構築した。性能を比較するために、ルールベースシステムの解決結果と比較を行なう。
  - 基礎例題２として、追跡ゲームについて問題解決システムを構築した。性能を比較するために、ルールベースシステムの解決結果と比較を行なう。
  - 現実的例題として、航空機操縦システムを構築した。性能を比較するために、ルールベースシステムの解決結果と比較を行なう。
- ６章では、以上の実験結果に対するディスカッションを行なう。
- ７章では、以上のシステム構築法及び例題実験を総合的に評価し、その有効性について結論を述べる。

以上の説明の流れ図を 1.10に示す。

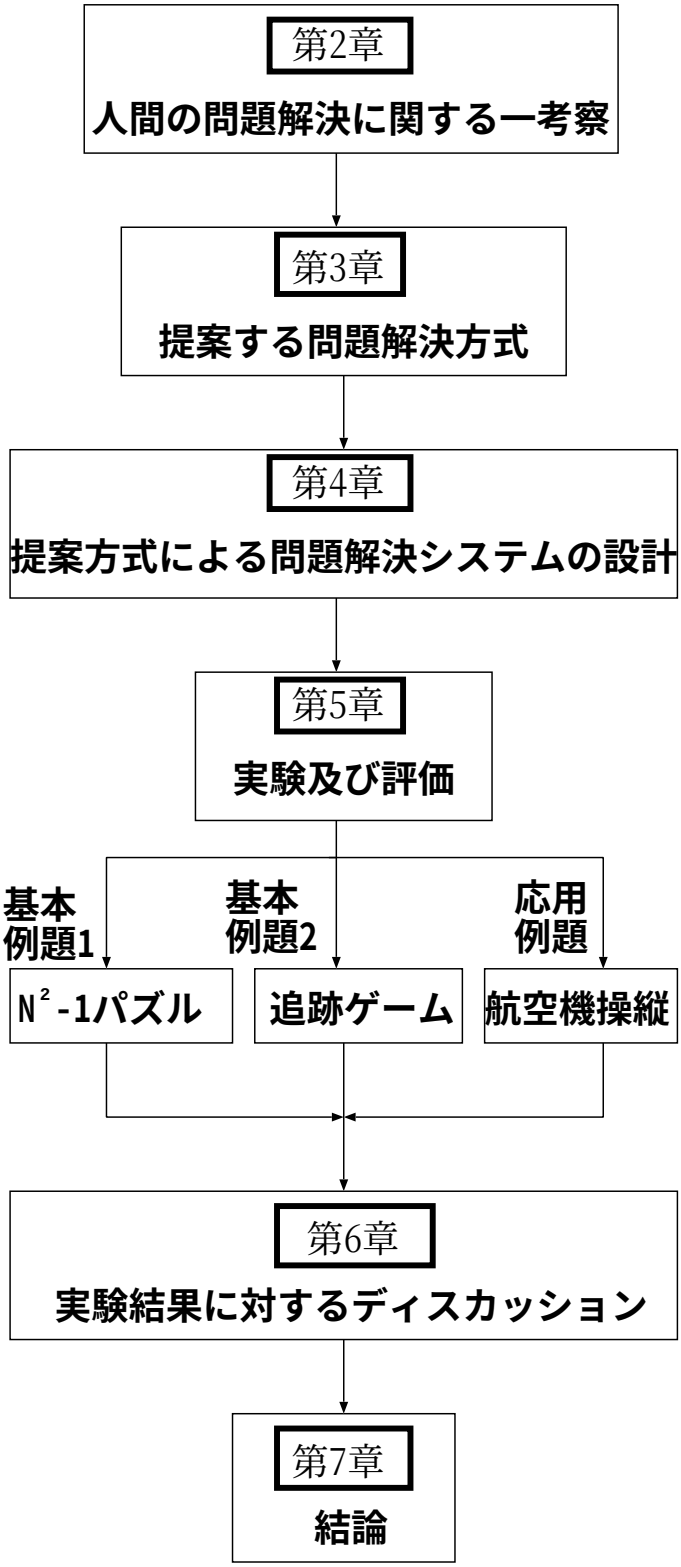

第2章
人間の問題解決に関する一考察
第3章
提案する問題解決方式
第4章
提案方式による問題解決システムの設計
第5章
実験及び評価
基本
例題1
基本
例題2
応用
例題
N²-1パズル
追跡ゲーム
航空機操縦
第6章
実験結果に対するディスカッション
第7章
結論

図 1.10: 論文構成

第 2 章

人間の問題解決に関する一考察

2.1 はじめに

問題解決システムの対象問題の解は１つとは限らない。現実の多くの問題には複数の解が存在する。得られた解が最適解であることは望ましいことであるが、すべての問題において、最適解が要求されるわけではない。

一般に、ルールベースシステムでは、最適解を求めようとすると、計算時間が指数関数的に増大する。したがって、人間が要求する範囲の解が見つかったところで、解を出力すれば良いと考えることも可能であるが、どちらにしても広大な問題空間の中の解を探すのは決して簡単なことではない。

一方、人間は、多くの場合、ルールベースシステムと同等かそれ以上の答えを容易に見つけることができる。この、人間は、あらかじめ、問題解決の大まかな流れを体系的にもっており、また、その流れの中の各段階で使う解決ルールも大まかに決まっていて、悩むことなくそれを使っているのではないかと感じることがよくある。

本章では、このような人間による問題解決とルールベースシステムによる問題解決の比較事例をあげ、考察する。

2.2 使用する用語の定義

以後の議論のために、予め、いくつかの用語を定義しておく。

2.2.1 問題解決のコストと最適性

アルゴリズムの理論の分野では、計算コストは計算やデータ処理時間を決定する要素を意味する。一方、問題解決のコストは、解決対象への、ルールを適用する際の金銭的コストやエネルギー消費量などの様々なコストが含まれる。従って、この場合、「最適解」と

いうものは、一概に、最短ステップの適用ルールの系列とは限らず、ゴールに至るまでに適用するルール（オペレータ）の適用コストの和となる。具体的なコストの評価方法は問題によって異なる。

2.2.2 解の質

最適解に近い解ほど、「解の質が高い」あるいは「質の高い解」と呼ぶことにする。本論文においては、この解の質という概念を、以降、頻繁に用いる。

2.2.3 解決戦略

本論文において使用する解決戦略は、以下のとおりである。

- 問題解決のストーリー（漠然とした解決の流れから、明確なアルゴリズムまで対象）
- ルール（または、オペレータ）
- 状態評価関数（または、状態評価基準）
- 個々の問題解決ストーリー、ルール、評価関数等の優先順位

2.2.4 エキスパート

本来の定義よりも広い意味で用いる。通常、最適性の高い解を見い出すことのできる少数の人間（「真のエキスパート」または「真の専門家」ということとする）を意味するが、本論文では、多少無駄な手続きが入ったとしても確実に解決できる人であればエキスパートと呼ぶこととする。

2.3 人間の問題解決の事例

ここでは、ある問題に対し、その真の専門家が解決する様子を観察するのではなく、普通の人間が直感的に思い付くような知識が問題解決にどの程度有効かということについて事例をあげる。

本節では、２つの具体的な事例を上げる。ここで示すのは、心理学や認知科学の立場の実験ではなく、人間が、簡単な問題に対してどのような戦略で取り組むのかを確認したものである。本実験の被験者には１４人の小学生（高学年）をランダムに選び、簡単なパズルとゲームを解決してもらった。このように、小学生を被験者に選んだ理由は、思考が柔軟であること、過去の経験などから特定の思い込みで問題解決をする可能性が少ないことなどの理由である。

14 人中、10 人の被験者は時々ゲーム遊びをするとのことであったが、特に、ゲームやパズルが得意な人ばかりではない。また、前もって、実験にたいする予備知識は与えず、実験会場にて始めて説明を行った。

2.3.1 １５パズル

（１）実験内容および状況

１５パズルは、４マス×４マスのボード上に、１から１５までの数字の書かれた１５枚のタイルを、１箇所のブランクをワーキングスペースとしてゴール配置に変えていくパズルである。

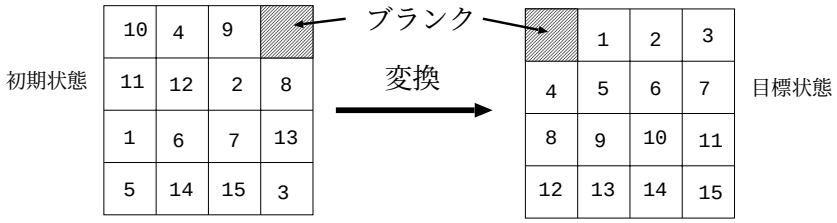


図 2.1: １５パズルの例

被験者の中で 15 パズルを、今までに経験したことがある人は、半数であった。しかし、最初のリハーサルの問題が出来た人は３人であった。また、しばらく取り組んでもらったところ、１時間以内に半数の被験者ができるようになった。

（２）人間の戦略

１５パズルを解くことができるようになった被験者に問題解決のアプローチをどのように行っているのか聞き取り調査を行ったところ、以下のとおりであった。

- 戦略１： 下段から順番にそろえる。（３人）
- 戦略２： 上段から順番にそろえる。（３人）但し、ブランクには隣接タイルをダミーとしていれる。
- 戦略３： パズルを良く見て、上下左右揃えやすいところから揃える。（１人）

被験者の意見を総括してみると、戦略 1 及び 2 と答えた人は、始めから解決のためのサブゴールの系列を決めておくということであった。つまり、自分の決めたサブゴールのみに集中し、たとえ、他にそろえやすい部分があったとしても見向きもしないのである。また、戦略 3 と答えた人は、熟考型とでもいうか、状況によって、サブゴールを変更する

戦略である。しかし、基本的にサブゴールの選択肢を決めておいて、状況次第で選び方が変わるだけなので、前者の応用と考えられる。

（３）実験結果

実際に、被験者の代表３人に 10 題の例題を解いてもった。15 パズルを解くことに対して、3 人のうち 2 人は、今回が初めてであった。彼らの解決状況は、ビデオカメラにて収録し、あとで、移動回数をカウントし解決過程を更に確認した。

最初に揃った、パズル周囲のどこか１辺を、第一サブゴールということにする。

実験の結果、人間の場合、この第一サブゴールに到達するのが、全解決ステップに対して、平均で、スタートから約 40 %の段階という早い時期に成立し、以後、基本的に、それを崩さないで解決を進めていることが分かった。

また、たとえば、戦略１の被験者は、第一サブゴール到達後、次に下から 2 段目の解決を目指し、それが、解決後、やはり、それを壊さずに以降の解決を進めるという方法をとった。その他の戦略でも同様であった。

（４）ルールベースシステムとの比較

図 2.3に、ルールベースシステムによる解決結果との比較結果を示す。ここでの、ルールベースシステムは、問題空間全体を対象とするが、全探索では、時間がかかり過ぎるため、実時間探索を用い、適当な深さで探索を打ち切りながら、逐次最良優先にて解決進める方式である。用いたマシンは 133MHz のノート型パーソナルコンピュータを使用した。処理概要は、図 2.2のとおりである。

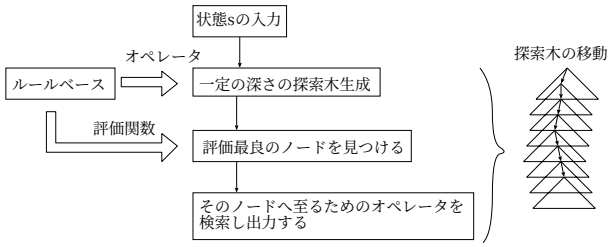
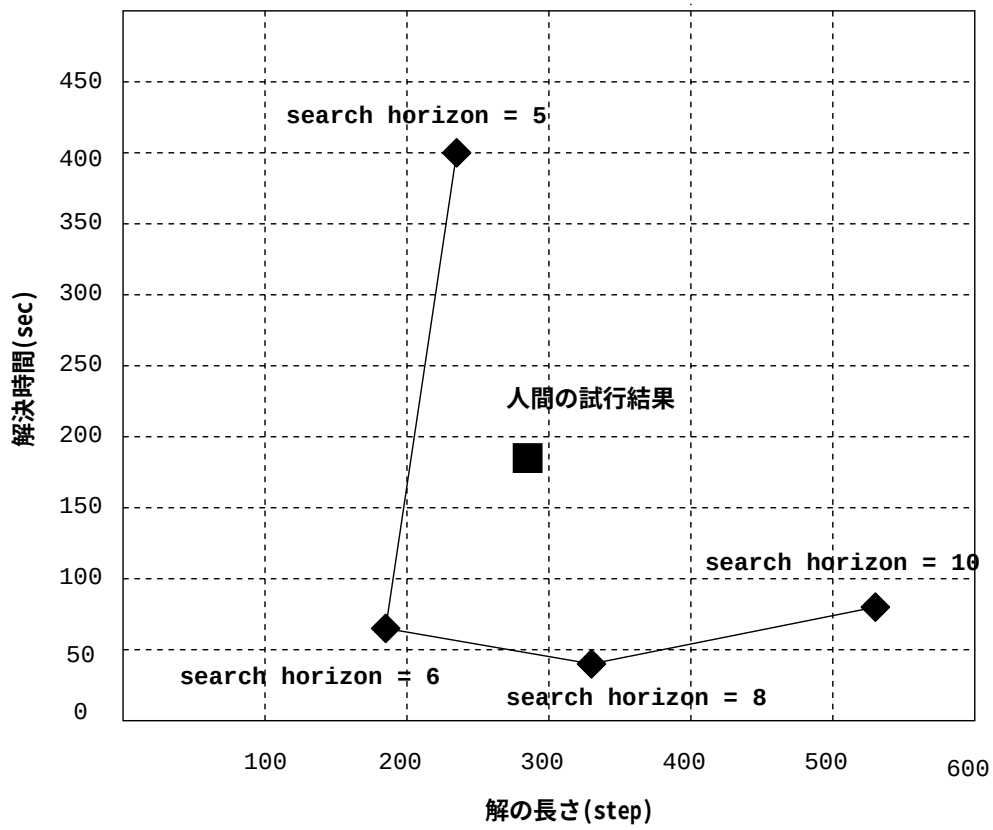


図 2.2: ルールベースシステムの処理概要

本ルールベースシステムでは、search horizon（先読み深さ）が浅いうちは、人間のような第一サブゴールは現れにくく、サブゴール達成後もすぐにそれをすぐに壊してしまう傾向が強い。徐々に深くしていき、search horizon が 10 程度で、人間の傾向に近付いた。これは、逆に、このパズルに、関して、人間の採った戦略自体が始めから、強力な問題解決能力を持っていたことを示す 1 つの例と言える。



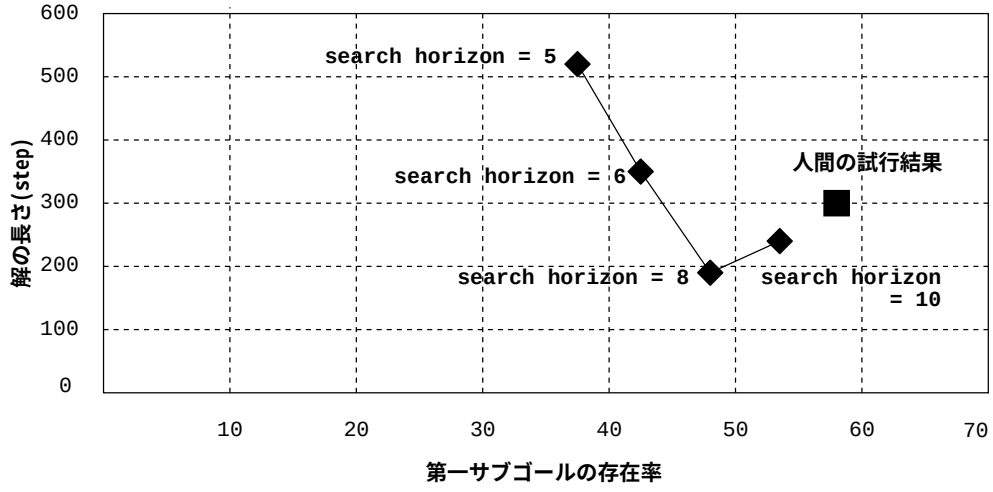


図 2.3: １５パズルの結果

2.3.2　追跡ゲーム

（１）実験内容および状況

第２の問題は、８マス×８マスの盤面を用いたゲームで、４つの追跡エージェントが１つの逃亡エージェントを追跡し追いつめる問題である。この際、各エージェントの１ step 当たりの動きは、周囲の８方位のうちの１マスとし、追跡エージェント側、逃亡エージェント側がそれぞれ交互に動かすものとする。追跡エージェントは、４つのエージェントすべてが移動可能であり、１から順に動かすか、４から順に動かすかを選択できるものとして、少なくともどれか１つは動かすものとする。ゴールは、追跡エージェントが逃亡エージェントを囲み動きを封じることである。被験者は、２つのグループに分かれて、それぞれ考えてもらった。

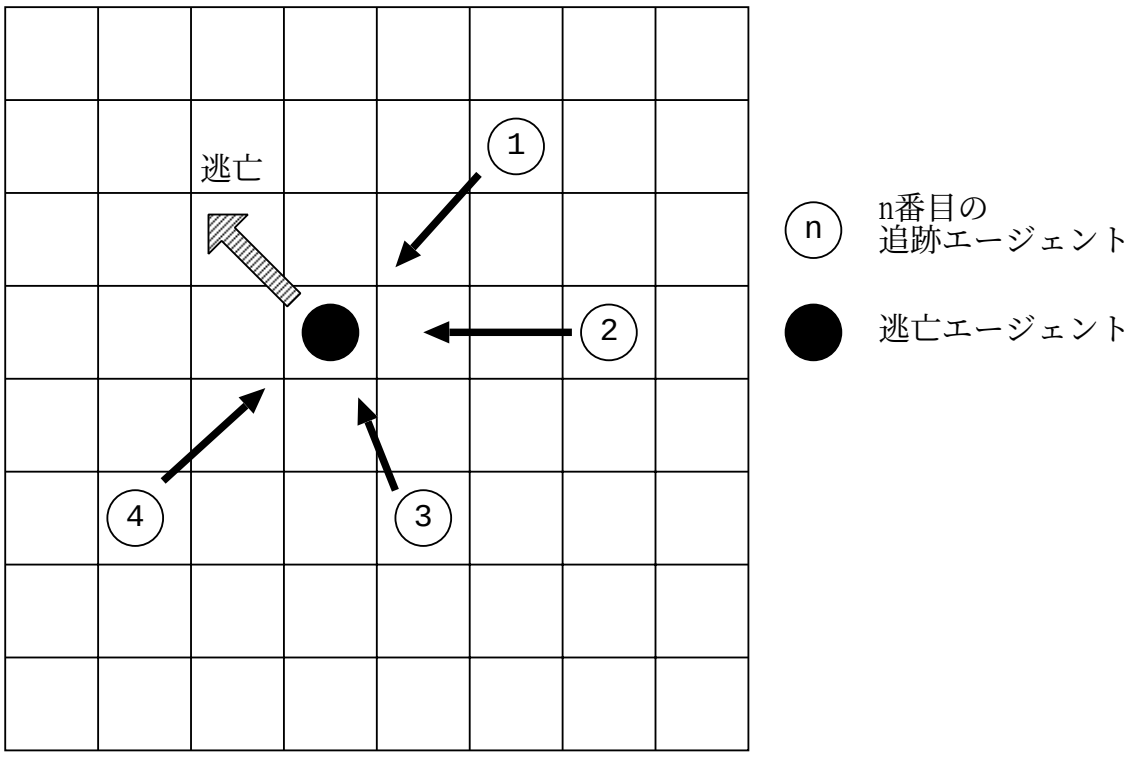


図 2.4: 追跡ゲームの例

（２）人間の戦略

被験者たちの取った戦略は、どちらのグループも同様のものであった。

- 逃亡エージェントの近くで、早く、Ｌ字形を作ること。
- Ｌ字形で徐々にコーナーに追い込むこと。

これは、大人であれば、容易に思い付く戦略であるように思われる。しかし、小学校の低学年の数人にも、考えてもらったが、追跡エージェントのフォーメーションを考慮せず追跡するため、いつまでも追いつめることができなかった。高学年では、チームプレイやフォーメーションなどを意識し、すぐにＬ字形を形成する戦略をとるようになった。

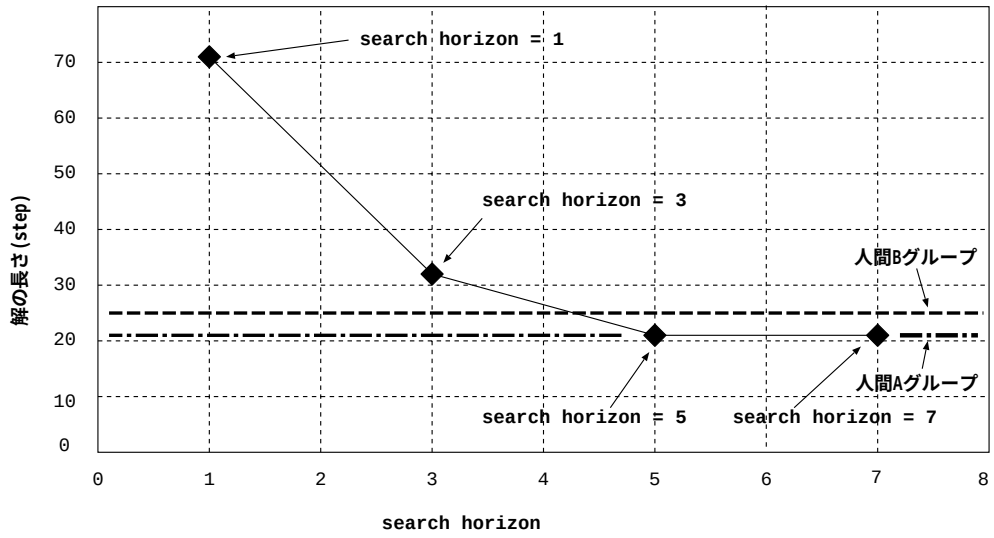
（３）実験結果

この結果、２つのグループは、それぞれ、２１ステップと２５ステップにて解決した。

（４）ルールベースシステムとの比較

本ルールベースシステムの構造は、先の 15 パズルにて用いたものと同じ構造であるが、大きな違いは逃亡エージェント側の機能と追跡エージェント側の機能を持ち、交互にエージェントを動かすことである。また、追跡エージェント側には、逃亡エージェントの評価関数を与えてあり、逃亡方向の予測が可能である。

次ページでは、この結果と実時間探索よる結果を比較したものを示す。本例のおいても、人間の極めて直感的な発想がある程度の深さの先読み深さをもつ実時間探索に匹敵することが分かった。



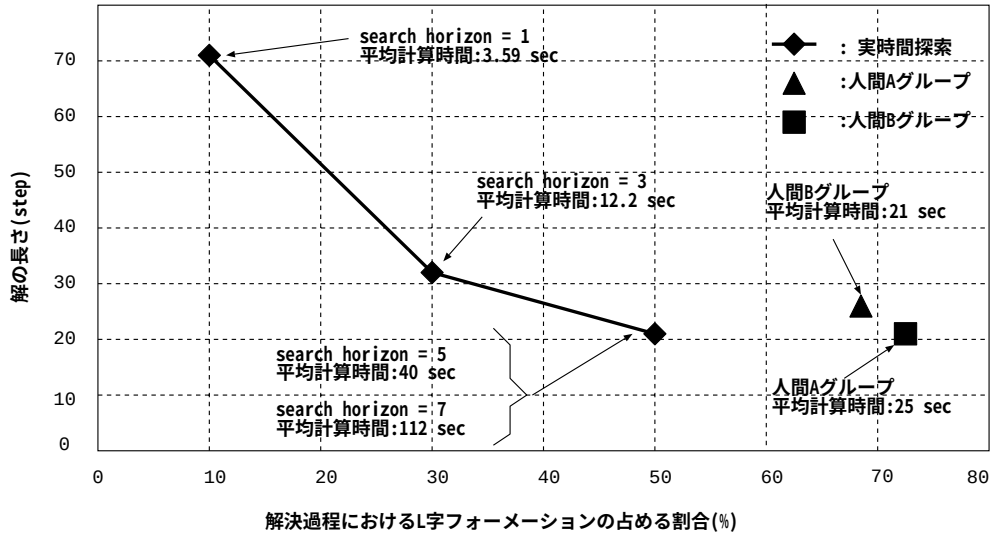


図 2.5: 追跡ゲームの結果

2.4 人間の問題解決に対する仮説

人間の問題解決能力には、以下の様な多様な側面が存在する。

- 未知の問題を試行錯誤しながら解く能力
- 経験などから解決戦略を学習し応用する能力
- 問題構造を類推し別の問題の解決戦略を応用する能力等

2.3 項では、15 パズルや追跡ゲームに対して、人間の採る解決戦略が問題解決に有効である事を示した。これは、解決戦略がある程度確立した問題に対する解決能力と考えられる。前章の事例では、解決者は、経験的に得られた解決戦略により、容易に、しかも良質の解を獲得することが出来た。表 2.1 の左欄は、被験者らの様子を観察及びインタビューにて解決の特徴をまとめたものである。右欄は仮説を示す。

表 2.1: 人間の問題解決に対する仮説

人間の問題解決の特徴	仮説
1. 解決者は、いくつかの中間目標を予め決めて問題解決を行なっていた。 2. 問題解決はいずれかの中間目標にて思考が途切れることなく進んでいた。	(人間は、ある程度、熟練した問題に対して) • 視点１： 問題解決ストーリーを、簡単な部分問題の系列として知識を構築し、順次解決しているのではないか。 • 視点２： 仮説１において、隣接する問題空間は１点ではなく空間で結合されていることにより、各部分問題の解決と隣接する部分問題への移行が容易なのではないか。
(個々のアクションの決定) 解決者のアクション決定サイクルが短時間であった。	• 視点３： 各状況において、使うアクションの候補が絞り込まれているのではないか。 • 視点４： 膨大な計算や深い探索思考を行なっていないのではないか。 • 視点５： アクションの決定には単純な判断を用いているのではないか。

2.5　まとめ

2.3 項では、実験により、人間の持つ問題解決解決戦略は、問題解決に対して、高い解決能力をもっていることが分かった。また、2.4 項では、今回の実験の結果を基に、人工知能の観点から、何故被験者らが効果的な問題解決が行なえたのかということについて仮説を立てた。そして、次章では、この仮説を用いて問題解決方式を提案する。

第 3 章

提案する問題解決方式

3.1 はじめに

２章では、人間が直感的に発想する問題解決戦略が問題解決に有効に機能する事例を示した。本章では、このような、人間の直感（実際には、経験的要素が大きいと考えられる） に基づく発想を計算機上で有効に利用するための問題解決方式を提案する。以下のステップにて、提案及び議論を進める。

1. 問題解決方式の提案
2. 問題解決方式の数学的定式化
3. 提案方式の解決可能性の検討
4. 制約条件を伴う問題解決の検討

3.2 問題解決方式の提案

提案する問題解決方式の枠組み

本項では、まず、前章にて実験した人間の持つ問題解決戦略の有効性に関する仮説を基に問題解決方式を提案する。　一般に、人工知能における問題解決の枠組みでは、Sを問題空間,Wをゴール空間、Rをオペレータ集合とすると、問題 Pは、$P =< S, W, R >$ の３項組で考えることができる。ここで、2.4 項の仮説を、この枠組みにて扱おうとする場合、評価関数 Hを加えて、

$$P =< S, W, R, H >$$

の枠組みにて考える必要がある。また、本提案において、S は、離散空間のみならず連続空間をも含むものとする。

以降、前章の仮説に基づき問題解決方式を提案する。ここで、提案する問題解決方式は、ある問題において、そのエキスパートが持っている問題解決ストーリーを部分問題の系列として抽出しそれを計算機上に知識表現することで問題解決を行う方式である。まず、前章にて示した仮説をもとに、従来の枠組みに順次重ね合わせていく。今、図 3.1のように、$k+1$ 個の部分問題にて構成される問題解決のストーリーがあるものとし、また、任意の部分問題を $P_i =< S_i, W_i, R_i, H_i >$ とする。

1. 仮説の視点１から、下図のように、ゴールに向かう部分問題の系列が存在する。
2. 仮説の視点２から、下図のように、各部分問題が空間にて結合されている。
3. 仮説の視点３から、オペレータは、各部分問題毎、それに適したオペレータが割り当てられているものとする。
4. 仮説の視点４、５から、オペレータの選択基準は、単純な計算、単純な評価基準を用いる。

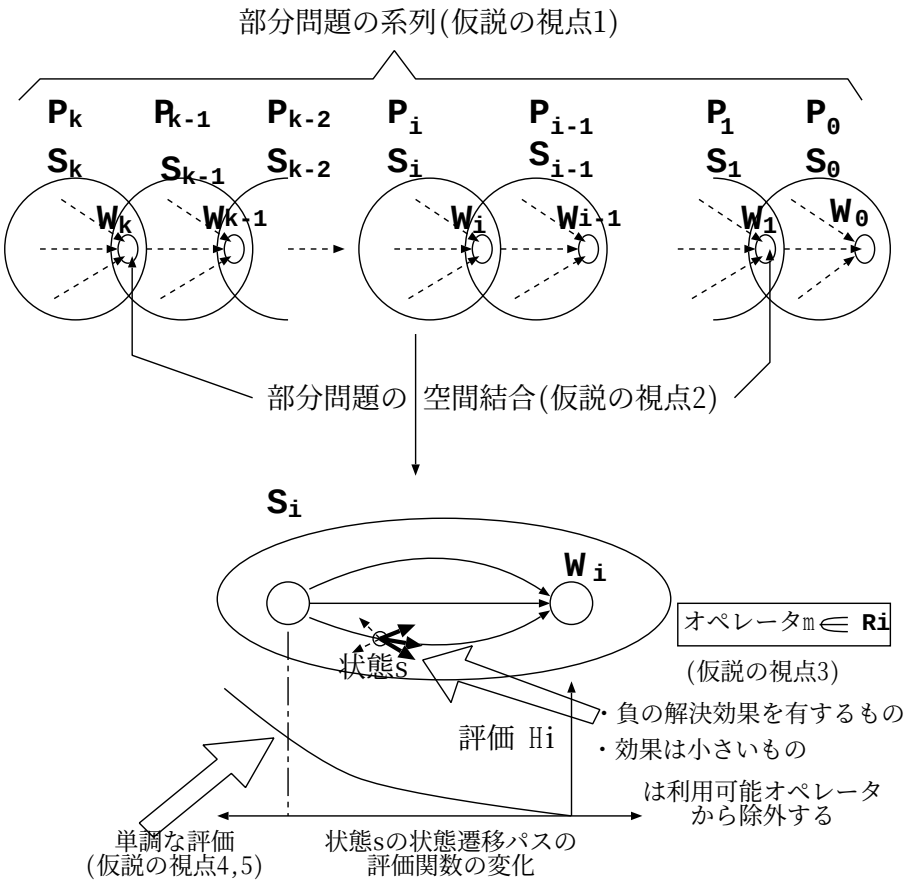


図 3.1: 問題解決モデル

ここで、W_0は、問題全体をとおしてのゴールである。なお、全部分問題は、以下の条件を満足していること。任意の問題Piにおいて、

$$(1 \leq \forall i \leq k)(\exists S_{i-1})(\exists S_i)(W_i \subseteq S_i \& W_i \subseteq S_{i-1})$$

図3.1は、前章の仮説を従来の問題解決の枠組みに反映したものである。ここで、さらに、各P_iをグラフの節点、また、隣接する部分問題同士(P_i, P_{i-1})をグラフのエッジと考えると、図3.1は、図3.2のように、単純な有効グラフとして表すことができる。

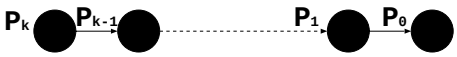


図 3.2: 部分問題のグラフ

この有効グラフのエッジの向きは、サブゴールW_iの存在方向である。よって、図refは、グラフの定義 $< V, E >$ を用いて、節点の集合 $V = P_k, .., P_0$、エッジの集合 $E = \{(P_k, P_{k-1})(P_{k-1}, P_{k-2}), ..., (P_1, P_0)\}$ であるので、

$$< \{P_k, .., P_0\}\{(P_k, P_{k-1})(P_{k-1}, P_{k-2}), ..., (P_1, P_0)\} >$$

と表すことができる。　このグラフは、問題解決の流れを表したものであり、また、各部分問題のインデックスは、問題解決のフェーズを表したものである。そこで、この直列に並んだ有向グラフをフェーズフローと呼ぶこととする。

一般的に問題解決の流れをモデル化するときに、１つのフェーズフローのみで表現できる問題よりも、複数のフェーズフローを用いて表現すべき問題の方が遥かに多いことは、容易に想像される。そこで、下図のような、複数のフェーズフロー$F_0, ...F_N$が有機的に結びついたものを、フェーズ木と呼ぶことにすることにする。

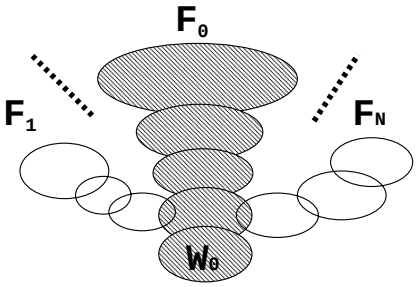


図 3.3: フェーズ木

次項にて、提案方式による問題解決方法を具体的に与える。

3.2.1 提案する方式の問題解決方法

計算機上で処理するためには、フェーズ木のままでは、扱い難い。そこで、フェーズ木を下図の様に展開したものを考え、処理上最多の部分問題からなるフェーズ・フローを基準に各フローの部分問題数(i)を決定する。問題解決の中心となるフェーズフローをメインフロー、その他のフローをサブフローということにする。ここで、メイン・フローとN個のサブ・フローから構成される問題を考える。全体として$N+1$個のフェーズ・フローが存在しメイン・フローF_0, サブ・フロー$F_1, ..., F_N$とし、個々のフェーズ・フローF_iは、部分問題$P_{i0}, ..., P_{in}$から構成されるものとする。また、予めフェーズ・フローに、ウェート$w_0, ..., w_N$を与えることで、オペレータが競合する場合の回避（ウェートの高い方を優先）するものとする。

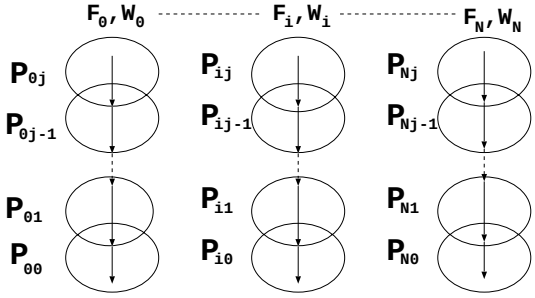


図 3.4: フェーズ木の展開

ここで、状態sが、どの部分問題空間に属しているのかということを判定するための条件を$C_{S_{ij}}$(フェーズフロー番号; i フェーズ番号; j)、サブゴール空間については、条件$C_{W_{ij}}$とする。問題解決は、以下の方法を順次用いることで可能となる。その流れを、図3.5に示す。

1. 問題発見方法
   全ての部分問題空間の条件$C_{S_{ij}}$　について、問題の状態 s が、その条件に合致（以下、「ヒット」ということとする）するかどうか調べる。

2. 問題選択方法
   （１）にて条件を満たしたフェーズに対し、以下の優先順位により問題を選択する。

   (a) 現在の問題の状態 s が、あるフェーズフローにおいて２個所以上の部分問題に対しヒットしている場合は、インデックスの小さいほう（部分問題の解決側）にある方を優先する。

(b) 現在の問題の状態 s が、２本以上のフェーズフロー上で、ヒットしており、そのヒットしている部分問題は、共に同じ、オペレータを使用するとき、フェーズフローウェートの大きいほうを優先する。

3. 問題解決方法

（２）にて選択された部分問題の問題解決を行う。問題解決は、その部分問題において利用可能なオペレータにより、仮想的に状態遷移を行い評価関数にて最良の評価が得られるものを採用する。

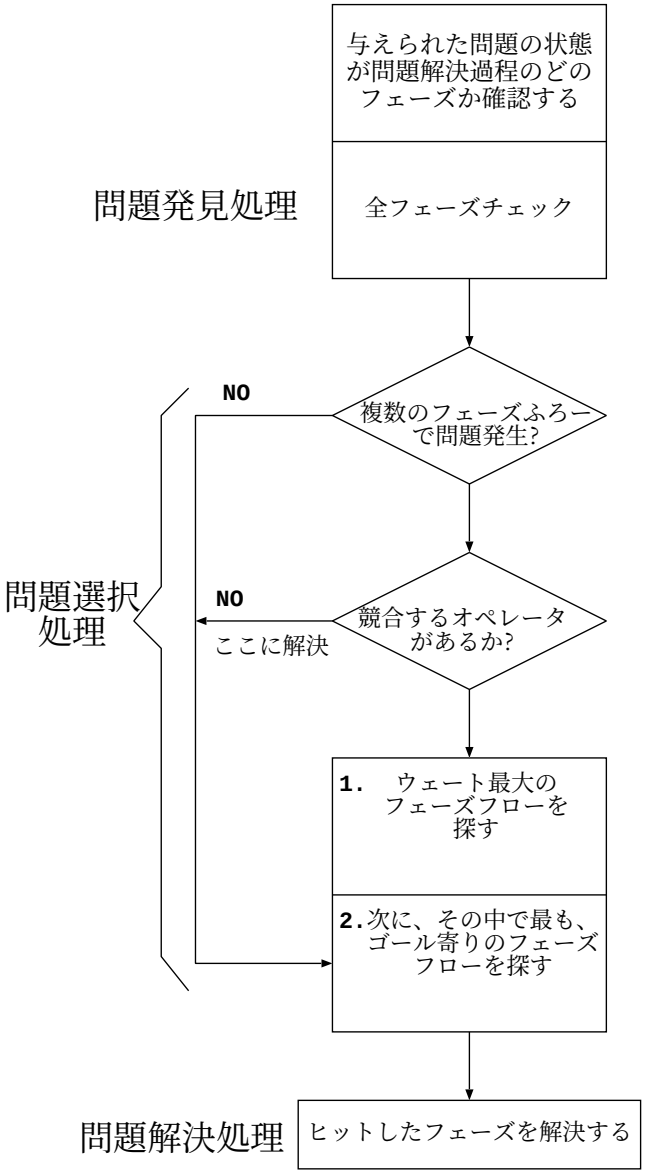


図 3.5: 問題解決プロセス

3.3 提案する問題解決方式の数学的定式化

3.2 項では、2 章の仮説に基づいて問題解決方式を提案した。ここでは、提案方式を数学的に再定義し、従来の問題解決方式との違い、問題解決が安定に進行するための条件等について検討する。

これまで研究されてきた多くの問題解決方式(または、システム)の定式化は、R.B.Barnerji による定式化 [Banerji 1969][Banerji 1983] に基づいている。そこで、本提案方式の定式化においても、Barnerji の定式化に基づいて構築する。まず、“問題” を定義し、そして、本問題解決の枠組で重要な部分問題を定義する。Barnerji の部分問題の定義では、初期状態を１状態として扱っていたが、本定式化では、１つに限定せず、状態集合として扱い部分問題同士の交わり部分は点ではなく空間として扱うこととした。そして、本定式化の最終目的は、部分問題を節点として見たとき問題解決ストーリーを一種のグラフとして表現できることを示し、これが問題解決を矛盾なく進めることを示す。定式化の流れは、図 3.6 のとおりである。

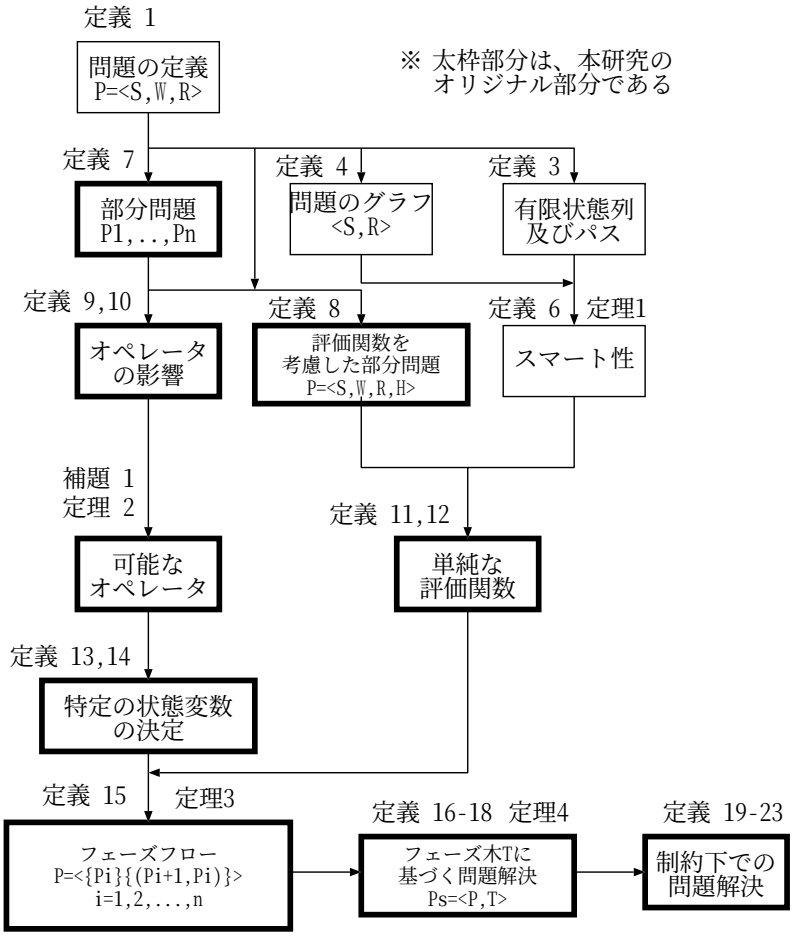


図 3.6: 定式化の体系

3.3.1 問題に関する定義

[Banerji 1983] の定式化では、問題 $(problem)P$は、“ S ”を“ 状態集合 ”、“ W ”を「勝ち状態集合」、そして、“ R ”を「動き $(move)$ の集合」とするとき、三つ組$< S, W, R >$によって与えられていた。問題 Pに関して、本研究においても基本的に、この定式化を踏襲するものとするが、S, W, Rの呼び方については、状態空間探索（または、状態空間探索システム）で用いられる用語（即ち、“ 問題空間 ”“ ゴール空間 ”“ オペレータ ”）を用いることとする。また、定義２、３にて、ゴール空間を条件式にて表現する考え方を導入する。この考え方は、部分問題空間同士の空間結合という考え方に必要となる。

定義 1 問題の定義 [Banerji 1983]

問題 Pは、三つ組$< S, W, R >$によって与えられる。ここで、Sは“ 問題空間 ”、$W \subseteq S$は“ ゴール空間 ”、また、$R \subseteq S \times S$は、“ オペレータの集合 ”という。そして、Rの個々の要素を、単に、“ オペレータ ”という。問題空間 Sは状態ベクトル $(p_1, p_2, ..., p_n)$ として表わされるとき、n 次元空間となる。また、任意の状態 $s_0 \in S$にオペレータを適用して生成された状態系列

$$s_0, s_1, ..., s_k \qquad where \qquad \forall(i)(s_i \in S)\&(s_k \in W) \tag{3.1}$$

を「解」という。

定義 2 問題 Pにおいて、ゴール条件 C_Wが定義できるとき、ゴール Wは、

$$W = \{s | C_W(s) = true\} \tag{3.2}$$

である。

定義 3 問題 P において、状態 $s \in S$がゴール空間 Wに存在するための条件が状態ベクトル $(p_1, ..., p_n)$ の各要素毎に、ゴール条件 $C_{W_1}, ..., C_{W_n}$ として定義できるとき、

$$if \qquad (\forall i)(C_{W_i}(p_i) = true) \qquad then \qquad s \in W \tag{3.3}$$

このとき、「問題 P は解決済」という。

また、特定の変数 p_iについて $C_{W_i}(p_i) = true$ ならば「p_iについて解決済」という。

3.3.2 問題のグラフおよびパス

「問題のグラフ」、「パス」、及び、「パスの長さ」について定義する。

定義 4 問題のグラフおよびパス [Banerji 1983]

組 $< S, W >$ は、「問題のグラフ」であるという。Rの各要素（s, s'）は、グラフの辺に対応する。各 i に対して $s_i R s_{i+1}$ならば、状態の列（$s_0, s_1, ..., s_n$）は、「パス $(path)$」という。このとき、n を s_0, s_nの間の「パスの長さ $(length)$」という。

3.3.3 パスが問題の解である条件

パスが解である条件を示す。

定義 5 パスが問題の解である条件 [Banerji 1983]
任意の有限列 $(s_0, s_1, ...s_n)$ は、$s_n \in W$かつ $s_i \in S$かつ（$s_i = s_{i+1}$あるいは、$s_i R s_{i+1}$）のとき、$s_0 \in S$に対する解であるということとする。$s_n \in W$ となる任意のパスは解である。

3.3.4 スマートな解

Banerji は、同じ状態を含まない（即ち、閉路を持たないグラフとしての）解の状態列を「スマートである」と呼んでいる。本研究でも同概念を用いる。

定義 6 スマートな解 [Banerji 1983]
問題 $< S, W, R >$ の解 $(s_0, s_1, ..., s_n)$ は、$i < n$ に対して、$\neg(s_i \in W)$ かつ $i \neq j$のとき $s_i \neq s_j$が成立するならば、「スマート $(smart)$ である」という。

定理 1 スマートな解 [Banerji 1983]
$s_0 \in S$に対する $P =< S, W, R >$ の解が存在するとき、必ず s_0に対する Pの「スマートな解」が存在する。(証明は $Appendix - A$ に示す。)

3.3.5 部分問題

[Banerji 1983] の中では、特定の初期状態をサブゴールに変換する問題 $< s, T, M >$ (ただし、$s \in S, T \subseteq S, M \subseteq R$) を部分問題としていた。本定式化では、以降、空間的概念が必要となるため拡張して定義する。

定義 7 部分問題
問題 $P = < S, W, R >$において、$S_i \subseteq S,\ W_i \subseteq S_i$ 及び $R_i \subseteq R$ が与えられ、問題 $P_i = < S_i, W_i, R_i >$ を形成とするとき、P_iは部分問題という。特に、S_iは部分問題空間という。

3.3.6 状態評価関数

[Banerji 1983] の中では、評価関数にて問題解決を進めるという議論はほとんどなく、問題自体の性質の解法の議論が中心であり、本定式化のなかでは、状態評価関数 Hを問題解決の枠組に含める。

定義 8 状態評価関数
問題 $P =< S, W, R >$ において、Sの要素を入力とする状態評価関数 Hを問題の構成要素に含む P', 即ち $P' = < S, W, R, H >$は「評価関数を有する問題」ということとする。

同様に、Pの部分問題 $P_i =< S_i, W_i, R_i >$ において、S_iの要素を入力とする状態評価関数 $H\,i$ が与えられたとき、$P' =< S_i, W_i, R_i, H_i >$ は、「評価関数を有する部分問題」ということとする。

3.3.7　オペレータの影響

$Banerji$ によれば、問題 $P = <S、W、R>$が与えられたとき Rの分割 Mは、各 $m \in M$に対して、$(s_1, s_2) \in m \& (s_1, s_3) \in m$ ならば $s_1 = s_3$が成り立つとき、$<S、W、R>$の「動きの集合 $(setofmove)$」と呼び、また、三ツ組$<S、W、M>$は$<S、W、R>$の「動き表現 $(moverepresentation)$」と呼んでいた。しかし、現実の問題を考える場合、この動き表現のように、完全に冗長性を排除できる問題は希である。そこで、本提案方式では、この様な厳しい条件を設けず、$m \in M$の定義域 S_mを、

$S_m = \{s | (\exists s' \in S)((s, s') \in m)\}$ とするとき、$s \in S_m$ならば、m は s に「適用可能」ということとする。また、$(s, s') \in m$ を、$s' = m(s)$ と表記するものとする。

なお、以下、動き表現については、敢えて考えないで定式化を更に進める。

$Banerji$ は、部分集合 $S' \subseteq S$および $m \in M$として、m の定義域を S_mとすれば、m が S'に「関連している $(relevant)$」とは、すべての $s \in S_m \cap S'$に対して$\neg(m(s) \in S')$ となるときをいい、m が S'に「無関係 $(irrelevant)$」であるとは、m が適用可能な任意の $s \in S'$に対し $s \in S'$となるのが $m(s) \in S'$のときであり、また、そのときに限るとと定義していた。即ち、状態集合に対する関係である。

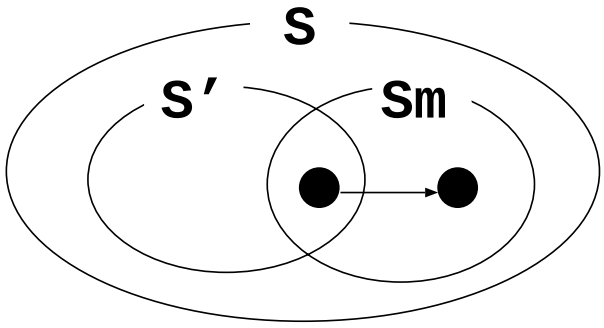


図 3.7: relevant 及び irrelevant

本項では、s を構成する状態変数のそれぞれについて「関連している」のか「無関係」なのかということについて議論する。

定義 9 問題 $P = \langle S, W, R \rangle$ において、空間 Sが n 個の状態変数 $p_1, p_2, ..., p_n$から構成されるものとする。また、ゴール空間に内包される空間 W'、即ち、$W' \subseteq W$は、

$$W' = (W_1, W_2, ..., W_n) \quad where \quad W_k = \{x | \text{整数}\, x, low_k \leq x \leq high_k\} \tag{3.4}$$

であるものとする。ここで、k番目の変数に対してのみ作用するオペレータの集合 $R_k \subseteq R$を考え、空間 Sの任意の状態 s、即ち、

$$\forall s = (p_1, p_2, ..., p_n) \in S$$

に R_kを適用する。ここで、変数 p_kに着目すれば、

$$(\exists m_k^+ \in R_k)\, m_k^+(s) = (p_1', p_2', \ldots, p_k + \Delta p_{k^+}, .., p_n')\, where\, 0 < \Delta p_{k^+} \leq high_k - low_k \tag{3.5}$$

$$(\exists m_k^- \in R_k)\, m_k^+(s) = (p_1'', p_2'', \ldots, p_k - \Delta p_{k^-}, .., p_n'')\, where\, 0 < \Delta p_{k^-} \leq high_k - low_k \tag{3.6}$$

なるオペレータ m_k^+, m_k^-が存在し、s に対し、オペレータ m_k^+または m_k^-を有限回適用し p_kを p_k'(但し $p_k' \in W_k$) に変換できるならば、R_kは、変数 p_kに対して解決可能ということとする。

定義 10 評価関数を考慮した問題 $P = \langle S, W, R, H \rangle$ において、空間 Sが n 個の状態変数 $p_1, p_2, ..., p_n$から構成されるものとする。任意の変数 p_iに対し解決可能なオペレータ集合 $R_i \in R$があるものとする。ここで、R_iの変数 $p_1, p_2, ..., p_n$への影響を関数 $influence$ によって、

$$influence(Ri) = (e_{i1}, e_{i2}, ..., e_{in}) \tag{3.7}$$

但し、各 $e_{ij}(j = 1, ..., n)$ に対して、

$$\begin{aligned} e_{ij} \quad = \quad & 0 \quad (R_i\text{の任意のオペレータは } p_i\text{に「無関係である」とき}) \\ & 1 \quad (R_i\text{が } p_i\text{に「関係している」とき}) \\ & 2 \quad (R_i\text{の解決対象が } p_i\text{であるとき}) \end{aligned} \tag{3.8}$$

と表わすものとする。$influence(Ri)$ が状態変数 p_jに与える影響を、e_{ij}と書くものとする。このとき、$i = 1, ..., n$ 及び $j = 1, ..., n$ によって構成される行列 $[e_{ij}]$ は、行列 Eにより

$$E = \begin{bmatrix} e_{11} & e_{12} & e_{13} & \cdots & e_{1n} \\ e_{21} & e_{22} & e_{23} & \cdots & e_{2n} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ e_{n1} & e_{n2} & e_{n3} & \cdots & e_{nn} \end{bmatrix}$$

この行列 Eをオペレータの影響マトリックスとよぶこととする。

補題 1 問題 Pにおいて状態空間 Sの各状態変数を独立に制御するオペレータが存在するとき、影響マトリックス $E = [e_{ij}](i = 1, ..., n$ 及び $j = 1, ..., n)$ において、

$$e_{ij} = \begin{cases} 2 & (i = j) \\ 0 & (otherwise) \end{cases}$$

ならば、$s \in S \& \neg(s \in W)$ である任意の状態 s を $s \in W$ とすることが可能である。

証明

問題 Pにおいて、影響マトリクス $E = [e_{ij}](i = 1, ..., n$ 及び $j = 1, ..., n)$ において $e_{ij} = 2(i = j) \& e_{ij} = 0(i \neq j)$、即ち、解決対象以外のパラメータは無関係である。解決対象が可解であるならば、その対象変数について解決することが可能なオペレータを $m \in R_i$、解決対象状態 s とするとき、s に m_iを有限回適用して、p_iを解決し、新たな状態を生成することを $R_i[s]$ と書くものとすれば個々のパラメータを所望の値に変換可能である。即ち、

$$R_1[s] = s_1 \quad where \quad C_{W_1}(s_1) = true$$

$$R_2[s_1] = s_2 \quad where \quad C_{W_2}(s_2) = true$$

$$\vdots$$

$$R_n[s_{n-1}] = s_n \quad where \quad C_{W_n}(s_n) = true$$

さらに、仮定より、$R_i(i = 1, ..., n)$ は、p_iのみを変化させ、その他の変数は変化させないので、

$$C_{W_1}(s_n) = C_{W_2}(s_n) = \ldots = C_{W_n}(s_n) = true$$

よって、可解である。

定理 2 問題 $P = \langle S、W、R \rangle$ において、$s = (p_1, p_2, \ldots, p_n) \in S$また $R_i \in R$は、i 番目のパラメータ p_iを操作するオペレータとするとき、オペレータ R_jにオペレータ R_iを有限回適用し合成し、R_iの解決対象の影響を打ち消したオペレータを、$R_i[R_j]$ と記述するものとする。ここで、以下のような新たなオペレータ $R'_i \in R$を考え、

$$R'_i = R'_{i-1}[R'_{i-2}[\ldots R'_1[R_i]\ldots]] \quad i = 1 \sim n$$

オペレータの生成を実施し、以下の２つの条件のどちらかが満足されているならば可解である。

1. $influence(R_i, j) = 0(i = 2, ..., n$ 及び $j < i)$
2. $influence(R_i, j) = 1(j < i)$ のとき $influence(R'_j, i) = 0$

なお、$influence(R_i, j)$ は、R_iの j番目のパラメータに対する影響を示す。

証明

1. の条件については、自明。

2. の条件については、以下のとおり。

＊は、0 または 1 のどちらかの値をとるとき、

$influence(R_1) = (2, *, \ldots, *)$ のとき $R'_1 = R_1$とすると $influence(R'_1) = (2, *, \ldots, *)$

$influence(R_2) = (*, 2, *, \ldots, *)$ のとき $R'_2 = R'_1[R_2]$ とすると $influence(R'_1) = (0, 2, *, \ldots, *)$

$influence(R_3) = (*, *, 2, \ldots, *)$ のとき $R'_3 = R'_2[R'_1[R_2]]$ とすると $influence(R'_1) = (0, 0, 2, *, \ldots, *)$

$$\vdots$$

$$influence(m_3) = (*, *, *, \ldots, 2) \text{ のとき } R'_n = R'_{n-1}[R'_{n-2} \ldots R'_1[R_n] \ldots]$$

とすると、

$$influence(R'_1) = (0, ..., 0, 2)$$

更に、合成して

$$\begin{bmatrix} influence(R'_n[R'_{n-1}[\ldots R'_2[R'_1]\ldots]]) \\ influence(R'_{n-1}[\ldots R'_2[R'_1]\ldots]) \\ \vdots \\ influence(R'_1) \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 & \ldots & 0 \\ 0 & 2 & 0 & \ldots & 0 \\ & & \vdots & & \\ 0 & 0 & 0 & \vdots & 2 \end{bmatrix}$$

3.3.8 単調な状態評価関数に基づく問題

ここでは、状態評価関数が単調関数である場合について以下の定義を行なう。

定義 11 問題 $P = \langle S、W、R、H \rangle$ において、任意の状態 $s \in S$について、オペレータの集合 Rの各要素を１回適用することによって生成される状態の集合を

$$R'(s) = \{s' | m \in R, s' \in m(s)\}$$

と書くものとすれば、$R'(s)$ から最良 (仮に、最小の評価) の状態評価量を有する状態 s' を選択する関数 $best$ は、ｓを入力として

$$s' = best(s) \quad where \quad (\exists s' \in R'(s))(\forall s'')(s'' \in R'(s) \& H(s'') \geq H(s'))$$

と書くものとする。また、s に対して関数 $best$ を n 回適用することを $best^n(s)$ と書くものとする。

定義 12 問題 $P = \langle S、W、R、H \rangle$ において、任意の状態 $s_0 \in S$に対し、$s_0, s_1, \ldots, s_n$ は $s_{i+1} = best(s_i)$ によって生成したスマートな解であり、

$$H(s_0) > H(s_1) > \ldots > H(s_n)$$

$$where \quad i = 0, \ldots, n-1 \text{ のとき} \quad \neg(s_i \in W)$$

また

$$i = 0 \text{ のとき}, \ldots s_i \in W$$

が成り立つとき、Pは「単調関数によりモデル化されている」という。

3.3.9 特定の状態変数の解決

定義 13 一般に、問題 $P = \langle S, W, R \rangle$ において、空間 Sが n 個の状態変数からなり、そのうち２つの状態 s, s'を、

$$s = (p_1, p_2, \ldots, p_n) \in S$$

$$s' = (p'_1, p'_2, \ldots, p'_n) \in S$$

とし、

$$s = m(s')$$

なるオペレータを $m \in R$が存在とするものとする。即ち、

$$(p_1, p_2, \ldots, p_n) = m((p'_1, p'_2, ..., p'_n))$$

このとき

$$(\exists j)(p_j \neq p'_j)$$

ならばオペレータ m は、変数 p_jに作用するということとする。

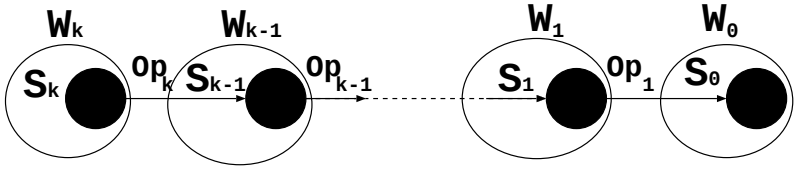
定義 14 問題 $P = \langle S, W, R, H \rangle$ において、その状態空間 Sが n 個の状態変数からなるものとする。ここで、W_kは状態変数 p_kに関するゴール、R_kは p_kに作用するオペレータ集合、および、評価関数 H_kは、p_kに関する評価関数であるものとする。このとき、問題 $P' = \langle S, W_k, R_k, H_k \rangle$ の解がスマートであり、

$$H_k(s_0) > H_k(s_1) > ..... > H_k(s_n) \quad where \quad s_n \in W_k$$

が成り立つとき、問題 P'は「変数 p_kに関して単調関数によりモデル化されている」という。

3.3.10 フェーズ・フロー

[Banerji 1983] においては、問題 $P = \langle S, W, R \rangle$ の、状態集合 Sの部分集合 $\{W_i\}_{i \geq 0}$のことを「状態集合 Sの評価」といい、各 W_iは、Pの解の各状態のサブゴール集合と位置づけられており、1回のオペレータの適用により、次のサブゴール空間へ遷移することを前提に考えられていた。また、$\cap_{k \geq 0} W_k = \phi$と定義されていた。



しかし、現実の問題は、このような単純な定義は、当てはまり難く、このようにW_iの分割を厳密に与えることは困難である。そこで、ここでは、各サブゴール空間間にオーバーラップ領域を持つことを、を許容する定式化を行う。定義 15 で与える「フェーズ・フロー」は、問題解決の流れを示しており、また、個々の部分問題空間は問題解決フェーズを意味している。

定義 15 問題 $P = \langle S, W, R \rangle$ に関して、評価関数を有する部分問題 $P_i = \langle S_i, W_i, R_i, H_i \rangle (i = 0, 1, 2, \ldots, k)$ が定義でき、各部分問題毎の特定の変数 $p_j (j = 1, \ldots, n)$ において、以下の条件を満足するものとする。S_iは部分問題空間、W_iは、そのゴール空間であり、

1. $W_i = \{s | s = (p_1, p_2, \ldots, p_n) \in S_i, low_i \leq p_j \leq high_i\}$ 但し, $W_0 \subseteq W$
2. H_iは部分問題における評価関数であり、p_jに関する単調関数
3. R_iは、p_jに対して制御可能なオペレータ集合

また、部分問題 P_i及び P_{i-1}において、

$$W_i \subseteq S_{i-1} \qquad (i \geq 1) \tag{3.9}$$

となる関係を (P_i, P_{i-1}) と表するものとすれば、全部分問題が $P_0, \ldots, P_k$からなる場合、

$$\text{組} \langle \{P_0, P_1, P_2, \ldots, P_k\} \{(P_k, P_{k-1}), \ldots, (P_3, P_2), (P_2, P_1), (P_1, P_0)\} \rangle \tag{3.10}$$

は、各部分問題の関係を1列にリンクされたグラフによって表わす。これを Pに関するフェーズ・フローと呼ぶこととする。特に、$S_F (= S_0 \cup S_1 \cup, \ldots, \cup S_k)$ はフェーズフローの定義域、さらに $W_F (= W_0)$ はフェーズ・フローの値域ということとする。

3.3.11 フェーズフローの可解性

ここでは、フェーズフローの可解性について検討する。

補題 2 問題 $P = \langle S, W, R \rangle$ を解決するためのフェーズ・フローを Fとし、また Fの定義域を S_Fとする。このとき、Sと S_Fの間には以下の関係が成り立つ。

$$S_F \subseteq S$$

証明

Fを構成する任意の部分問題 $P_i = \langle S_i, W_i, R_i \rangle$ において、$S_i \subseteq S$かつ $S_F = \cup_{i \geq 0} S_i$なので $S_F \subseteq S$となる。

補題 3 問題 $P = \langle S, W, R \rangle$ を解決するためのフェーズ・フローを Fとし、定義域を S_Fとする。このとき、$S_F \subseteq S$内の任意の状態に対し問題 Pを解決可能である。

証明

$P = \langle S, W, R \rangle$ に関するフェーズ・フロー Fが存在し、Fの定義域を S_F、値域を WFとするとき、フェーズ・フローの定義より $W_F = W_0$かつ $W_0 \subseteq W$が成り立つので $W_F \subseteq W$となる。ここで、$\forall s \in W_F$ならば $s \in W$となる。

一方、Fにおいては、$\forall s \in S_F$に対して、$\exists s' \in W_F$に変換可能である。とすれば、$\forall s \in S_F$に対して、$\exists s' \in W$に変換可能であり、S_Fの任意の状態に対して問題 Pを解決可能である。

定理 3 問題 $P = \langle S, W, R \rangle$ を解決するためのフェーズ・フロー Fが存在し、その定義域を S_F, 値域を W_Fとするとき、任意の状態 $s \in S_F$に深さ 1 の最良優先探索によるオペレータの決定と適用を繰り返すことで、s を $s' \in W_F$に変換することが可能である。

証明

フェーズ・フロー Fにおいて、その任意の部分問題 $P_i = \langle S_i、W_i、R_i \rangle$ とするとき、これは特定のパラメータに関して単調な関数でモデル化されており、フェーズ・フローの定義より、任意の状態 $s \in S_i \& s \in W_i$ならば

$$(\exists s' \in W_i)(\exists n)(s' = best^n(s) \in S_{i-1})$$

$i = 0$ のとき　任意の状態 $s \in S_0 \& s \in W_0$ならば

$$(\exists s' \in W_0)(\exists n)(s' = best^n(s) \in S_0)$$

なので問題全体が解決済み。
$i = k$のとき　任意の状態 $s \in S_k \& s \in W_k$ならば

$$(\exists s' \in W_k)(\exists n)(s' = best^n(s) \in S_{k-1})$$

として、

$i = k+1$ のとき　任意の状態 $s \in S_{k+1} \& s \in W_{k+1}$ならば $(\exists s' \in W_{k+1})(\exists n)(s' = best^n(s) \in S_k)$

が成り立つ故に、

$$s \in S_0 \cap S_1 \cap S_2, \ldots \text{ならば } (\exists s' \in W_0)(\exists n)(s' = best^n(s) \in S_0) \qquad (3.11)$$

（証明終わり）

定義１３及び補題 2,3 より、フェーズ・フローの定義域内の状態は、値域内の状態に変換可能である（即ち、元の問題 P の枠組み内で解決する）と言える。
フェーズ・フローの定義域に対する可解性が証明できたところで、以下の定義を行う。

補題 4 問題 $P = \langle S, W, R \rangle$ を解決するためのフェーズ・フロー Fを関数と見なし、S_Fを定義域、W_Fを値域とすれば、$(\forall s_1 \in S_F)(\exists s_2 \in W_F)(s_2 = F(s_1))$ が成り立つ。

証明

補題２、３及び定理３より明らか。

3.3.12 フェーズ木

定義 16 問題 Pを解決するためのフェーズ・フロー F(即ち、$S_F \subseteq W$) が存在するものとする。Pに関する別のフェーズ・フロー F'が存在して、このとき F及び F'の定義域が S_F 及び $S_{F'}$, 値域が W_F及び $W_{F'}$であるとき、$W_{F'} \subseteq S_F$かつ$\neg(W_F \subseteq S_{F'})$ ならば Fはメイン・フロー、F'はサブ・フローということとする。

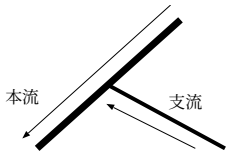
定義 17 問題 Pとそれを解決するためのメイン・フロー及び 0 個以上のサブ・フローからなるフェーズ・フローの結合体 T_{Flows}をフェーズ木 P_sは以下のように表わすものとする。

$$P_s = \langle P, T_{Flows} \rangle$$

定義 18 問題解決 $P_s = \langle P, T_{Flows} \rangle$ とする。また、を構成するメイン・フロー F_0, サブ・フロー $F_1, F_2, \ldots$ とし、それらの定義域 $S_T(= S_{F1} \cup S_{F2} \cup, \ldots)$、値域は $W_T(= W_0)$ とする。

このとき、Pに関する別のフェーズ・フローが存在して、$W_{F'} \subseteq S_T$かつ$\neg(W_{F'} \subseteq S_{F'})$ ならば F'ものサブ・フローである。

ここで、フェーズ木を川に喩えれば、メイン・フローという「本流」に対し、サブ・フローという「支流」が結合した形態と考えることができる。相対的な関係、即ち、2つのサブ・フローが存在し、一方が、他方の中間位置に結合している場合は、結合されている方が「本流」となる。



定理 4 T_{Flows}が 2 つ以上のフェーズフローから構成されるものとする。問題解決 $P_s = \langle P, T_{Flows} \rangle$

$$F_A = \langle \{P_{A_0}, P_{A_1}, \ldots, P_{A_K}\} \{(P_{A_K}, P_{A_{K-1}}), \ldots, (P_{A_2}, P_{A_1})(P_{A_1}, P_{A_0})\} \rangle$$

$$F_B = \langle \{P_{B_0}, P_{B_1}, \ldots, P_{B_K}\} \{(P_{B_K}, P_{B_{K-1}}), \ldots, (P_{B_2}, P_{B_1})(P_{B_1}, P_{B_0})\rangle \}$$

とするとし、任意に 2 つの交わり空間 L_1, L_2とするとき、
F_Aにおける交わり空間 $S_{A_i}, S_{B_j}(i, j = 0, \ldots, n)$
F_Bにおける交わり空間 $S_{B_k}, S_{B_l} = 0, \ldots, n)$
において

$$L_1 \subseteq S_{A_i} \cap S_{B_k}$$

$$L_2 \subseteq S_{A_j} \cap S_{B_l}$$

であるものとするとき、P_sが解決可能であるための必要条件は、以下 (1), (2) を満足することである。

1. $0 < i < j$かつ $0 < m < n$
2. Bが本流で、かつ F_Aおけるインデックス $i = 0$ のとき $W_{A_0} \subseteq L_1$
   または
   A が本流で、かつ F_Bおけるインデックス $k = 0$ のとき $W_{B_0} \subseteq L_1$

証明

1. 無限ループの回避

   条件 1.は、フェーズ・フローが同一方向を示している。即ち、L_2から L_1へ向かう流れは符号を考えるとき、

   $$F_A\text{について } i - j < 0$$

   $$F_B\text{について } n - m < 0$$

   となり、共に減少の傾向にある。したがって、L_1, L_2において、どちらのフェーズ・フローの流れに沿った解決を実施しても逆戻り現象は生じない。

2. 終端条件

   条件 2.は、交わり領域を有するフェーズ・フローの終端条件を示す。F_Bが本流で F_A において、フェーズ $i = 0$ のとき F_Bと交わりを持つならば、$W_{A_0} \subseteq L_1$となる。即ち、$\forall S_a \subseteq S_A F_A(S_a) \in L_1$ならば $L_1 \subseteq S_B$なので本流 F_Bにおける問題解決に続く。FA が本流で、FBにおいて、フェーズ $k = 0$ のとき F_Aと交わりを持つならば $W_{B_0} \subseteq L_1$となる。即ち、$\forall S_b \subseteq S_B F_B(S_b) \in L_1$ならば $L_1 \subseteq S_A$なので本流 F_Aにおける問題解決に続く。

3.4 可解性および解決不能状態に関する考察

問題解決においてゴールに到達できない根本的要因として、

1. 計算量爆発を起こす場合
2. オペレータによる状態の変化量が大き過ぎる場合
3. 解決処理が Absorbing State に陥る場合

が考えられる。これらの要因を整理し考察する。

1. フーズ・フローに基づく問題解決は、「探索深さ」が１段の最良優先探索であるため「探索の広さ」即ち「オペレータの種類」を制限すれば、計算量爆発を生じることはない。従って、オペレータを無限に生成する機能を持つシステムでは、保証されない。

2. オペレータによる状態の変化量に関してゴール空間に対して大き過ぎるの場合は、たとえ、問題解決の方向が正しいとしても、状態変化をゴールにて止めることが出来ず、結果として解決できない状況に陥る。フェーズフローの枠組みは、このような事を回避する枠組みとなっている。（定義 15）

3. Absorbing State の可能性として、個々の「部分問題」に関しては、本枠組みが定義 4 のとおり、スマートな解を与えることが前提となっている。スマートな解を与える条件は、「単調な状態評価関数」(定義 11,12) と「可解条件を満たすオペレータ」(定理 2) である。また、「フェーズフロー木 T_{Flows}」については、定理４により、フェーズフロー相互において、解決方向が逆行する交差が生じないことが枠組として盛り込まれている。従って、Absorbing state は生じない。

3.5 制約を伴う問題解決

3.5.1 問題空間に制約を伴う問題

現実の世界における問題空間は、オペレータを無限に作用させてできる空間を持っているわけではない。たとえば、ボードゲームにおいては、ボード大きさの制限がある。また、航空機の制御においては、高度は地面より低くはならない。そこで、まず、空間的制約について考える。

定義 19 問題 $P = \langle S, W, R \rangle$ に空間的制約条件 C_{SP}を与えて新たな問題を定義する。

$$S_{C_{SP}} = \{s | s \in S \& C_{SP}(s) = true\}$$

$$W_{C_{SP}} = \{s | s \in S \& C_{SP}(s) = true\}$$

とするとき、問題 $P_{C_{SP}} = \langle S_{C_{SP}}, W_{C_{SP}}, R \rangle$ の任意の部分問題の問題空間とゴール空間も C_{SP}の制約を受ける。

定義 20 問題$P = \langle S, W, R \rangle$において、空間的制約$C_{SP}$を伴う問題を$P_{C_{SP}} = \langle S_{C_{SP}}, W_{C_{SP}}, R \rangle$とする。このとき、$P$の任意の解 $s_0, s_1, \ldots, s_n$の中の任意の状態 s_iは、$S' = \{s | C_{SP}(s) = true\} \& s_i \in S'$を満たすものとする。もし、ならば、それは、解ではなく、また、他に解が存在しないとき、「空間的制約よって可解性が失われた」という。

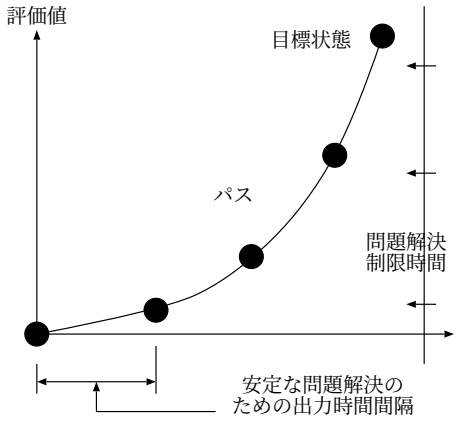
定義 21 問題 $P_{C_s} = \langle S_{C_s}, W_{C_s}, R \rangle$ において、それを解決するためのフェーズ木任意の解 $s_0, s_1, \ldots, s_n$の任意の状態 s_iは

$$(\exists S') \quad S' = \{s | C_s(s) = true\} \& s_i \in S'$$

である。

3.5.2 実時間制約

これまでの定義は、問題解決に要する時間を考慮せず、解決出来るのか、出来ないかのを問題にしており、ある意味で静的な問題の定義と言える。ここでは、時間的制約を考慮する。実時間で変化する環境で安定に問題を解決する場合には、オペレータを適切な時間内に出力する必要があると同時に、適切な時間内にて問題全体を解決する必要がある。



定義 22 問題 Pにおいて、解決方法を M、オペレータの決定のための制限時間を T_o、問題解決のための制限時間 Tp とすれば、実時間制約下での問題解決 Ps を 4 項組で以下のように書くものとする。

$$P_s = \langle P, M, To, Tp \rangle$$

いま、$P = \langle S, W, R \rangle$ は解決可能な問題であるとする。任意の状態 $s_i \in S$において、オペレータを決定するのに要する実質的所用時間 $T(s_i)$ とし、また、この問題を解決するのに要する実質的所用時間を T_{total}とするとき、

$$T(s_i) < T_o \& T_{total} < T_p$$

が成り立てば、問題 Pは方法 Mにより実時間で問題解決が可能であるということとする。

定義 23 問題 Pにおいて、フェーズ木が構成でき、オペレータの決定のための制限時間を T_o、全問題解決のための制約時間 T_pとすれば、このとき問題解決 P_sは

$$P_s = \langle P, T_{Flows}, T_o, T_p \rangle$$

と書ける。

3.6　まとめ

本章では、問題解決方式提案し、数学的に定式化した。そして、一定の条件を満たして問題を定義していけば、矛盾なく解決できることを示した。次章では、これを基にシステムとしての枠組を考え、その設計法について考える。

第 4 章

提案方式による問題解決システムとその設計

4.1 はじめに

本章では、前章の数学的な定式化の内容に基づき、システムを構築するために必要な（問題固有の）戦略知識を明らかにし、問題解決エンジンと共に、問題解決システムとしての枠組みを定め、そして、さらに、そのような要素を如何に獲得し、どのような手順でシステムに組み込むかということを明らかにする。また、システムの構成及びシステム設計法において従来のシステムと何が違うのか、メリット・デメリットについても議論する。

4.2 システムの枠組み

本問題解決方式の中心的考え方、すなわち、フェーズフロー

$$< \{P_0, P_1, .., P_k\}\{(P_k, P_{k-1}), ..., (P_2, P_1), (P_1, P_0)\} > \tag{4.1}$$

は、エキスパートから獲得した問題解決戦略を、部分問題の系列に置き換え、個々の部分問題を節点としたグラフとして表現したものである。この {} にて囲まれた２つの項は隣接する問題を定義したものであるが、隣接の定義は、個々の問題 P_iの中にある。たとえば、$P_i = < S_i, W_i, R_i, H_i >$は、当然、$W_i \in S_i$であるが、$W_i \in S_{i-1}$なら、$(P_i, P_{i-1})$ というように有向辺を構成でき、全体として問題解決の方向を与えることができる。

ここで、提案方式を用いて問題解決システムを構築する場合、フェーズフローを構築するために以下の情報が必要となる。なお、添字 i は、フェーズフローのインデックスであり、添字 j は、フェーズのインデックスである。

表 4.1: 解決戦略データベース

項目	定式化上の記号	実装上の扱い	実装上の記号
問題 Pの問題空間	S	条件式	C_S
問題 Pのゴール空間	W	条件式	C_W
問題 Pのオペレータ	R	状態遷移関数集合	R
問題 Pの評価関数	H	評価関数	H
部分問題 P_{ij}の問題空間	S_{ij}	条件式	$C_{S_{ij}}$
部分問題 P_{ij}のサブゴール空間	W_{ij}	条件式	$C_{W_{ij}}$
部分問題 P_{ij}のオペレータ	H_{ij}	状態遷移関数集合	R_{ij}
フェーズフローウェート	w	定数	w_i

4.3 システム構成

問題解決のためのソフトウェアは、6 つの機能モジュール（入力インターフェース、解決戦略データベース、問題発見モジュール、問題選択モジュール、問題解決モジュール及び出力インターフェース）からなる。相互の関係は、図 4.1のとおりである。斜線の入った矢印は解決戦略、白抜きの矢印は通常の問題解決の流れ、また、二重線の矢印は適切かつ効果的な問題解決を制御するためのデータの流れを示している。

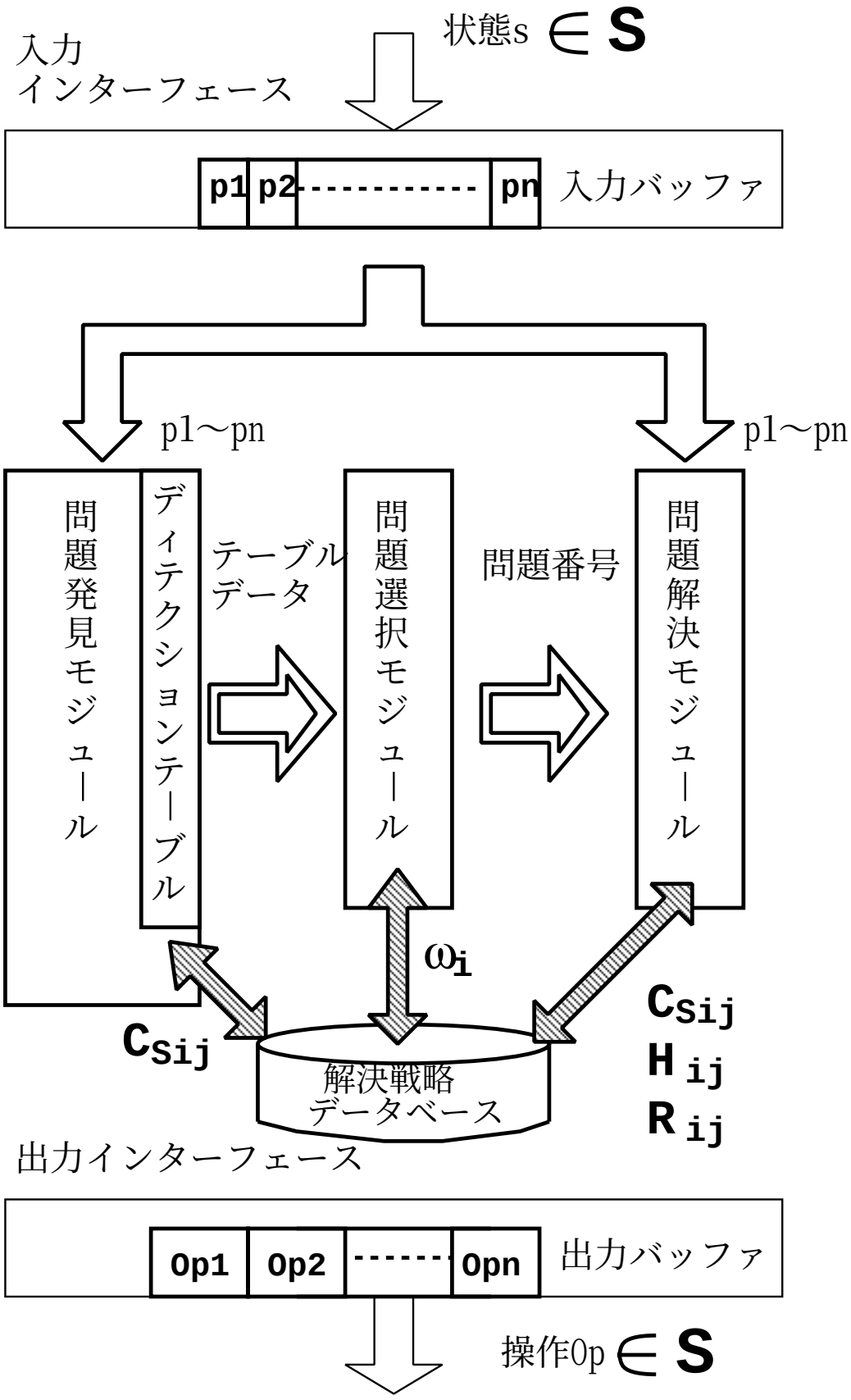

状態s ∈ S
入力
インターフェース
p1 p2 ---------- pn
入力バッファ
p1～pn
p1～pn
問題発見モジュール
ディテクションテーブル
テーブルデータ
問題選択モジュール
問題番号
問題解決モジュール
ωi
Csij
Csij
Hij
Rij
解決戦略
データベース
出力インターフェース
Op1 Op2 ------ Opn
出力バッファ
操作Op ∈ S

図 4.1: システム構成

4.3.1 入力インターフェース

解決対象の状態変数が入力データとして取り込まれる。入力データは、必要なデータ加工 (単位変換、データ・アンパック etc) を実施した上で、システム内で計算に使用するための共通データバッファにセットされる。共通データバッファのデータは、主に問題発見モジュールと問題解決モジュールにて使用される。

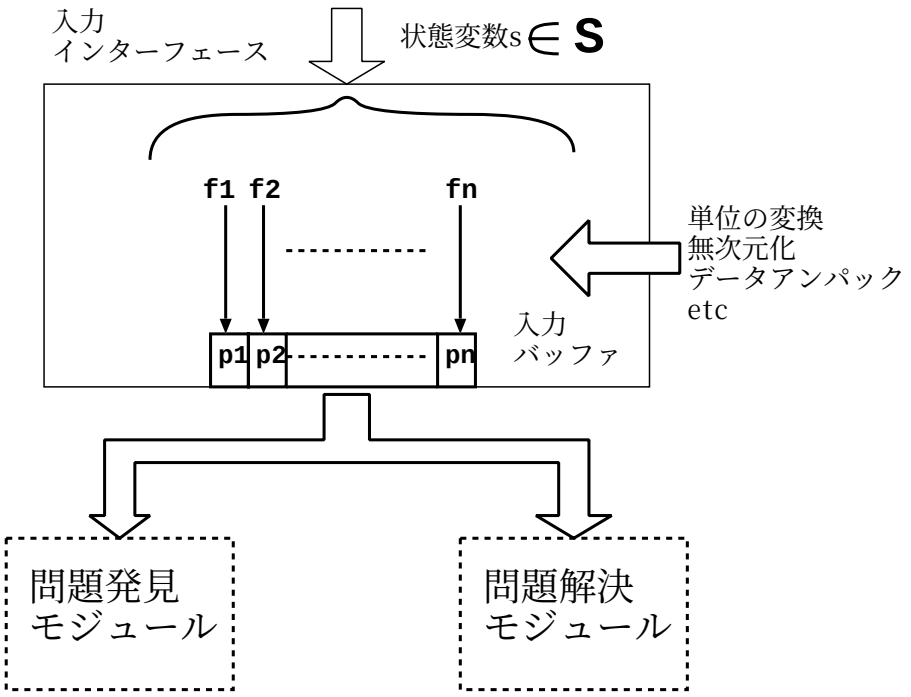


図 4.2: 入力インターフェース

4.3.2 解決戦略データベース

フェーズフローに関するデータ、フェーズフローウェートを格納する。本データのクライアントは、問題選択モジュール、問題解決モジュールである。

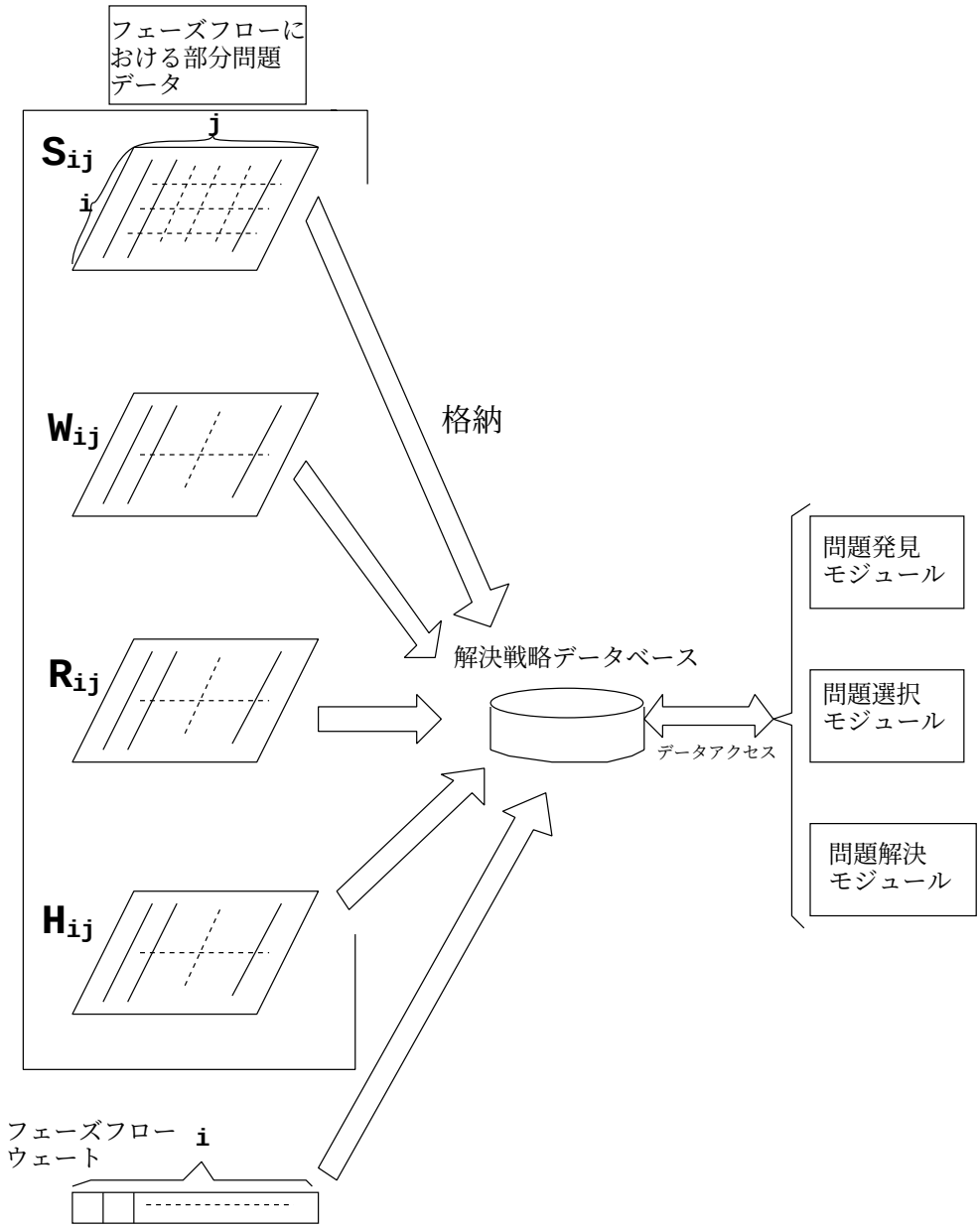


図 4.3: 解決戦略データベース

4.3.3 問題発見モジュール

問題発見モジュールは、複数存在する部分問題のどこに、問題が発生しているかをチェックし、発見するモジュールである。このモジュールでは、各フェーズフローのフェーズ条件を順次調べていく。マッチしたフェーズ条件が各フェーズのゴールでなければ、問題が発生していることになる。問題が発見されたフェーズは、直ちに、detection table に値１がセットされる。また、問題の解決が確認されたフェーズには、値０がセットされる。

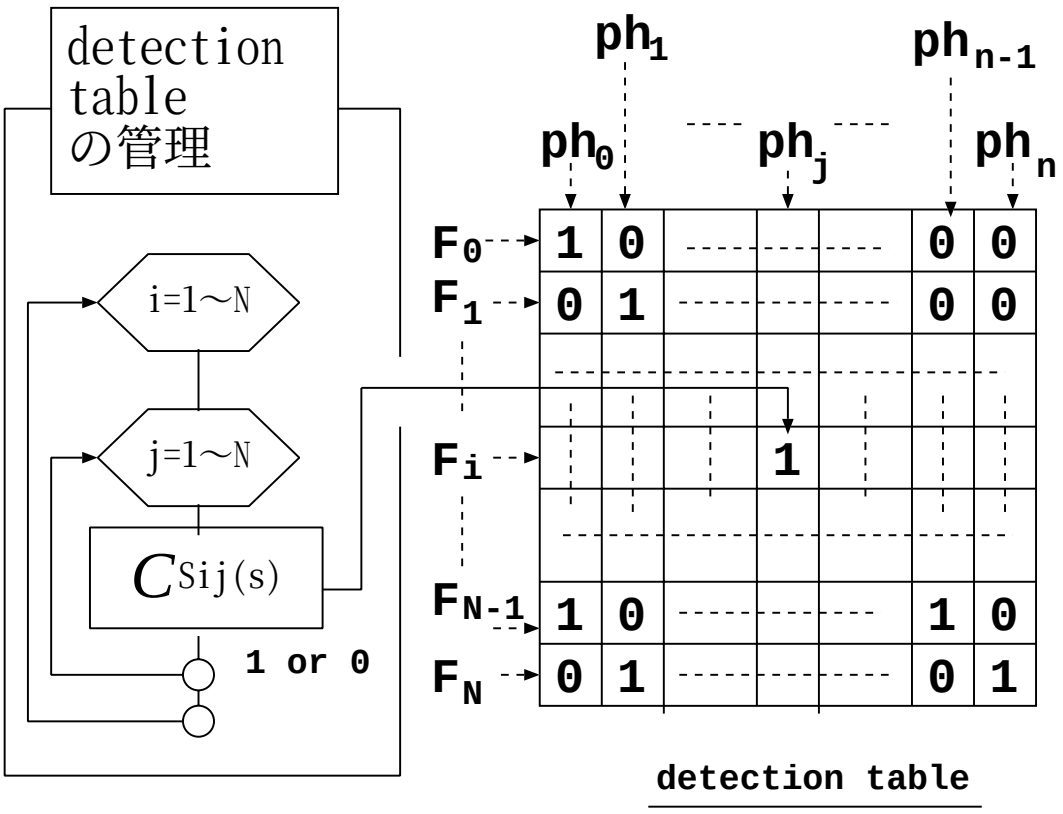


図 4.4: 問題発見モジュール

4.3.4 問題選択モジュール

問題選択モジュールは、detection table のデータをもとにどの問題を解決するかを選択する。対象問題の選択は、フェーズフローの重みテーブルとオペレータの競合をチェックする。もし競合が発生したら、ウェートの大きい方を優先する。

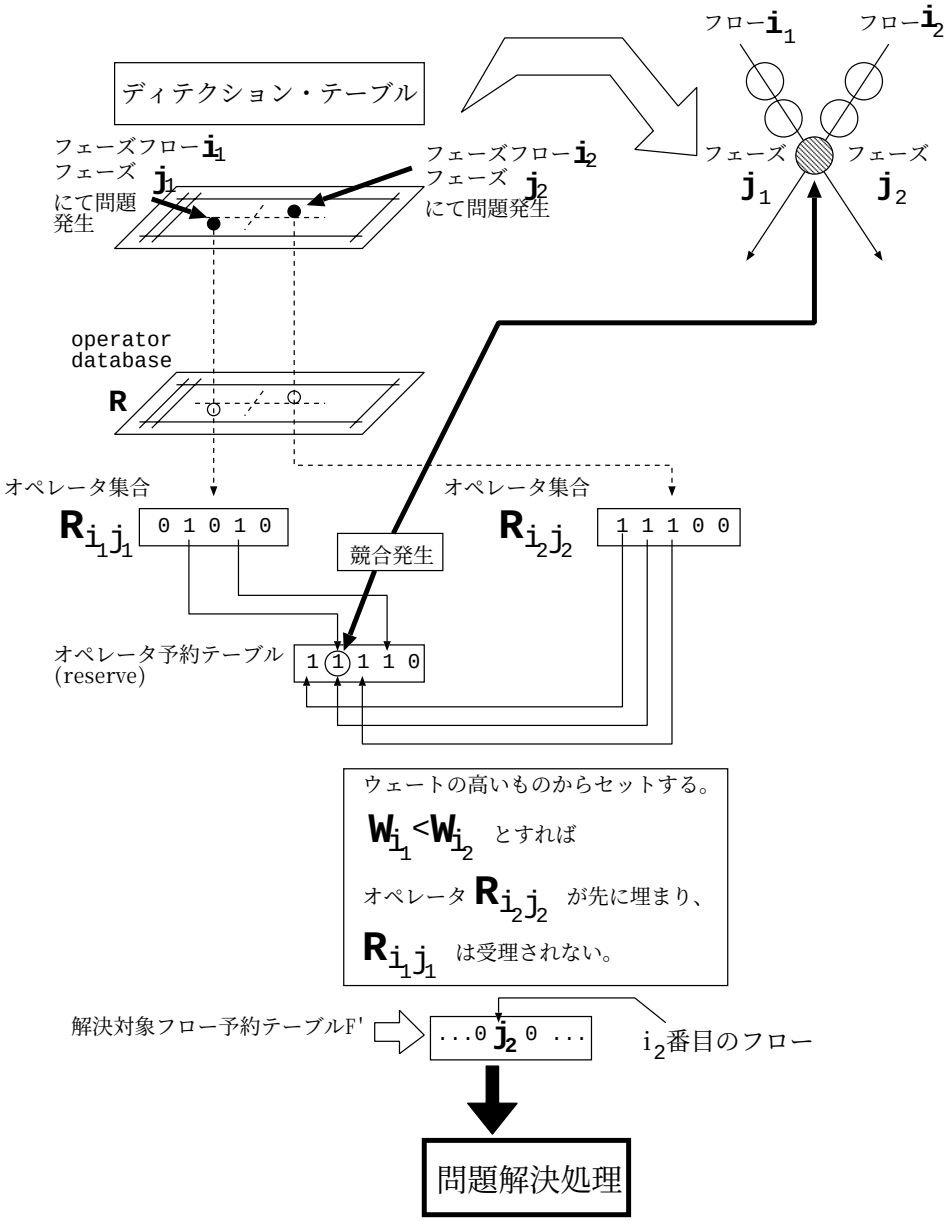


図 4.5: 問題選択モジュール

4.3.5 問題解決モジュール

本モジュールは、問題選択モジュールにて選択されたフェーズに対し、その該当する部分問題を解決する。解決戦略データベースのオペレータ R および関数 H を用い、現在の状態 s を、オペレータにて仮想展開し、関数 H にて評価最良となるオペレータを選択する。本モジュールの目的は、最良のオペレータを決定することである。

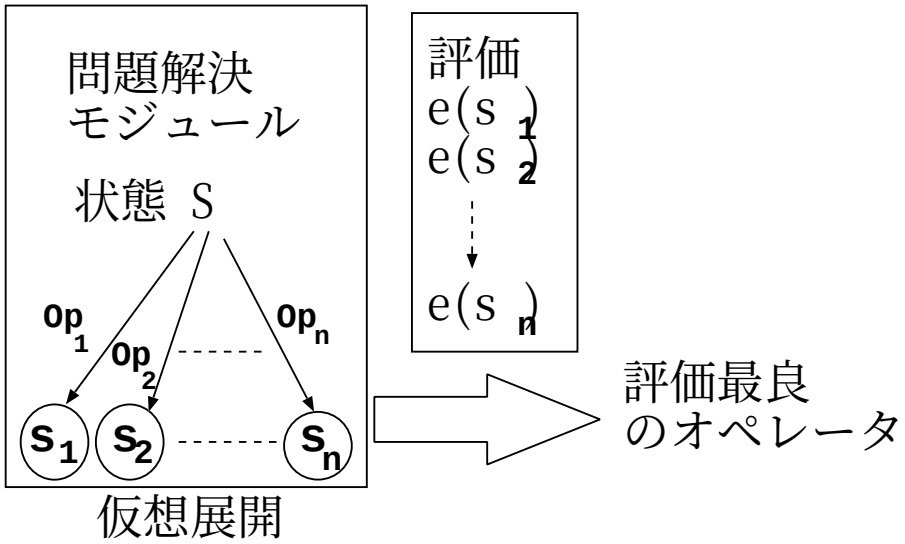


図 4.6: 問題解決モジュール

4.3.6 出力インターフェース

問題解決モジュールにて決定したオペレータを実現するための具体的操作を外部出力バッファにセットする。問題選択モジュールにて、予め、競合解消されているので出力テーブルの競合は生じない。

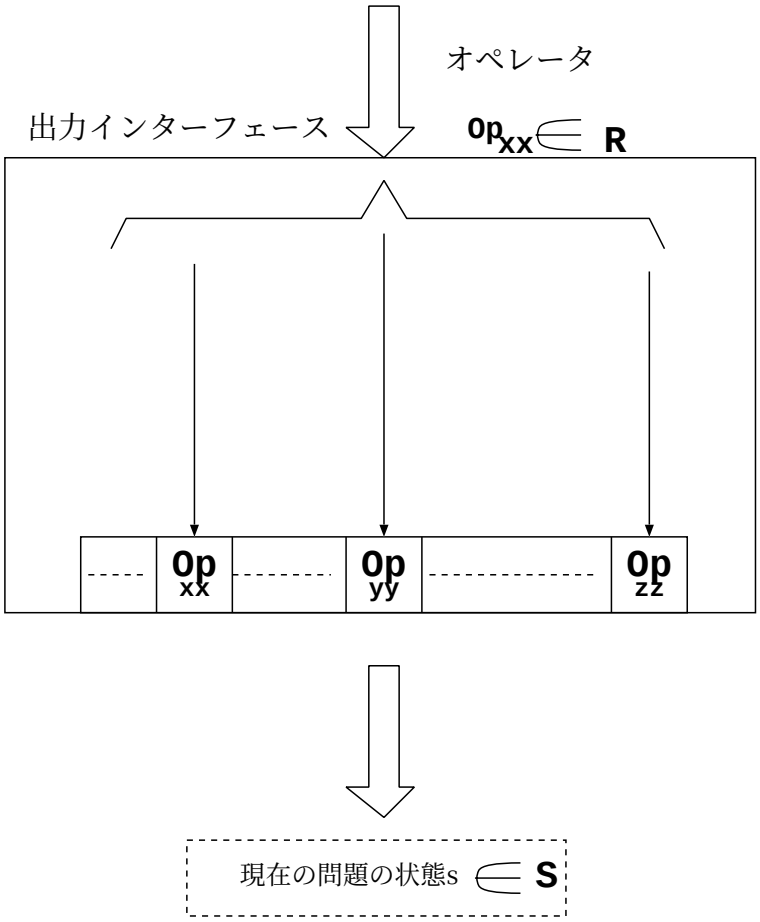


図 4.7: 出力インターフェース

4.4　システムの問題解決プロセスの分析

前節では、システム構成要素を示した。本節では、そのシステム構成要素を辿りながら本問題解決システムの処理時間を決定するための要素について分析する。

4.4.1　入力インターフェース

入力インターフェースでは問題の状態変数の値を、開発者やシステムユーザーにとって分かりやすい表現に変え内部バッファにデータをセットする。

たとえば、

- パズルやボードゲームなどでは、特定の定跡を実行するために、ボード上の特定の駒の配置や兆候など、読みとってバッファにセットする。
- ビルのエネルギーの経済的な管理においては、温度センサーデータ、湿度センサーデータなどの正規化データをバッファにセットする。
- 航空機制御においては、速度センサーデータ、高度センサーデータなどの正規化データをバッファにセットする。

従って、ここでの処理負荷は、１回当たりのデータ変換負荷と変換回数に比例して大きくなる。ただし、変換回数は入力バッファの数以上にはならない。なお、単純なデータの移し変えも本処理の範囲に含める。ここで、変換１回あたりの平均変換コストを C_{conv} とし、セットするバッファ数を N_{iBuf} とすれば、入力インターフェースに関する全負荷 C_{CONV} は、

$$C_{CONV} \leq \sum_{i=1}^{N_{iBuf}} C_{conv} \tag{4.2}$$

となる。

4.4.2 問題発見処理

問題発見処理は、入力バッファにセットされた値によって、フェーズフローデータベースのフェーズ条件を計算し、ディテクションテーブルにデータセットする。フェーズ条件計算回数と１回あたりの負荷が、本処理時間を決める重要なファクターとなる。

（１）入力

状態 $s = (p_0, p_1, \ldots, p_n)$, すなわち, p_0から p_nまでの整数のリスト。

（２）出力

ディテクションテーブル F[i][j]

（３）方法

フェーズ判定条件を順次走査する。条件に合致したフェーズは、F[i][j]=1 をセットし、その他は 0 をセットする。

（４）アルゴリズム

```
 begin
1 for i ← 0 until N_FLOW do        // N_FLOW  : フェーズフロー数
2 for j ← 0 until N_PHASE do       // N_PHASE : フェーズ数
   begin
3   F[i][j] ← 0
4   if C_{ij}(s) then F[i][j] ← 1
   end
 end
```

（５）問題発見処理の処理コスト

全フェーズフロー数 N_{FLOW},1 フローあたり、フェーズ数 N_{PHASE} とすれば、１つのフェーズ条件の平均計算コストを C_{detect}とすれば、平均問題発見コスト C_{DETECT}は、

$$C_{DETECT} \leq C_{detect} \times N_{FLOW} \times N_{PHASE} \tag{4.3}$$

4.4.3 問題選択処理

問題選択処理は、ディテクションテーブルにセットされた値によって、問題の発生部分（問題発生フェーズ）、問題の進行段階（フェーズフローの段階）、適用するオペレータの競合状況、問題の重要性（ウェート）を総合的に判断し、解決する問題を決定する。

（1）入力

ディテクションテーブル $F[i][j]$

（2）出力

問題の発生しているフローを F'[i] および 解決対象問題予約テーブル reserve[i]

（3）方法

1. ディテクションテーブル $F[i][j]$ からフェーズ・フロー毎に、問題が発生している部分問題 (即ち $F[i][j] = 1$) のフェーズ・インデックス i を抽出し、解決対象フロー予約テーブル $F'[i](i = 0, ..., N_{FLOW})$ にそれをセットする。問題が発生しなければ“ 0 ”をセットする。

2. $F'[i]$ の配列データのうち、ウェートの高いデータから 0 以上の値を有する配列データを読み出し、オペレータ番号 m を 1 から N_{op}へ走査し、部分問題毎に用意しているオペレータ集合 $R_{ij}[m]$ に対しオペレータ予約テーブル $reserve[i]$ の先約がないことを確認し、１箇所でも先約があった場合は、そのフェーズフローは解決対象から外す。すなわち、$F'[i] \leftarrow 0$。i を 0 からインクリメントしながら、$F'[i]$ をウェートの高い順に 0 以外のフェーズ番号を抽出する。

3. 競合のない場合は、オペレータ予約テーブル $reserve[i]$ に 1 をセットする。

（４）アルゴリズム

```
1    for  i  ←  1  until  N_FLOW  do        ~|
2     for  j   ←  1  until  N_PHASE  do      |  方法1
3       if ( F[i][j]=1 ) then  F'[i] ← j    _|
4    for  i  ←  1  until  N_FLOW  do                        ~|
5     k  ←  F'[i ]                                           |方法2
6     if  ( j > 0 )                                          |
7     for m ← 1  until  N_PHASE do                          _|
8     if ( Rij[m] >= 1 & reserve[m]=1 ) then  F'[i]=0 , goto skip
9     for m ← 1  until  N_OP  do                           ~|
10    if(Rij[m]>=1) reserve[m] ← 1                          |方法3
11   skip                                                  _|
```

（５）問題選択処理の処理コスト：C_{SELECT}

以上より、問題選択は、問題数、つまり$N_{FLOW} \times N_{PHASE}$によって決まるので、ウェート比較や競合解消処理を$C_{select}$とすれば、全選択コストは、$C_{SELECT}$は、

$$C_{SELECT} \leq C_{select} \times N_{FLOW} \times N_{PHASE} \tag{4.4}$$

4.4.4 問題解決処理

問題解決処理は、問題選択処理にて決定された問題に対し、その部分問題空間の状態を全オペレータで一度仮想展開する。その際生成される状態をその部分問題の評価関数で評価し最良の結果が導かれるオペレータを選択する。

（1）入力

解決予約テーブル [i] 及び 現在の状態 s

（2）出力

出力オペレータ・バッファ

（3）方法

1. i を 0 から N まで走査して、解決対象フロー予約テーブル $F'[i]$ が 1 以上のフェーズフローインデクッス i を順次抽出する。その際のフェーズ j を $j \leftarrow F'[i]$ にて合わせて抽出する。

2. jが抽出される毎に、部分問題 P_{ij}を解決する。解決は、現在の状態を仮想的に展開し、展開された状態 $s_1, s_2, ..., s_{N_{op}}$を評価関数により評価する。

3. 仮想展開された状態のうち、最良の評価が得られるものを選択し出力オペレータ・バッファ $Buf_op[i]$ にセットする。

（４）アルゴリズム

```
      begin                                          ~|
1       for  i ← 0  until  N_FLOW  do                 | 方法１
2         if  F'[i]>=1 then j ←F'[i]                  _|
                                                  // N_op : 部分問題毎のオペレータ数
3             for k ← 0 until N_op do                  ~|
4                 s_k ← Op_ijk(sk)                      |方法２
5                 v_k ← h_ij(sk )                       |
6                 if ( max(vk) > Imax )  then           _|
                                                                              -
7                  Buf_ph[i] ← j、Buf_op[i] ← k 、Imax ← v_k                  _| 方法３

      end
```

（５）問題解決処理の処理コスト：C_{SOLVE}

最終的に解決すべき問題数はN_{FLOW}以上にはならないので、各問題の利用可能なオペレータ数の平均を、N_{op}とすれば、展開された状態を評価するコストをN_{eval}とすれば、問題解決コストC_{SOLVE}は、

$$C_{SOLVE} \leq C_{eval} \times N_{op} \times N_{FLOW} \tag{4.5}$$

4.4.5 出力インターフェース処理

出力インターフェース処理は、出力バッファにデータをセットする。

- 単にコンピュータのみで処理するのであれば、画面表示バッファにデータをセットする。
- 実際に何らかにアクチュエータを作動させるのであれば、analog output チャンネル、ON/OFF 信号であれば、descreat output チャンネル用のバッファにセットすることになる

ここでの処理負荷は、１回当たりの変換負荷と変換回数によるに比例して大きくなる。ただし、変換回数は出力バッファの数以上にはならない。なお、単純なデータの移し変えも本処理の範囲に含める。ここで、変換１回あたりの平均変換コストを C_{conv} とし、セットするバッファ数を N_{oBuf} とすれば、出力インターフェースに関する全負荷 C_{CONV} は、

$$C_{CONV} \leq \sum_{i=1}^{N_{oBuf}} C_{conv} \tag{4.6}$$

となる。

4.5　提案システムと従来システムの構成の違い

従来のルールベースシステムは、悪構構造を解決するために、問題解決のための知識を“ if ～ then ～ ”という基本構造をもつ均一のプロダクションとしてデータベースに格納し、データの前件部と後件部の連鎖的な相互マッチングをとおして何らかの結論を引き出そうとするものであった。従って、推論エンジンとルールは、完全に分離していた。

我々が、今回提案する方法は、従来のルールベース同様、解決知識と処理エンジンは、分離しているという点については、同じであるが、単なるルールではなく、解決知識の中に明確な解決の方向が埋め込まれている点が大きな違いであり特徴である。

4.6　提案システムの設計法

4.6.1　従来のルールベースシステム構築法

従来のシステム構築は、一般にシステム構築ツール（市販の汎用パッケージ）をもちいて構築される場合が多い。ここでは、従来の構築手順を整理する。知識システムの開発は、

1. システムアナリシス
2. プロトタイピングアナリシス
3. 知識プログラミング
4. 保守、運用

の段階に大別される。システム構築手順を図 4.8に示す。

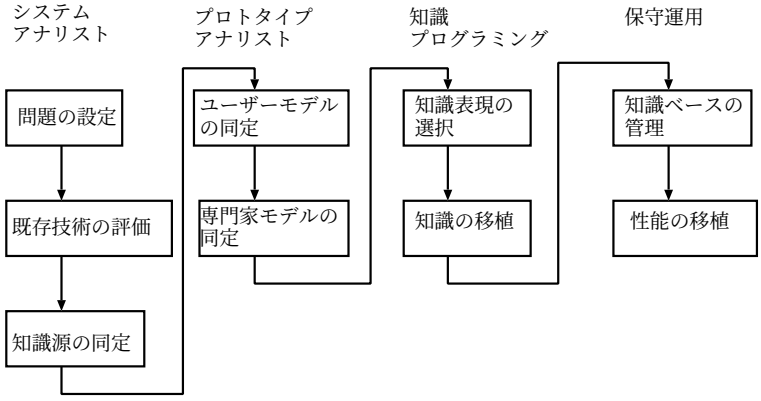


図 4.8: ルールベースシステムの構築手順

提案システムを設計するための方法について提案する。ここでは、システムの枠組みが存在しているものとして述べる。そこで、知識獲得とその構成法の観点にて、システム設計法を論じる。一般に、以下の手順で、システム設計を行う。

（1）システムアナリシス

システムの目的は、対象とすべき問題を設定し、既存技術との比較分析において、知識システム的接近の必要性を明らかにした上で、問題解決に必要な知識元の所在を同定することにある。

（a）問題の設定

知識システムの必要性および十分性をチェックした上で、対象の候補とされる問題領域から問題を切り出し定義する。

（b）既存技術の評価

従来技術（システム技術やソフトウェア技術）との比較分析を行い、知識システム的接近の妥当性や優位性を明らかにすること。問題によっては、既存形式の融合形式もありえることを示す必要がある。

（c）知識源の同定

問題解決に必要な知識が存在する知識源の所在を同定し、各種知識源に保有されている知識形態、信頼性、利用可能性などを評価すること。領域専門家は有力な知識源であるが、唯一のものではない。教科書、設計仕様書、操作手引き書、保管記録書なども有効な知識源である。

（2）プロトタイプアナリシス

プロトタイピングアナリシスの目的は、領域専門家との対話を通じて、専門家モデルを明らかにしたうえで、問題解決に必要な知識源の所在を同定することにある。

（a） 専門家モデルの同定

プロトタイプ開発を通じて、専門家がカバーする知識の範囲、知識の質と量、問題解決の基本制御ループ、推論の方法などを明きかにすること。

（b） ユーザーモデルの同定

プロトタイプ開発を通じて、ユーザーの仕様を顕在化させるとともに、ユーザーの使い方を反映したユーザーインターフェースを明らかにすること。

（３） 知識プログラミング

知識プログラミングの目的は知識の表現および知識利用の枠組みを決定し、知識源に表現および知識利用の枠組みを決定し、知識源に存在する知識を抽出し、これをベースに移植することを目的とする。

（a） 知識表現の選択

問題解決にとって有用と考えられる知識の表現および知識の利用の枠組みを検討し、最適な方式を選択すること。

(b) 知識の移植

各種知識源に、存在する知識をシステマティックに抽出し、これを利用可能な形式に変換して、知識ベースに移植すること。

（５） 保守、運用

保守運用の目的は、知識システムの評価を行い、重要な情報をドキュメント化し、さらに保守や拡張の方法を明らかにしておくことにある。

（a）システム評価

知識システムの機能的・性能的評価について、知識ベースの完全性や無矛盾性や、推論の妥当性、推論速度、メモリー容量、対話性などを評価すること。

（b）ドキュメンテーション

ユーザー向けに利用範囲、使い方、推論方式、メタ知識、高速化技術などをドキュメント化すること。

（c） 保守、拡張

知識システムの保守や拡張の方法および注意事項を、ユーザー向け／技術者向けに記述しておくこと。

4.6.2 提案システムの設計法の概要

前項では、従来のシステム開発の流れを示した。それをもとに、本提案システムの設計法について検討する。ここで、ドキュメント作成等に関する部分は除き、純粋にシステムを設計する部分にのみ焦点を当て、違いを表 4.1 の如く明確化する。従来のシステム構築ステップと異なる開発プロセスは、ゴシック文字にて示す。

表 4.2: 従来のシステムと提案手法の開発の流れの違い

大項目	中項目（従来の内容）	提案方式の作業内容
システムアナリスト	問題の設定 （提案方式の必要性・十分性の検討）	従来手法と同様。
	既存技術の評価 （他の技術との比較分析を行い、提案方式の妥当性を評価する。）	従来手法と同様。
	知識源の同定	従来手法と同様。
プロトタイプアナリシス	専門家モデルの同定 （専門家による知識範囲、制御ループ、推論方式などを決定するステップであった。）	**提案方式では、人間の解決過程が概ね分かっているものに適用するので、制御方式や推論方式を検討する必要はないが、その代わり、解決ストーリーの抽出と部分問題の抽出が必要。**
	ユーザーモデルの同定。 （ユーザーインターフェースの確立）	従来手法と同様。
知識プログラミング	知識表現の選択 （問題解決にとって有用と考える知識表現の決定）	**ルールだけでなく、問題の状態 s の定義 P およびその部分問題の知識表現を検討する。**
	知識の移植 （存在する知識をシステマティックに移植する。）	**フェーズフローの各部分問題のデータを格納する。また、ウェートを与える。**
保守運用		従来手法と同様。

4.6.3 提案システムの具体的な設計ステップ

ここでは、提案システムの開発プロセスにおいて、システムの設計に関する部分に着目して、具体的に考える。本提案手法は以下の手順でシステムへの問題解決戦略を収集するものとする。

（１）問題の設定

提案方式にて接近することの必要性・十分性の検討を行う。

（２）既存技術の評価

他の技術との比較にて妥当性を評価する

（３）知識源の同定

知識源の検討

- 専門家
- 専門家ではないが、問題を解決できる人
- マニュアル、設計書など

（４）解決ストーリーの検討

知識源が人間の場合、インタビューにより、解決にいたる過程を聞き取り調査を行い、場合によっては、実演してもらう。そして、その人がどのような項目に着目したか、それが、どのような属性のものか把握する。知識源が、文献の場合は、その過程を図解などにて把握し、解決ストリーを見出す。

（５）問題の定義

- 全問題の定義を行う 問題 $P =< S, W, R >$
  （ 全問題空間の範囲 S 、ゴール W の定義 、 オペレータ R の内容 ）
- 問題解決全般に渡って有効な、問題の状態表現 s の決定
- 部分問題の決定 $P_i =< S_i, W_i, R_i, H_i >$

部分問題の分割が分からない場合、幾つかの、解決例を集め、その共通項目を見出す。

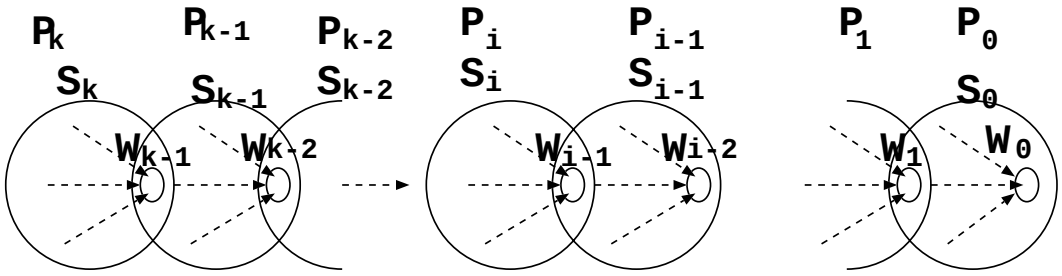
（６）可解性の検証

a. 解決ストーリー上、以下矢印の方向に解決する部分問題の系列があったとき、

$$P_k \rightarrow P_{k-1} \rightarrow \ldots \rightarrow P_0$$

であるためには、以下の条件を満足していること。

任意の問題 $P_i =< S_i, W_i, R_i >$とするとき



$$(1 \leq \forall i \leq k)(\exists S_{i-1})(\exists S_i)(W_i \subseteq S_i \& W_i \subseteq S_{i-1})$$

b. 各部分問題解決において、単調関数が定義できること。および、解決に十分なオペレータが定義されていること。

（７）処理の優先順位

フーズフローが複数存在するとき、全てについて以上の過程を実施し、最後にフロー毎ウェートを与える。

（８）フェーズフローの種類と記号化

フェーズフローを図解するための記号を表 4.3に示す。

表 4.3: フェーズフローのための関係記述記号

記号	意味
◯	SUBGOAL of MAIN FLOW
◎	GOAL of MAIN FLOW
$\oplus$	SUBGOAL of SUB-FLOW
$\odot$	GOAL of SUB-FLOW
$\Rightarrow$	DIRECTED LINK
$\rightarrow$	PRIVATE LINK
F_n	PHASE FLOW INDEX
P_n	PHASE INDEX

◯と◎は、すでに述べた、サブゴールとゴールである。どのフェーズからでも問題解決が始まる。$\oplus$は、手順に関するフェーズフローのサブゴールである。従って、最上流側に存在する隣接する$\oplus$のみで処理が開始する。$\odot$は、そのゴールである。実行が始まると評価関数に基づきオペレータを評価しながら次のフェーズへ順次シフトしていく。

Fn は、フェーズフローのインデックスである。Pn は、フェーズのインデックスである。

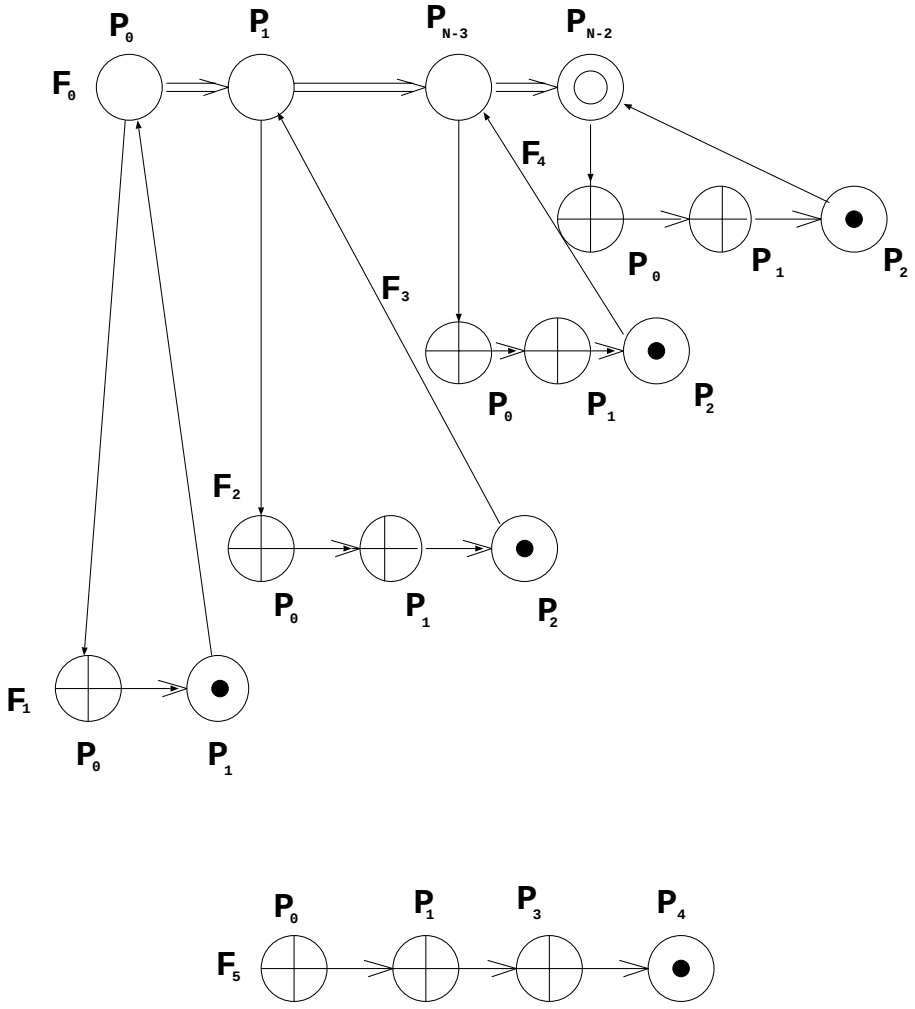


図 4.9: フェーズフロー図解例

F_0は、メインのフェーズフローである。そして、F_1〜F_4は、メインフェーズフローをサポートするサブフェーズフローである。F_5は、F_0と併存するフェーズフローであるが、独立に存在して問題解決を行なう。

これらの具体例は 5 章にて示す。

4.7　まとめ

本章では、問題解決システムを提案し情報の流れを整理した。そして、概略の処理内容について検討し、処理時間を決定する要素を明らかとした。さらに、システムの構築方法を従来のルールベースシステムの構築法と対比しながら検討し、提案システムの設計法を明確にした。次章では、本設計法に従って実際に設計を行ない評価する。

第 5 章

実験及び評価

第３章では、人間の問題解決知識をコンピュータ上に効果的に実装するための枠組みを示した。そして、４章では、システムの構成要素と設計法を示した。本章では、提案方式をプログラミングし、プロトタイプシステムを作成し、３種類の例題に適用し考え方の有効性を確認する。例題はすべて解析問題であり、内容としては、簡単なパズル、ゲーム、そして、より現実的な問題である航空機の操縦といった問題である。最後の例は、real-world 寄りの問題について実験する。ただし、パズルやゲームの例題も最適性の追求という命題にたいしては、現実の問題と同様困難な問題となる。例題の特徴は、表 5.1 のとおりである。

表 5.1: 例題の特徴

例題	タイプ	目的	環境の変化	時間遅れ	備考
基本例題１ (N^2-1 パズル)	解釈問題	最適性	なし	なし	状態空間のサイズ $6.2 \times 10^{23}(24-puzzle)$
基本例題２ (追跡ゲーム)	解釈問題	捕捉	あり	なし	ステップ数が 31 の場合、状態空間のサイズは、1.16×10^{77}
応用例題 (航空機の操縦)	診断問題 制御問題	安全性 経済性 実時間性	あり	あり	選択可能操作が 30 として、5 秒に１回の意思決定を行なったとしても、1 時間で空間サイズは 30^{720}

さて、本章では、４章にて定義した設計法にて実際にシステム構築できることを確認する。また、性能比較として、問題空間全体を対象にしたルールベースシステムを構築し比較する。ここでは、例題自体を単純化していることもあり、多くのルールを使用することは、逆に性能を落とすことにもなる。そこで、単純で、かつ、少数のルールを用い、探索機能（ヒューリスティック実時間探索）に重点を置いたシステムを構築し、できるだけ多くのノードを高速に生成検査するシステムを用いることとした。つまり、立場としては、できるだけ多くの解の候補を対象にするということである。

ここでの評価の観点は、生成される解の質にある。つまり、単に、人間が解決できる戦略を用いてシステム構築することが、有効で効率的なシステムにつながるか否かということについては、これまで明らかではなかった。一般的には、より多くの選択肢から吟味された解を生成する方が、高い質の解が選られるのではないかという想像が働く。

- **基本例題１**
  本例題では、ルールベースシステムと比較を行ない、経験的知識に基づいて解決する提案方式の結果が深い探索と同等であることを示す。
  (関連研究:[E.iida 1994][E.Iida 1998])

- **基本例題２**
  ゲームについても、基本例題１と同様の結果が得られることを示す。

- **応用例題**
  本例題は、実時間意思決定問題の一例である。ルールベースシステムは、簡単な制御に関する例題においても、適切な制御の実現は容易ではなく、一方、提案方式では、単純なフェーズフローにて制御できることを示す。
  (関連研究:[E.Iida 1998a][E.Iida 1998b][E.Iida 1998c][E.Iida 1998d])

5.1 N^2-1 パズル

5.1.1 はじめに

一般にスライディングタイルパズルは，$M \times N$のマス目を持つ矩形ボードと$(M \times N-1)$個のタイルからなる．マス目のうち１カ所はブランクと呼ばれる空位置であり，このブランクを利用して、パズルの配置を少しずつ変化させて行く．特に，$M = N$のとき，“N^2-1パズル” となる。

5.1.2 問題の定義

一辺Nマスの正方形のボード上に１から(N^2-1)までの数字が書かれた(N^2-1)枚のタイルが配置されたものを想定する．たとえば、$N = 4$のときは、図5.1のようになる。

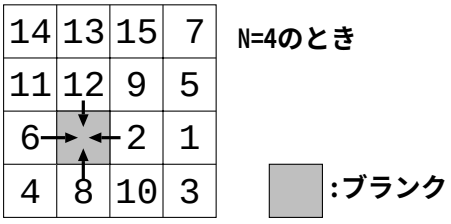


図 5.1: N^2-1 パズルの例

本パズルは、図5.2のように初期配置S_{int}およびゴール配置S_{goal}が与えられ，S_{int}に対しブランクの左右上下に隣接するタイルをブランク位置に移動する操作を繰り返し行ない、S_{goal}に再配置する問題である．

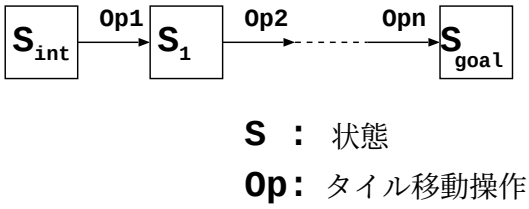


図 5.2: N^2-1 パズルの解決過程

この問題は、古くから，最適解（最短手順でS_{goal}に至る変化，または，配置変換操作の系列）を探索により求める問題として扱われてきた[Johnson et al. 1879][Shofield1967][Nilsson 1973] [Nilsson 1983] [Ratner et al. 1986][Reinefeld1993][Araya 1996]。なお、以下、ゴール配置における個々のタイルまたはブランクの位置を、そのタイルまたはブランクの “ホーム”

と呼ぶことにする。また、各タイルをホーム位置に移動させ揃えることを、単に、“解決する” ということにする。

5.1.3 問題解決システムの設計（その１）

４章にて示した設計法により設計を進める。

（１） 提案方式の必要性・十分性

本例題は、実用システムでないので必要性や十分性を説明するのは難しい。しかし、本問題を手続き型言語で記述する場合、プログラムが長くなり分かりにくい形になる。また、結果的に、あらゆる状態に対応して可解となるシステムを作るには、状態空間の概念は避けて通れないだろう。一方、従来のルールベースシステムにて、構築しようとすると、デバッグに時間がかかり、プロトタイプとして、まず解けるシステムを作るのであれば、今回提案する手法のアプローチは十分適しているものと考えられる。

（２） 他の技術の評価

N^2-1 パズルは、しばしば、探索手法の例題として用いられてきたが、深さ優先探索や幅優先探索などの力づくの探索手法では、すぐに計算量の組み合わせ爆発を招く．そして，それに対処するために発見的知識を用いて枝刈りを行う A^*[Hart 1968] や反復深化 $A^*(IDA^*)$[Kolf 1985a] などのヒューリスティック探索手法が提案されるようになったが，これらの方法でも計算量が問題サイズ（状態数）に対して指数的に増大してしまう．近年では，実時間探索が注目されるようになってきた。これは、最適性は保証されないものの限られた時間でより良い解を見つける方法であり、今後、動的環境下でのプランニングまたは意思決定のツールとして期待されている [Kitamura 1996][Kolf 1990][Ishida 1992]。この実時間探索を解決エンジンとするルールベースシステムは後で、比較実験に用いる。

（３）知識源の検討

ここでは、知識源となる解決戦略を記した文献が存在しないので、人間の戦略を用いる。そこで、２章の事例における被験者の戦略を使用する。

（４）解決ストーリーの検討

ここでは、２章にて既に示した戦略のうち、以下の戦略を用いることとする。

- 解決ストーリー
  最下段から順に解決する。特に番号の大きいものから順に解決するものとし、解決した段は基本的に動かさない。

（５）問題の定義

(a) 状態の知識表現

$$s = (X_1, X_2, ..., X_m)$$

$$m = N^2$$

ただし、N は、パズル１辺のマス目の数

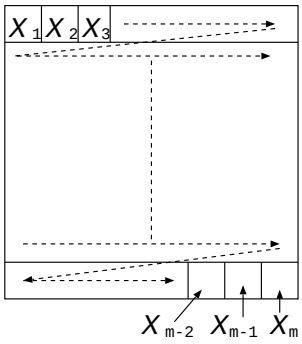


図 5.3: $N^2 - 1$ パズルの知識表現

(b) 問題空間 S

$$S = \{(X_1, X_2, \ldots, X_m) | X_i = 0 \sim N^2 - 1 \text{ の整数} \quad \& \quad (\forall i)(\forall j)(X_i \neq X_j)\}$$

(c) ゴール空間 W

$$W = \{(0, 1, 2, ..., N^2 - 1)\}$$

(d) オペレータ R

R は、以下の通り。最も原始的なオペレータの一回の操作は、上下左右いずれかへのブランクの１マス分の移動である。しかし、結果的みると、大きく３種類の動きに別れる。盤面への展開イメージにて示す。

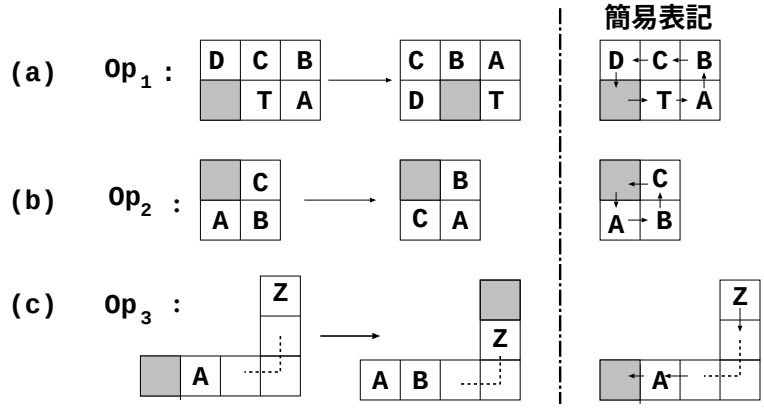


図 5.4: オペレータの基本パターン

評価関数 H

一般に、N^2-1 パズルをヒューリスティック探索によって解決する場合、個々のタイルのホームへの移動のコストは、タイルの現在位置とホーム位置とのマンハッタン距離をもとに評価している。これは、大局的に、この値が小さいほど手数が短くなるためであり、我々の方式においても、この性質をヒューリスティックスに使用する。具体的には、タイル T_i のマンハッタン距離 D_i は、タイル位置 (X_{Ti}, Y_{Ti})、ホーム位置 (X_{Hi}, Y_{Hi}) とすれば、

$$D_i = |X_{Hi} - X_{Ti}| + |Y_{Hi} - Y_{Ti}| \tag{5.1}$$

となる。

（６） 可解性の検証

(a) 指定タイルを任意の位置へ移動することは可能か

被験者の観察とインタビューした結果わかった、任意のタイルを指定位置へ移動することの基本戦略は、以下の手順であった。

- 指定タイルにブランクを隣接させる。
- 指定タイルの進行方向へブランクを隣接のまま移動させる。
- ブランクとなった進行方向へ指定タイルを１つ進める。
- これを繰り返す。

これを、図解すると図 5.5のとおりである。

本図は、一例として、タイル「１４」を「１５」の左隣に移動させるプロセスを示している。フェーズデータテーブルは、サブゴールを示しており、「１６」は任意 (“don’t care”) のタイルを意味する。問題の初期状態から、プリミティブなオペレータを適用しながら生成される状態を評価関数により評価しながら「１４」が最もサブゴールに近付くためのオペレータを選択する。

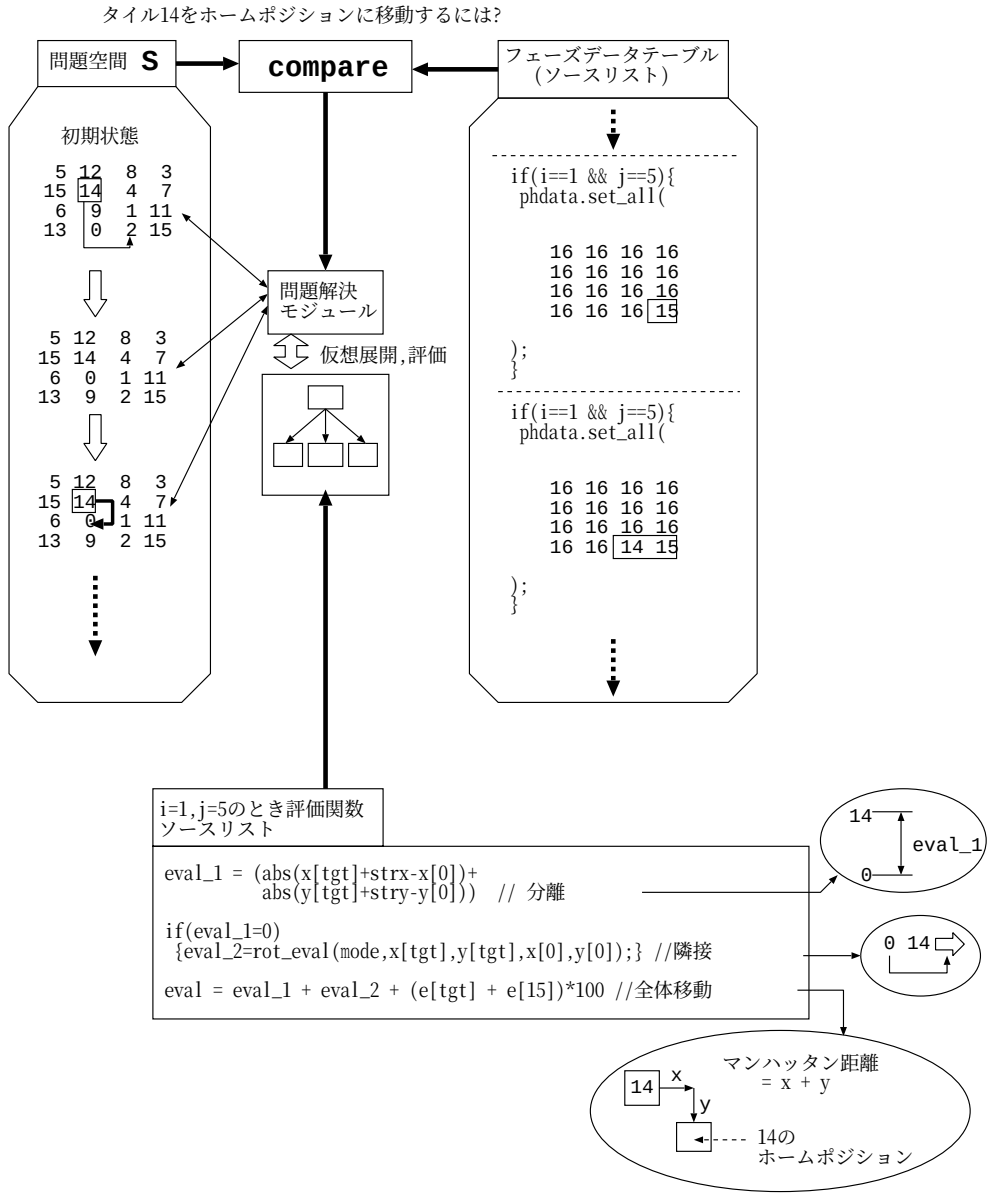


図 5.5: 可解性（指定タイルの移動）

(b) 問題解決が後戻りしないための戦略

問題解決において、解決した部分を後戻りせず、確実に前進することは重要なことである。本例題のおける、後戻り禁止は、図 5.6のように実現されている。

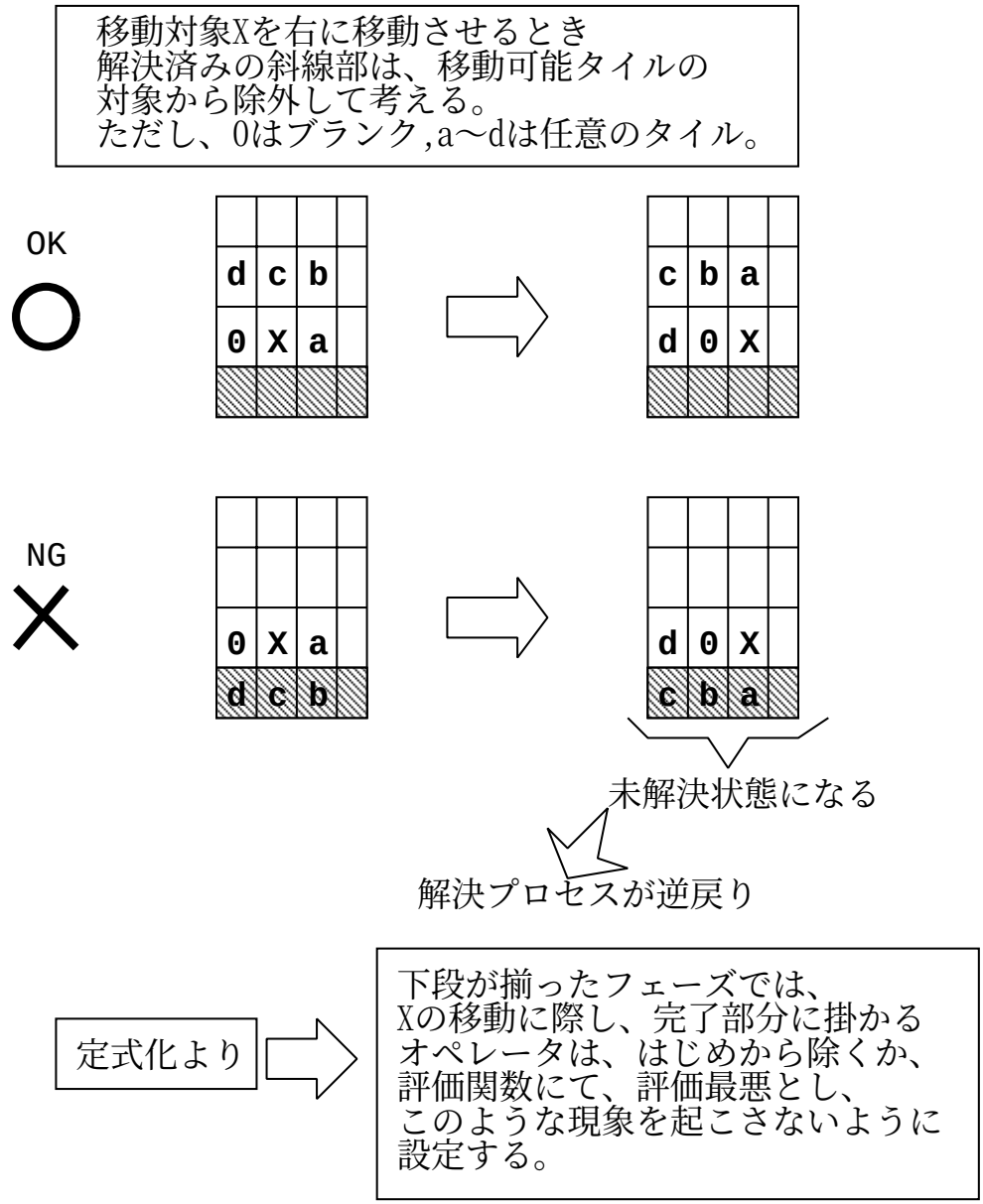


図 5.6: 可解性（後戻りしないための戦略）

（7）全フェーズフロー

全体のフェーズフローは図 5.7 ようになる。
処理の優先順位は $w_1 > w_2 > w_3 > w_4 > w_0$ の順である。

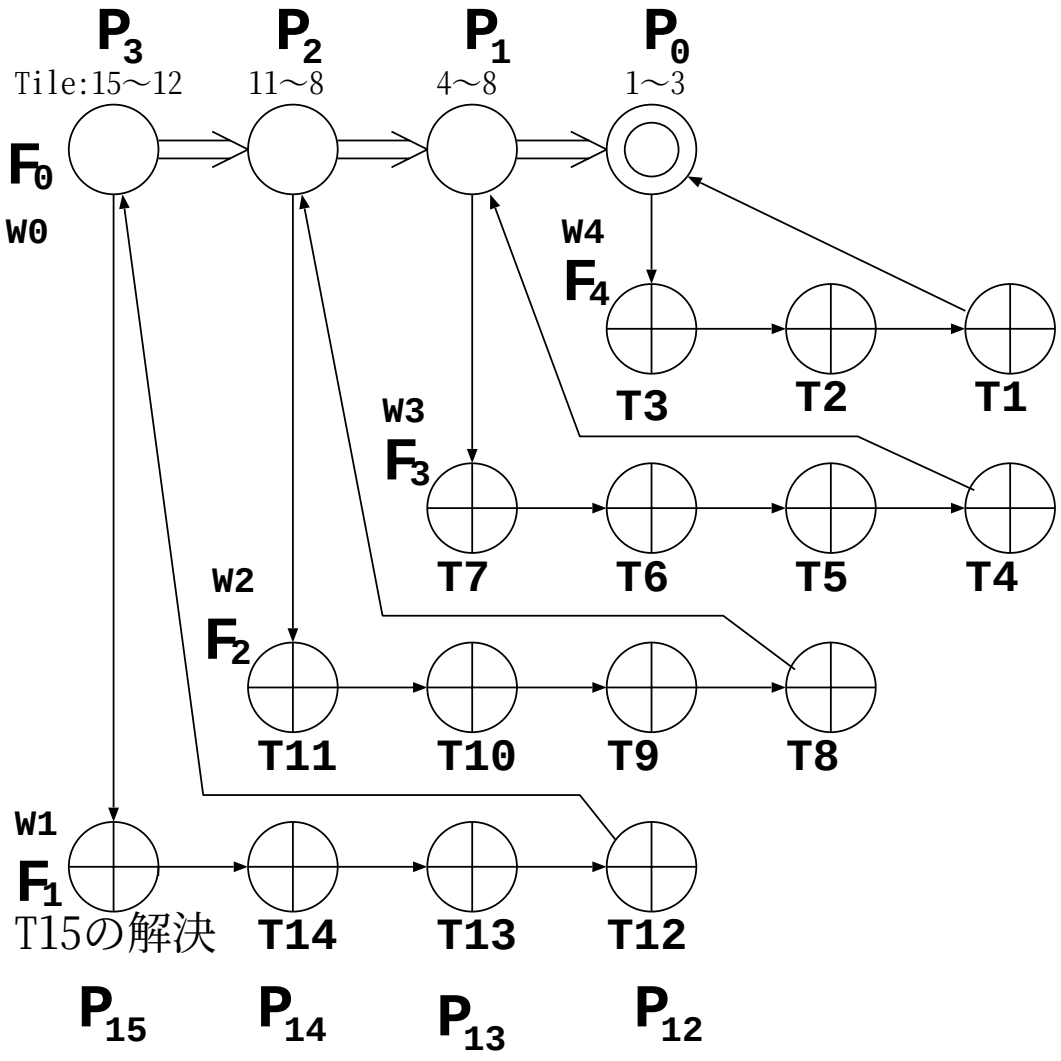


図 5.7: $N^2 - 1$ フェーズフロー

5.1.4　問題解決システムの設計（その２）

5.2.3 にて示した設計法を若干効率化し拡張する。

（１） 提案方式の必要性・十分性

5.2.3 に同様。

（２） 他の技術の評価

5.2.3 に同様。

（３）知識源の検討

5.2.3 の戦略知識をもとに効率化。

（４）解決ストーリーの検討

ここでは、便宜的に、パズルの形状に着目して、帰納的に考えることとする。まず、解決可能な状態を考えそこから一般化する。

- $N = 2$ のときの $N^2 - 1$ パズルは、状態数が４とおりであり、有限時間で明らかに解決であきる。
- 一般に N × N のパズルのサイズを１つ大きくするには、幾何学的に見て L 字形の領域を追加し、(N+1) × (N+1) の問題に拡張することが可能である。

そこで、解決の場合は、与えられた問題から、順次、L 字形領域を解決し、拡張とは逆にスケールダウンしながら全体を解決する。

（５）問題の定義

5.2.3 に同様。

（６） 可解性の検証

5.2.3 に同様。

（７）全フェーズフロー

まず、大局的フェーズフローと局所的フェーズフローについて考える。

- 大局的フェーズフロー：F_0

  1 回あたりのスケールダウンに際して、４とおりの L 字形が考えられる。そこで、これらのうち解決コストの評価値（後で示す）が、最小の領域から解決するものとし、サイズ $N \times N$から 2×2 までのフェーズとして定義する。ここでのオペレータは “L 字型” を採用する。

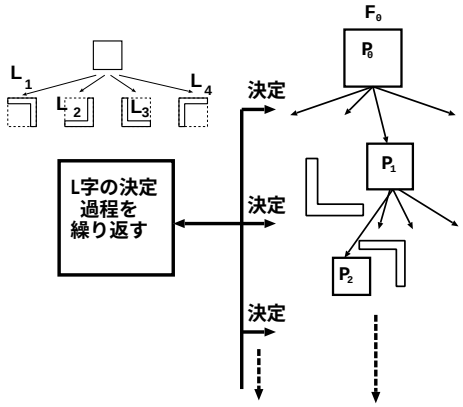


図 5.8: サブゴールの選択

- 局所的フェーズフロー：$F_1, .., F_{N-1}$

  F_0にて、解決対象の L 字形が決まれば、以下のように時計回りにゴールを指定しながらその該当タイルを順次移動する。移動方法としては、後で示す基本オペレータを用い、移動を実現する。移動方針としては、タイルをホームポジションの前方に直線的に移動し、次にホームポジションに向かって移動する。図5.9の場合T_iは、X 軸と並行に移動し、次にホームへ向かって Y 軸と並行に進む。

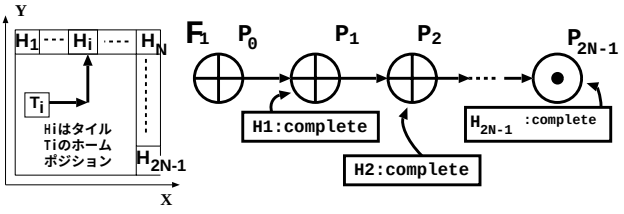


図 5.9: サブゴールの定義

処理の優先順位は $w_1 > w_2 > w_3 > w4 >, \ldots, w_{N-2} > w_0$の順である。

- フェーズフローを図 5.10に図解する。

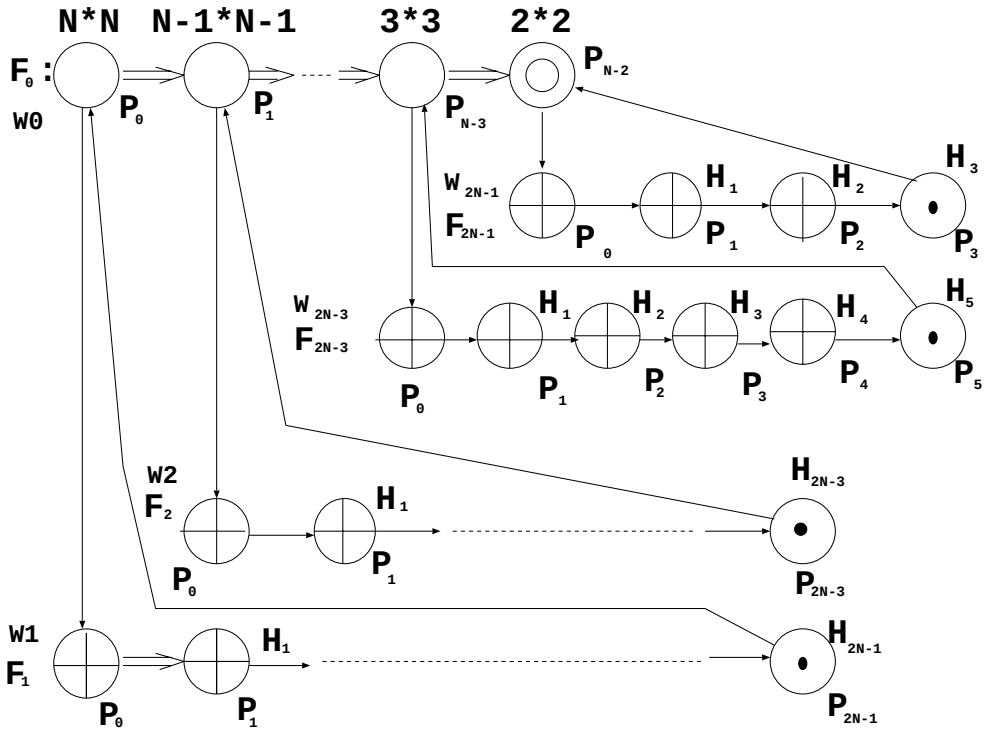


図 5.10: $N^2 - 1$ パズルのフェーズフロー（その２）

5.1.5 実験及び結果

（１）解の質に関する検討

ここでは、従来方式のルールベースのよるアプローチと提案方式によるアプローチの比較実験を行う。前者は、実時間探索機能を問題解決のエンジンとして構築され、serach horizon(先読み深さ) の設定を大きくするほど解の質が向上する性質をもつ。提案方式については、5.2.3 と 5.2.4 にて設計した２種類のフェーズフローに基づいて、問題解決システムを構築した。使用するマシンは PC であり、CPU（Pentium）のクロックは 133MHz のものを使用した。

問題サイズとして手ごろな１５パズルを中心に比較実験を行った。本例題は [Kolf 1985a] の 100 題の例題と同様のもであり、最適解が分かっている。比較システムの serach horizon は 10 以上では解の質は向上するものの計算時間がかかりすぎるため 5,8 及び 10 にて比較することとした。また、本報告中の計算時間は、使用マシンでも大きく変わるので参考値である。

図 5.11は解の長さと計算時間と平均値を示している。

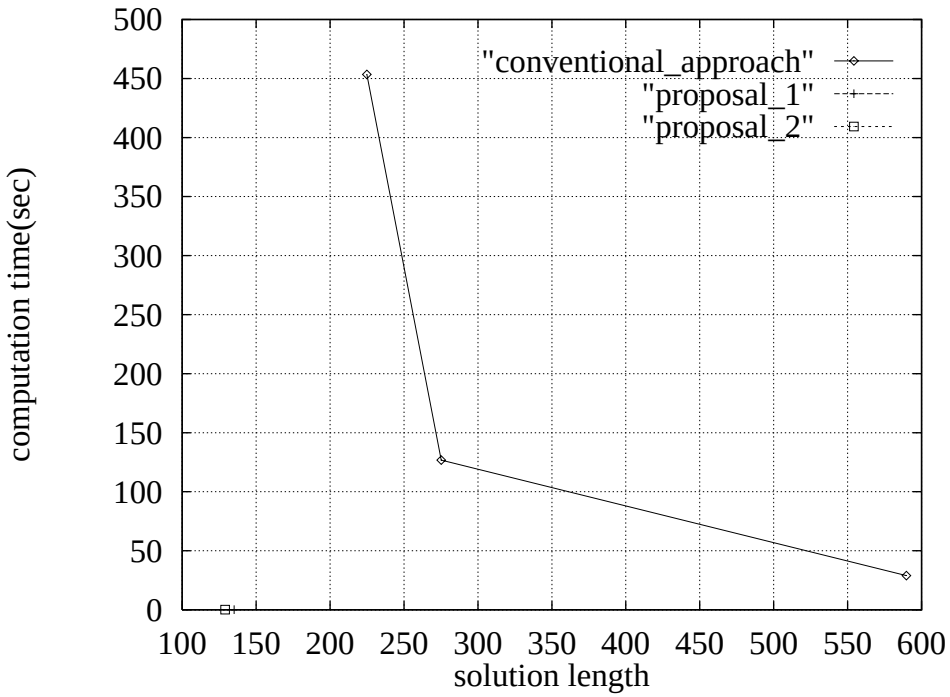


図 5.11: 解の長さと平均計算時間

図5.12は、解の長さと計算時間に関する分布図である。search horizon = 5 の分布が比較として示されている。上段は設計（その１）であり、下段は設計（その２）である。

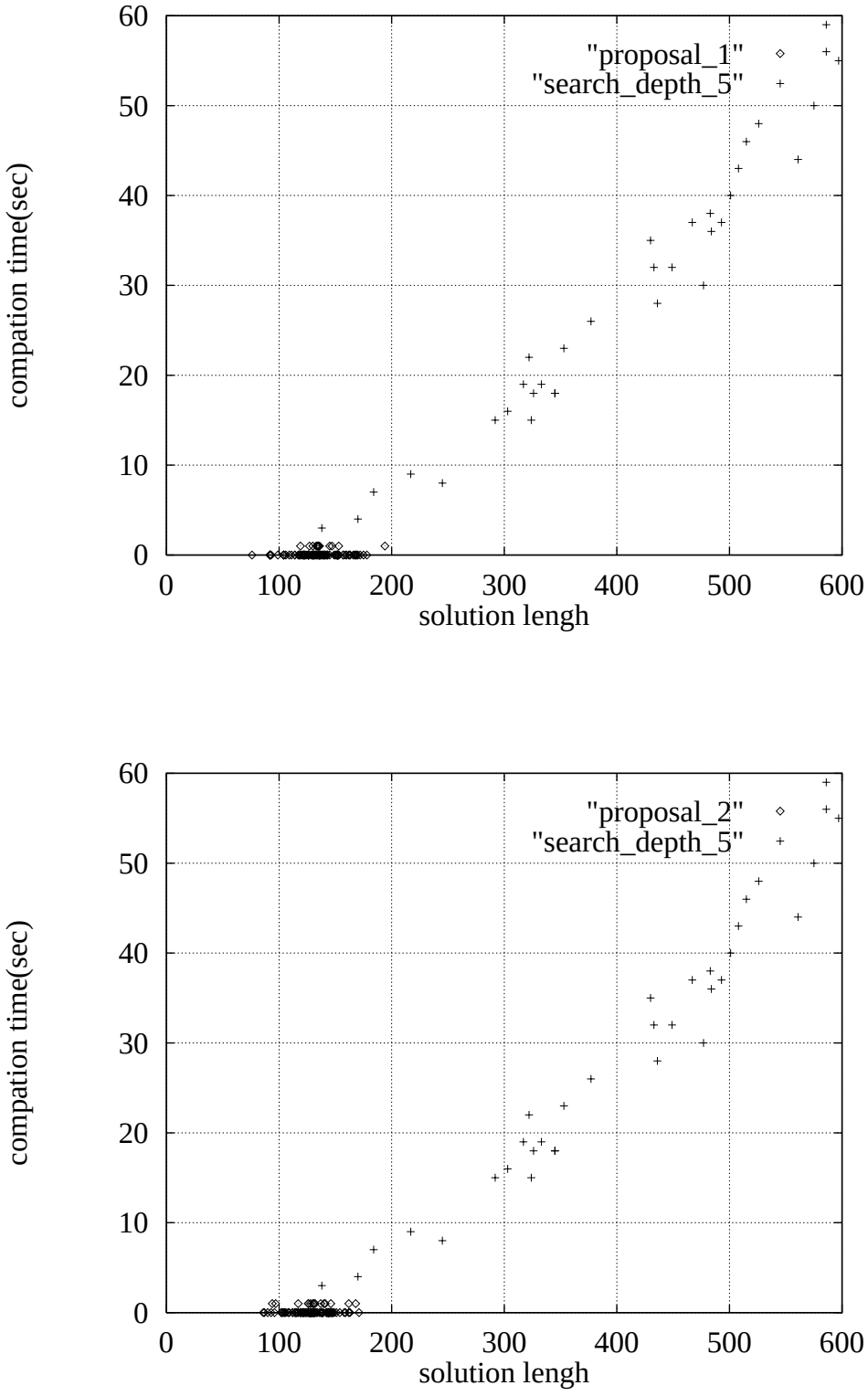


図 5.12: 解の長さと計算時間の分布

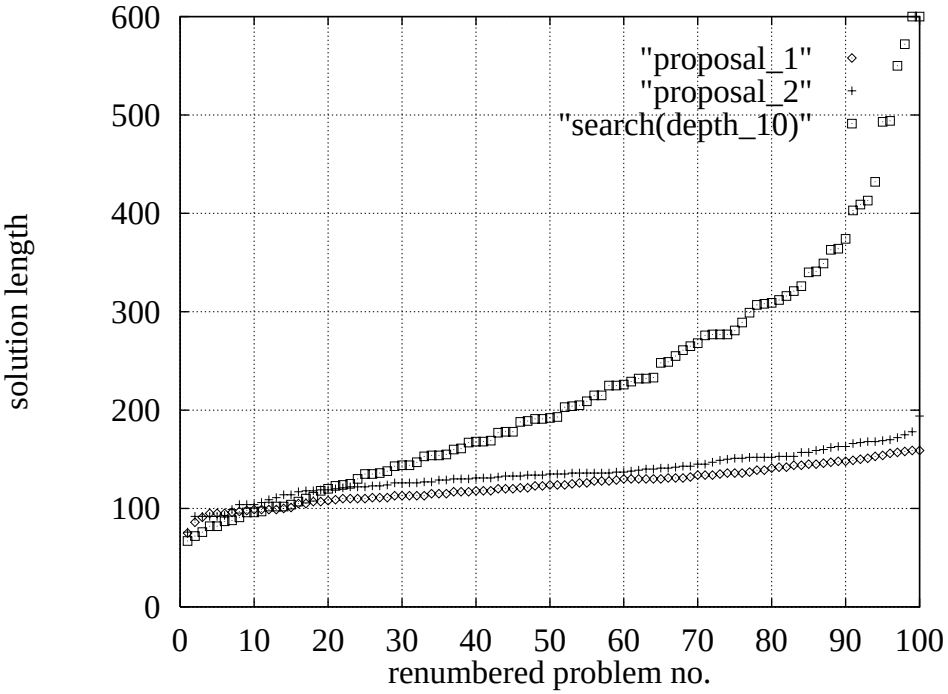


図 5.13: 解の長さの分布

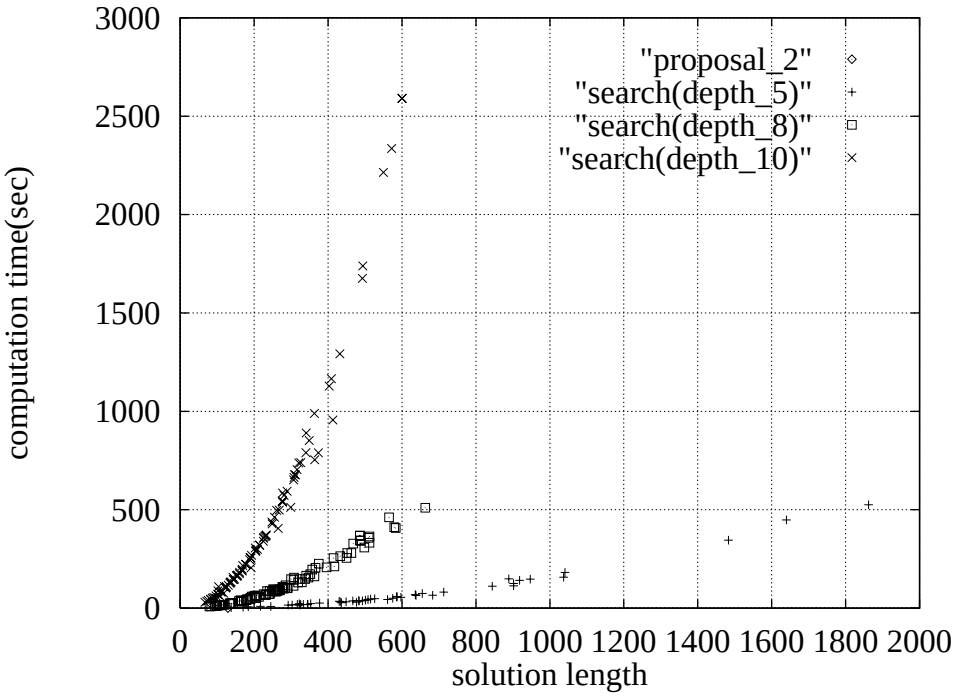


図 5.14: search horizon の違いにおける解の長さと計算時間の分布

5.1.6　評価

本実験から 15 パズルでは、提案方式の戦略を、下から揃える、または、上から揃えるという、かなりオーソドックスな戦略と解決段階で状況を考慮しながら揃える方向を変える戦略において大きな差はなかった。通常、後者の方が大局的にみて効率的であろうという予想はつくと思われるがそれが、平均で見た場合、５%程度の改善に終わった。これは、結局どちらも人間の解決戦略であり、敢えて、極端に非効率的な戦略を取らない限り、大きな差は生じないものと思われる。一方、従来方式では、全問題空間を考慮しながら解の導出を行っているにもかかわらず、良質の解を求めるのが容易ではなかった。たとえば、場合によっては、１時間近く計算しながらも人間の戦略に基づく解に及ばない。

全般的に、人間の持つ解決戦略は、有効であり、その問題解決力のポテンシャルは高いものと言えると同時に、本例題に関し、そのような戦略を組み込んだ提案方式に基づく問題解決システムも有効であることがわかった。

5.1.7　まとめ

ここで、用いた、フェーズフローの拡張は、２４パズル程度であれば、人間においても、目分量により簡単に実行できる、さらに、大きくても、目標とするタイルを盤面から発見することができれば、機械的に解決できるものと思われる。しかし、問題のサイズが大きくなると従来方式では、指数関数的に計算時間が増大し、更に、人間の戦略の優位性が際立ってくる。付録にて、Korf による実験データとの比較分析を掲載しているので参照して頂きたい。

5.2 追跡ゲーム (Pursuit Game)

5.2.1 はじめに

追跡ゲームは、主に、マルチエージェントの協調作業の研究例題として用いられた。[Benda et al. 1985] に定義によれば、碁盤のような格子平面上に、青い４つのエージェントが赤い１つのエージェントを囲い込むことをことをゲームのゴールとしている。各エージェントは無限に広がる２次元のグリッド上を動く。全ての青いエージェントと赤いエージェントの１つの動きがサイクルをなす。

1. 赤いエージェントの行動規則は以下のとおりである。
   (a) 水平または垂直方向に１ステップ動くことができる。またその位置に留まることも可能である。
   (b) 青いエージェントがいるグリッドには移動できない。
2. 青いエージェントの行動規則
   (a) 水平または垂直方向に隣接したグリッドの１つに移動できる。
   (b) 他の青いエージェントがいる場所にも移動できる。

しかし、本問題の必勝法は下図のように赤エージェントの上下左右の延長上に存在したするようにしながら徐々に詰めればよい。

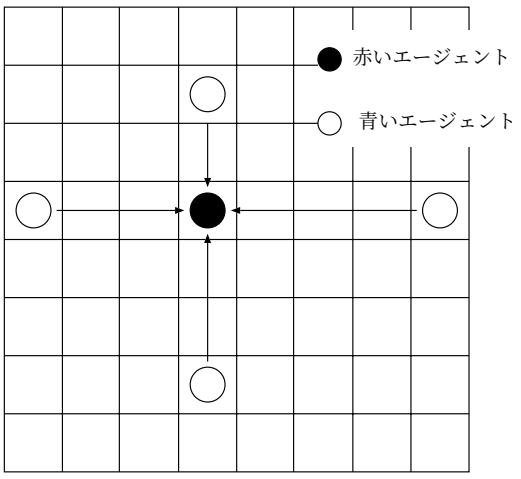


図 5.15: 追跡ゲーム必勝パターン

5.2.2 問題の定義

本研究におけるの、問題の定義は、状態空間に制約を与えると同時に、動きに制約を与えることで、適度な難易度を与える。共通な項目は青い４つのエージェントと赤い１つのエージェントが存在すること。また、異なる項目は、８×８の有限な格子平面上にそれぞれが存在していることである。赤いエージェントの動きを封じ込めることをゲームのゴールとする。したがって、コーナーに追いつめることが必要である。 行動規則は以下のとおりである。図5.16参照。

1. 赤いエージェント（●）
   (a) 周囲８方位について１ステップで移動できる。
   (b) その他は、[Benda et al. 1985] と同様である。
2. 青いエージェント（○）
   (a) １から４の通し番号をもち、青いエージェントの移動サイクルに際して、１から４の順または４から１の順のどちらかを選択し、順次動かす。
   (b) 青、赤を問わず、他のエージェントの存在するマスには移動できない。
   (c) その他は、[Benda et al. 1985] と同様である。

本ルールのもとでは、配置次第では、仲間のエージェントの存在が障害となり、追いつめることができない。

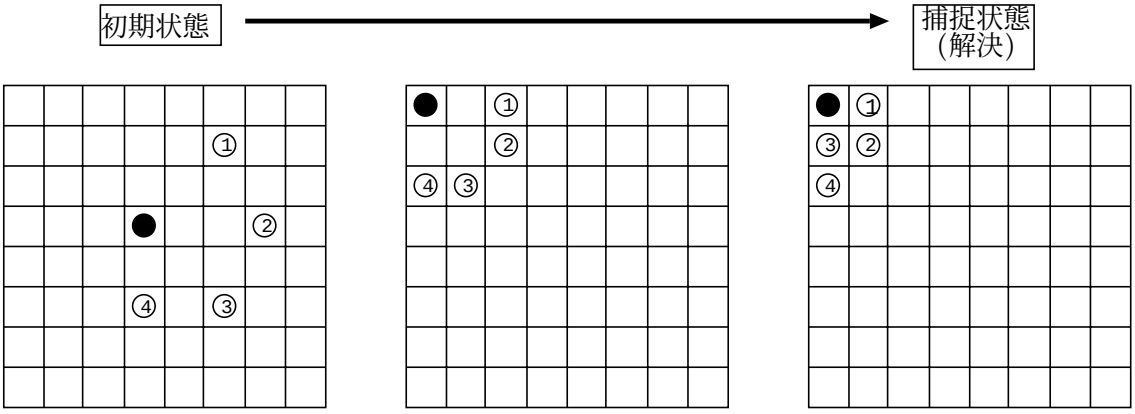


図 5.16: 追跡ゲーム解決例

5.2.3 問題解決システムの設計（その１）

４章にて示した設計法により、本問題に関する問題解決システムの設計を行う。

（１）提案方式の必要性・十分性

本例題も、前例題と同様、実用システムではないので、必要性や十分性を説明するのは容易なことではない。しかしながら、本問題はゲームであり、状況は逃亡エージェント次第では、思うように解決できない。しかし、結果的に各エージェントが１回の移動で動ける範囲も動けるバリーションも有限である。問題空間の概念を強化した本提案方式は有効であろうと考えられる。また、本問題に始めて取り組んだ小学生が、容易に問題解決出来たので、何らかの解決ストーリーが有りそうである。

（２） 他の方式の評価

従来のルールベースの枠組みにおいては、ゲーム木探索が問題解決エンジンとなる問題であろう。もしも、明確な解決ストーリーが存在しそうもない問題ならば、ゲーム木探索は強力なツールであろう。これは、あとで、比較実験に用いる。

（３） 知識源の検討

知識源としては、解決戦略を記した文献が存在しないので、人間の戦略を用いることとする。そこで、２章の事例における被験者の戦略を使用する。

（４） 解決ストーリーの検討

ここでは、２章にて示した戦略のうち、以下の戦略を用いることとする。

- 解決ストーリー
  出来るだけ早期に、以下のようなＬ字型フォーメーションを構成する。逃亡エージェントを後方に逃がさないようにしながら、密着してしていく。

（５）問題の定義

(a) 状態の知識表現

本問題の状態 $s \in S$は、以下のように表現するものとする。

$$s = (x_1, x_2, ..., x_i, .., x_{64})$$

図 5.17は、ゲームのある局面に対する知識表現の一例である。$x_i = \{j| -1 \leqq j \leqq 5$ なる整数 $\}$ で、-1 は赤エージェント、１から 4 は青エージェント、0 はブランクを示す。

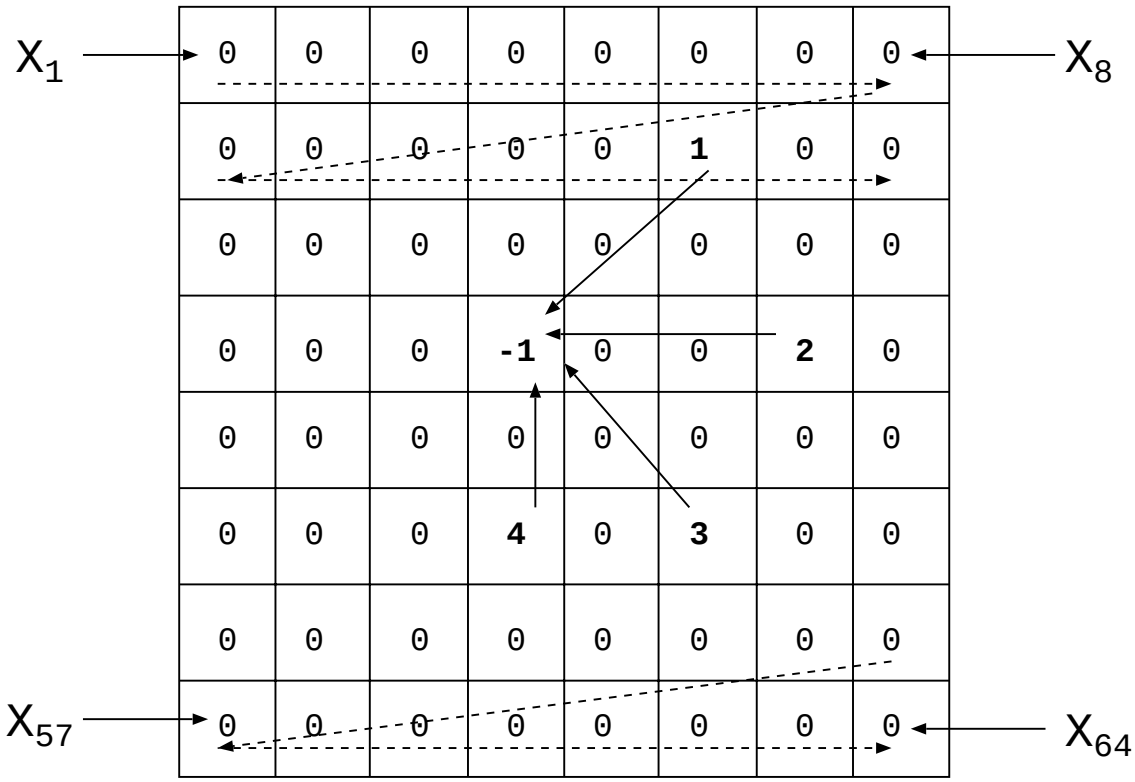


図 5.17: 追跡ゲームの知識表現

(b) **問題空間 S**

$$S = \{s|s \text{ は } 59 \text{ 個の } 0 \text{ と } -1, 1, 2, 3 \text{ 及び } 4 \text{ からなる整数からなるリスト }\}$$

(c) **ゴール空間 W**

状態 s においてエージェント i の移動可能箇所の数をカウントする関数 $count_{move}(s, i)$ を考えるとき、

$$W = \{s|s \in S, count_{move}(s, -1) = 0\}$$

（d）オペレータ R

R は、図 5.18のとおり、各エージェントが隣接する８方位のマスへの移動であり、各エージェントとも同様。

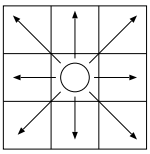


図 5.18: エージェント移動パターン (8 方位)

状態 s において、s の x 番目の要素に y をセットする関数 set(s, x, y) とする。いま、状態 s の要素のうち、エージェント i の該当要素のアドレス（s を構成するリスト上の位置番号）を Ad(i) とし、また、盤面上でそれに隣接するマスのうちの１つのアドレスを Ne(i) とする。このとき、

エージェント i を消去するオペレータは、$Op_1 : s = set(s, Ad(i), 0)$

エージェント i を盤上に置くオペレータは、$Op_2 : s = set(s, Ne(i), i)$

と書け、従って、移動のためのオペレータは Op_{move}は、

$$Op_{move} = Op_2 \cdot Op_1$$

となる。

（e）評価関数

逃亡エージェントにおけるの８方位の障害物または盤面エッジまでの見通し距離の和。本評価値が 0 のとき、ゴール局面 (逃亡エージェントは逃げ場無し) である。

（６）可解性の検証

（４）にて示した、解決ストーリーを順を追って検討する。

（a） L 字フォーメーションを必ず組めるか

この命題については、可能であり、図 5.19に図解する。

（b） L 字フォーメーションにて、逃亡エージェントに近づけば、必ずコーナーに追い詰めることが可能か

この命題についても、可能であり、図 5.16に図解する。

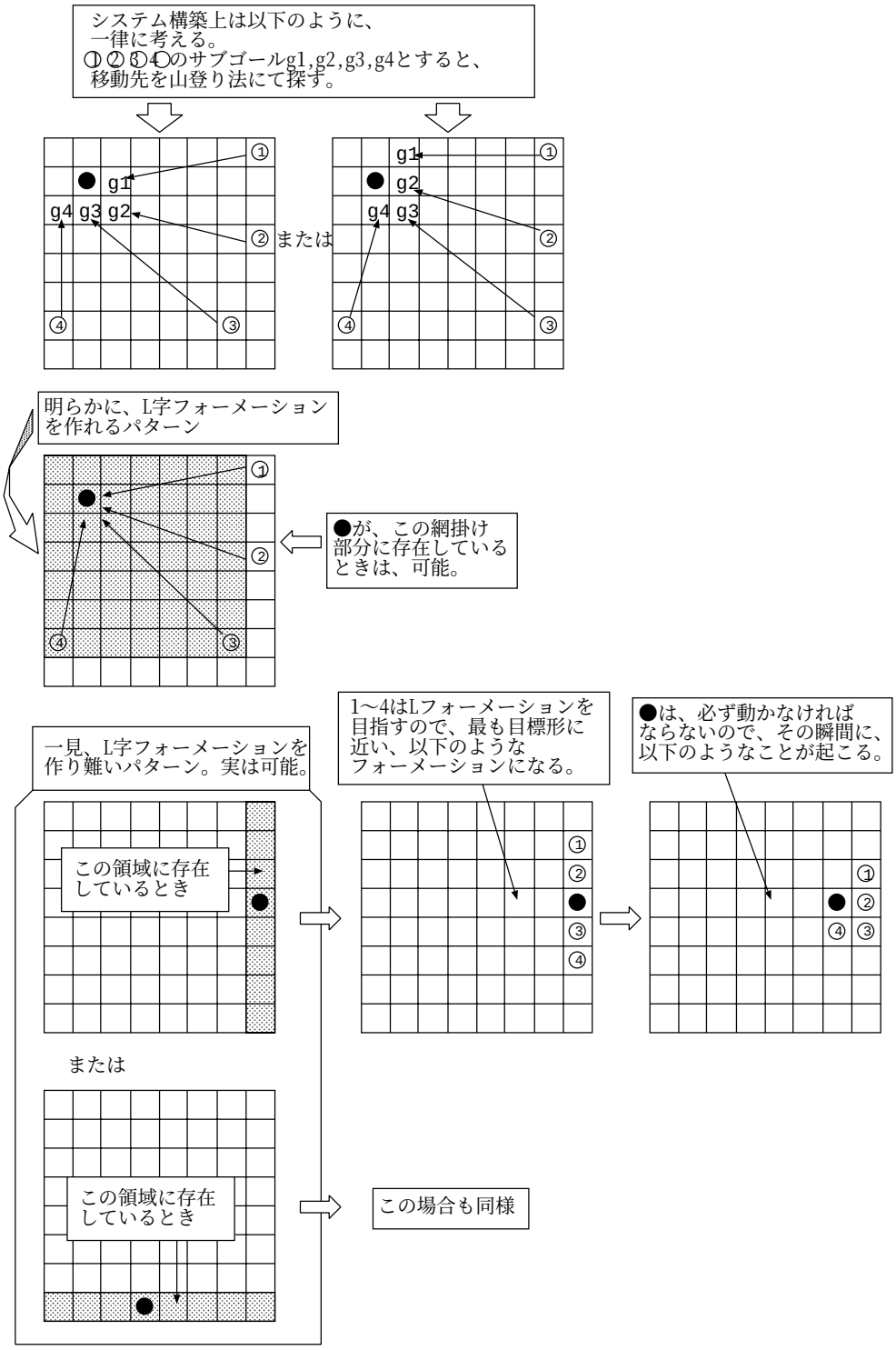


図 5.19: L 字フォーメーションの構築過程

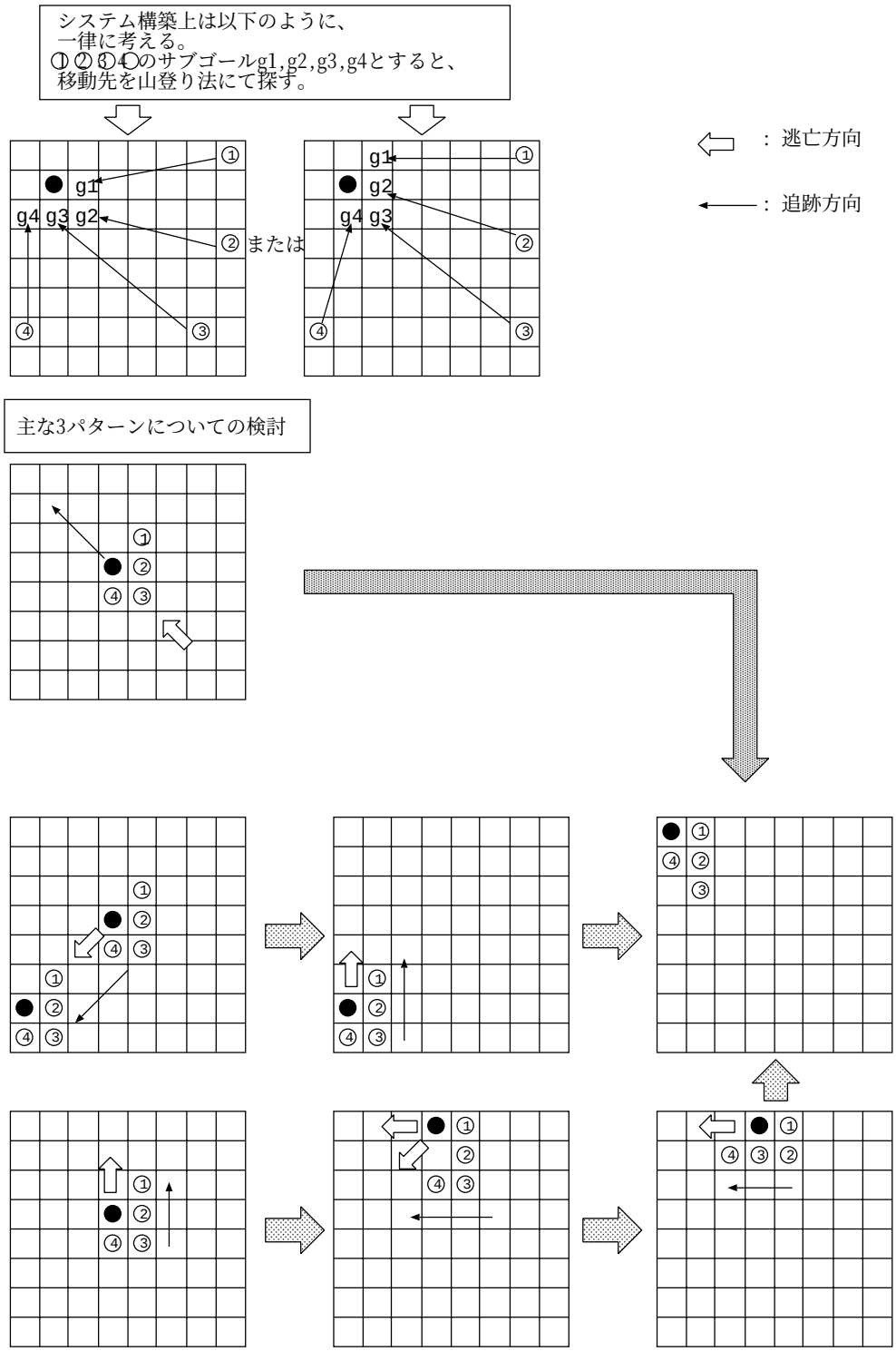

システム構築上は以下のように、
一律に考える。
①②③④のサブゴールg1,g2,g3,g4とすると、
移動先を山登り法にて探す。
g1
g4 g3 g2
または
g1
g2
g4 g3
：逃亡方向
：追跡方向
主な3パターンについての検討

図 5.20: 追い詰め戦略

（７）全フェーズフロー

図 5.21に図解する。

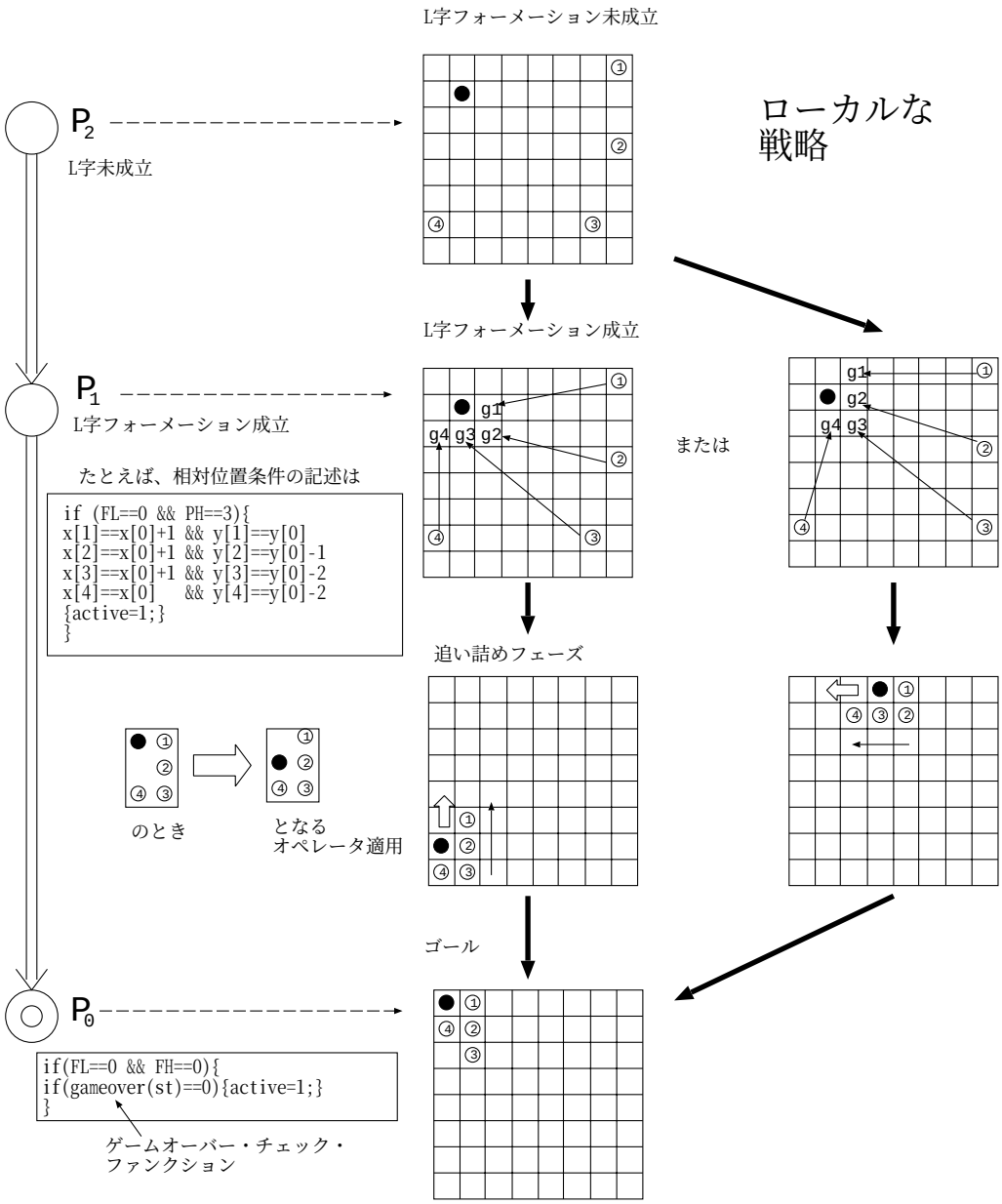


図 5.21: 追跡ゲームのフェーズフロー

5.2.4　問題解決システムの設計（その２）

４章にて示した設計法により、本問題に関する問題解決システムの設計を行う。

（1）提案方式の必要性・十分性

5.2.3 に同様。

（５） 他の方式の評価

5.2.3 に同じ。

（６） 知識源の検討

5.2.3 の戦略知識をもとに効率化。

（７） 解決ストーリーの検討

ここでは、便宜的に、ゲームの対称性に着目して、効率化を行う。

- 解決ストーリー
  下図のように、逃亡エージェントの存在領域を予め４分割し、それに応じて L 字フォーメーションの向きが変えることとした。

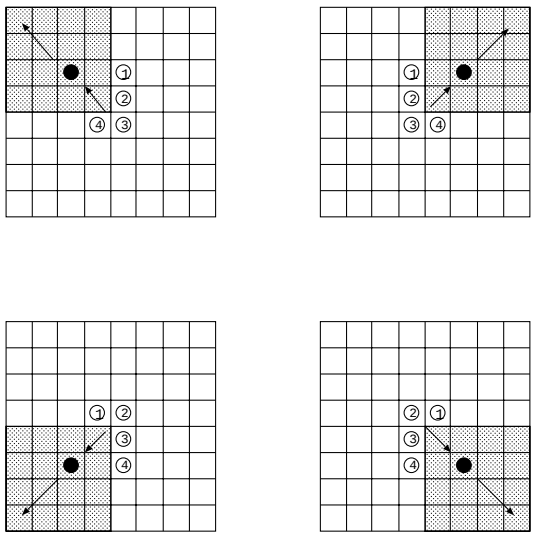


図 5.22: 解決ストーリー

（５）問題の定義

5.2.3 に同様。

（６）可解性の検証

5.2.3 に同様。

（８） 全フェーズフロー

下図による。基本的、問題解決原理は前項による。各エリアには重複部分がないので、逃亡エージェントの存在領域にて戦略が変わる。

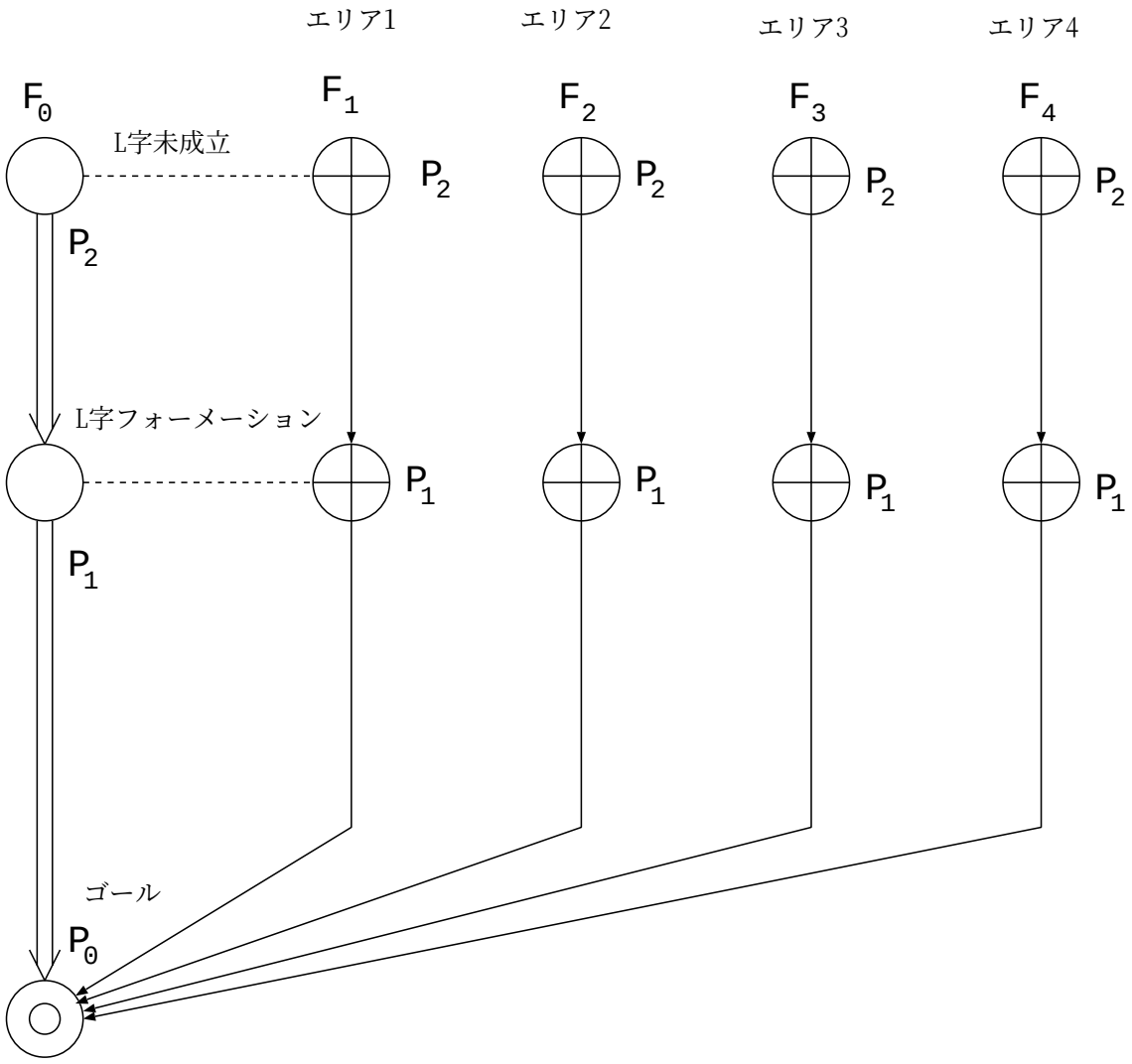


図 5.23: 追跡ゲームのフェーズフロー（その２）

5.2.5 実験及び結果

（１）解の質に関する検討

ここでは、先のパズルのときと同様に、従来方式であるルールベースのよるアプローチと提案方式によるアプローチの比較実験を行う。前者は、実時間探索機能を問題解決のエンジンとして構築し、search horizon(先読み深さ) の設定を大きくするほど解の質が向上する性質をもつ。提案方式については、5.2.3 と 5.2.4 にて設計した２種類のフェーズフローに基づいて、問題解決システムを構築した。使用するマシンは PC であり、CPU（Pentium）のクロックは 133MHz のものを使用した。

- 実験条件
  本実験条件として、まず、逃亡エージェントの行動は、自動的に行わせる。逃亡エージェントを中央付近に存在するようプログラムされている。万一、コーナーに追い込まれても、中央位置までのマンハッタン距離がもっとも小さくさなる位置へ移動するものである。また、ルールベースにおいては、先読み深さ最大１から７までの範囲にて実行した。追跡側は、逃亡側の評価関数を予め与えているものとする。このようにすると一見、従来、手法の方が有利であるように思えるが、実際どうであろうか。

図 5.24は、解の長さと計算時間の平均値を 2 種類の提案方式と従来方式について示している。

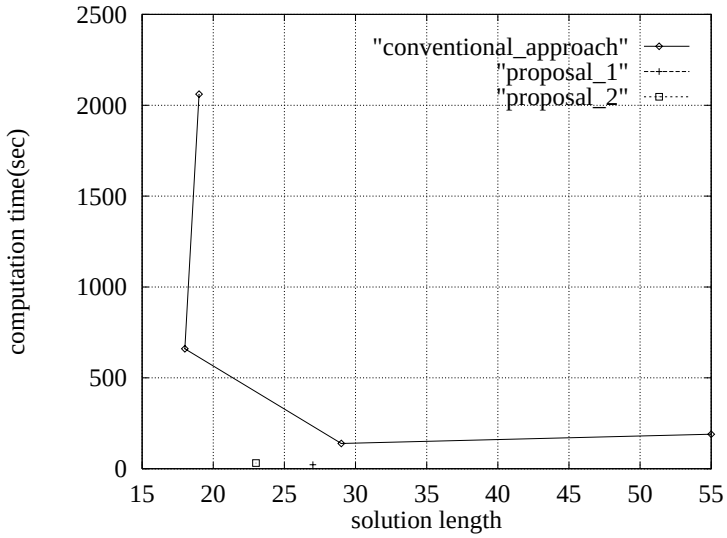


図 5.24: 解の長さと計算時間の平均値

図 5.25解長さと計算時間の平均値図 5.25 は、上段が、提案方式１で、下段が提案方式２の解決結果を示している。共に、比較用データとして、従来方式による解決結果が併せて示されている。大きな差は、無かったものの若干提案方式２の方が良かった。従来方式の search horizon $= 5$ に匹敵する解を、殆ど時間を掛けずにえることができた。

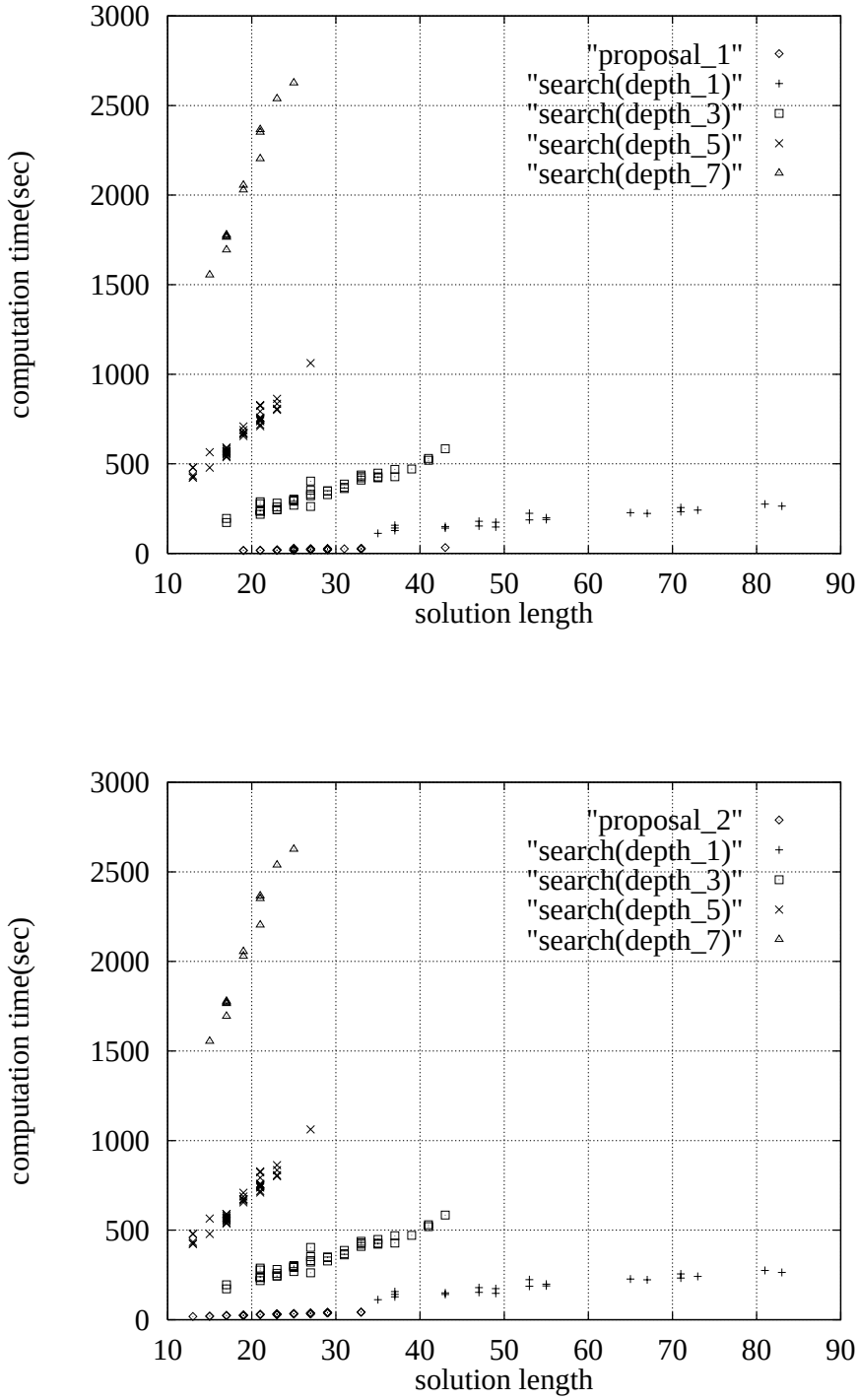


図 5.25: 解の分布状況

5.2.6　評価

提案方式１は、極めて単純な方法であり、とにかく左上コーナーに逃亡エージェントを追い詰めるものである。また、提案方式２は、追い詰める方向を４方向に拡張したものである。しかし解決能力は、４倍になる訳ではなく、平均で見た場合、１５%程度の改善であった。また、従来方式（ルールベースシステム）においては、全問題空間を考慮しながら解の導出を行っているにもかかわらず、良質の解を求めるのが容易ではなかった。これは、１５パズルの場合は、各局面にて採ることの出来るオペレータの数（探索空間における分岐因子）は、平均で２～３であった。本例題は一見簡単に見えるが、追跡エージェント側で、最悪８１９２通り選択肢の中から適切な動かし方を選択する必要がある。従って、先読みは、簡単なことではない。また、今回の場合意は、先読みの中に、相手の動きの評価関数を予め入れておいたので、相手の動きは一意に決まるようになっていた。にも関わらず、簡単には、解決できなかった。

本件についても、全般的に、人間の持つ解決戦略は、有効であり、その問題解決力のポテンシャルは相当に高いものと言える。また、そのような戦略を組み込んだ提案方式に基づく問題解決システムも有効であった。

5.2.7　まとめ

ここで、用いた、フェーズフローは、極めて単純な考え方に基づいている。パズルの場合は、タイルの並びを判定していたが、ここでは、逃亡エージェントと追跡エージェントとの相対関係を数式にて表わし判定に使用していた。たとえば、１つのＬ字形判定の例は、図 5.20 の中で示した。大きな解決の流れとして、ばらばらに分布していた追跡エージェントをＬ字形に結集して、後方に逃げないようにしてコーナーに追い詰めるということであり、パズルのときと同様、「問題解決の過程を後戻りさせない」というのは、重要なポイントであろう。これは、定式化の中の「 absorbing state の回避 」にて述べたとおりである。

5.3 航空機制御

5.3.1 はじめに

一般に、航空機の自動操縦装置は、制御理論の技術によって構築されている。しかし、複数の非線形制御変数を扱うことや不連続な操作系列を扱うことは、制御理論の枠組みで扱うことは難しい。そこで、ルールベースシステムのおける生成検査法に基づき、適切な操作の組み合わせを求めるという発想につながる。例えば、「XX 秒後に所望の状態となるための操作を決定する」問題を解決できる可能性かもしれない。しかし、組み合わせ対象変数が多ければ多いほど、組み合わせ爆発を招き易くなり、実時間での制御や情報処理は難しくなる。

さて、ここで人間が計器飛行をすることを考えてみよう。人間は、機体諸元を計器から読み取って操作を行っているが、これは、読み取った情報に基づいて、何らかの計算を行って操作量を決定している訳ではなく、多くの操作は、計器情報に基づいて、たとえば「条件Ａを満たす状態ならば操作ａにより条件Ｂを満たす状態に移行し、さらに、操作ｂにより条件Ｃを満たす状態へ……」と瞬時の判断による操作の決定と実行を繰り返しながら状態を変化させていく。

このような、航空機の操縦というものは、問題解決の枠組みとして見た場合、操作の流れが概ね決まっていて、その流れにそって、限られた、いくつかの操作を極めて単純化されたフィードバック制御あるいはフィードフォワード制御を行っているのではないだろうか。

本節では、まず、”航空機の操縦”という問題解決の枠組みが提案方式で記述できることを示し、ルールベースシステムとの比較検討を行う。また、提案方式において、問題解決の知識を増やした場合との比較なども合わせて行う。

5.3.2 問題の設定

本問題の定義は、仮想的な航空機の状態ｓの内容として、見かけ上、物理空間における位置、姿勢及び速度、つまり、

$$s = (X, Y, Z, \theta, \phi, \psi, V)$$

と考え、ある状態 s_1から別の状態 s_2へ、所定の操作を経て遷移する問題である。実際には、航空機の状態は、機体の可動部分の状態（方向舵、昇降舵、エルロン）、そして、あらゆる、コクピットのスイッチの状態などが含まれるが、ここでは、問題を分かり易くするため、上記のパラメータのみに着目する。

5.3.3　問題解決システムの設計（その１）

４章にて示した設計法により、本問題に関する問題解決システムを設計する。

（１） 提案方式の必要性・十分性

本例題は、実時間で意思決定を行い、適確な行動を起こす必要のある、自動車、列車、船舶、航空機の制御といった問題に対し、起こりえる、あらゆる、状況を条件式などで表現でき、問題状況ごとに、概ね解決ストーリーが決まっているのであれば、提案方式は有効な方法であろう。実際に、操作は、マニュアル化され、人間の専門家は、明確の操作内容を説明できる。特に、高い実時間性の要求下では、効果も大きいと考えられる。

（２）他の方式の評価

既に述べたように制御理論だけでは、不連続な操作系列を十分扱うのは難しい。また、ルールベースにおいても、対象操作が多い場合は実時間性が失われる。例えば、エキスパートシステムに関する航空機制御に関する研究は、不調に終わっている [KOUKUU 1989][KOUKUU 1990]。

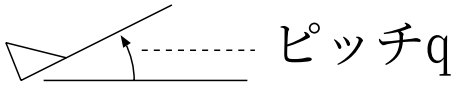
（３）知識源の検討

知識源としては、飛行マニュアル、パイロットのコメントなど有用なものが豊富である。これのものを利用しながら、提案方式を適用する。その過程においては、数学的な知識も利用していく。

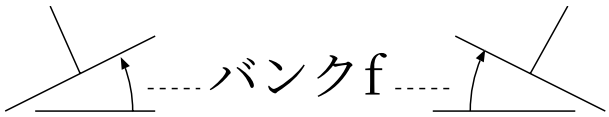
（４）解決ストーリー

既に述べたように、ここでは、３次元空間を飛行するという極めて基本的なことが、提案方式でフォーマルに記述でき、システム化できることを目指す。以下、パイロットのコメントを基に、そのための基本原理を整理する。

1. 上昇・降下は、機体がサイドバンクしていない状態にて、操縦桿を引く。(ピッチ θが増減)



2. 旋回は、機体の進行ベクトルが地面に対し、平行な状態において、機体にバンクϕを与え、操縦桿を引く。（機首方位 ψが、増減）



3. 加速は、スロットルを前方に押し出す。また、減速は引き戻す。（速度Ｖの増減）急減速は、スピードブレーキをオンとする。（速度Ｖの急減速）

これらを図解すると、以下のようになる。

操縦の一般的性質

航空機の操縦桿、スロットル及びスピードブレーキと運動との関係を整理する。ここでは、航空機の運動の定性的な運動の傾向を述べる。ここでは、機体上の３軸周りの運動と前進速度のコントロールを中心に述べる。

性質１ ： 操縦桿を前後に倒せば、Y 軸周りの回転が生じる

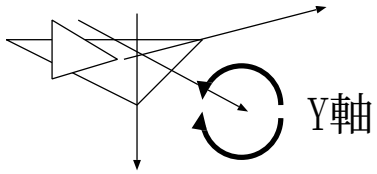


図 5.26: Ｙ軸回りの変化

性質２ ： 操縦桿を左右に倒すことで、X 軸周りの回転が生じる

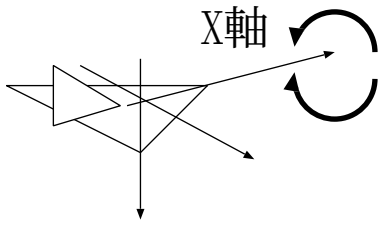


図 5.27: Ｘ軸回りの変化

性質３ ： 機体を傾け（バンク）操縦桿を後方に倒すことで、旋回する

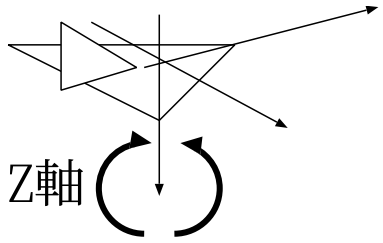


図 5.28: Z 軸回りの変化

性質４ ： 速度制御

スロットルレバーを前方の押し出すことでX軸方向に加速、また、後方に引くことで減速。スピードブレーキを ON とすることで、同様に、減速。ただし、重力の効果により、降下時、加速。

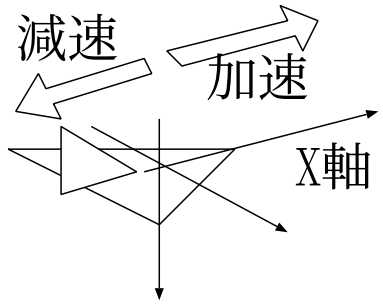


図 5.29: 速度変化

以上より、それぞれの操作が相互に影響し合っている。具体的な可解性は、以降、（６）にて示す。

（５）問題の定義

(a) 状態の知識表現

$$s = (X、Y、Z、\theta、\phi、\psi、V)$$

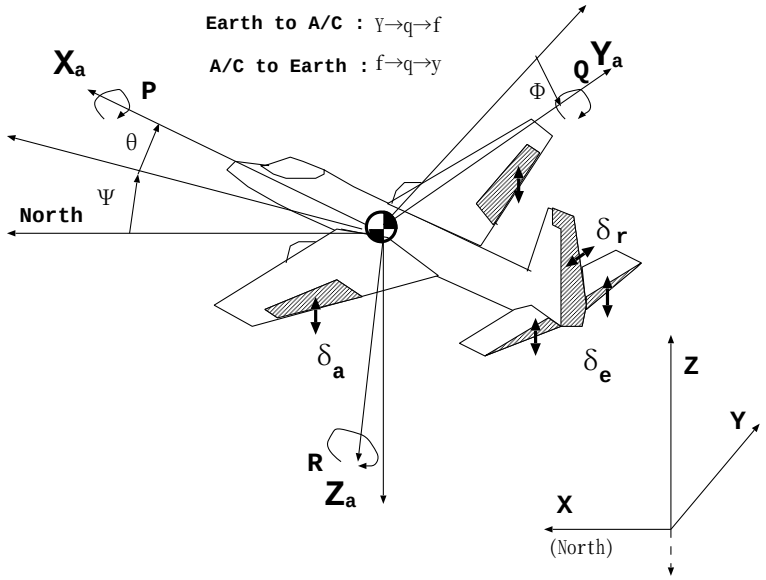


図 5.30: 航空機モデル

(b) 問題空間 S

$$S = \{(X, Y, Z, \theta, \phi, \psi, V)|$$

$$X = -\infty\sim+\infty(m), Y = -\infty\sim+\infty(m), Z = 0\sim10000(m),$$

$$\theta = -90\sim90(deg), \phi = -180\sim180(deg), \psi = 0\sim360(deg),$$

$$V = 200\sim600(knot)\}$$

(c) ゴール空間 W

所望のパラメータを $X_{desired}, Y_{desired}, Z_{desired}, \theta_{desired}, \phi_{desired}, \psi_{desired}, V_{desired}$ とすると、

$$W = (X_{desired}, Y_{desired}, Z_{desired}, \theta_{desired}, \phi_{desired}, \psi_{desire}, V_{desired})$$

(d) オペレータ R

Rは、操縦桿操作(前後: $stick_{(fa)}$、左右: $stick_{(lr)}$)、スロットル操作($throttle_{(1)}$〜$throttle_{(9)}$)及びスピードブレーキ操作($spdbrk_{(ON/OFF)}$)のみを考慮する。

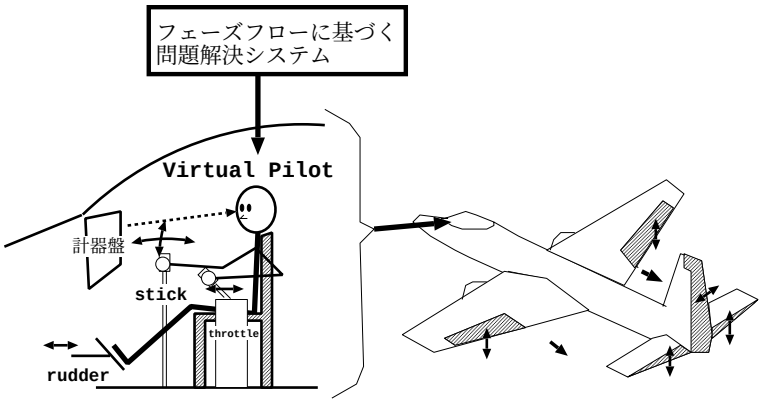


図 5.31: オペレータ

(e) 評価関数

個々の部分問題において、各問題における任意の解決過程が単調増加または単調減少となるような関数を個別に定義する。

（６）可解性の検証

先に述べた航空機操縦の一般的性質より、操縦桿、スロットルレバー、スピードブレーキは、航空機の姿勢、速度に対して一定の傾向にて変化を与える。特に、航空機では、ピッチθ、機首方位ψ、速度Vが３次元空間の任意の点に移動するための手段になっていることは、数学的に容易に説明がつく。即ち、３次元空間座標 X,Y,Z の各速度成分は、

$$V_x = V cos\theta cos\psi$$

$$V_y = V cos\theta sin\psi$$

$$V_z = V sin\theta$$

初期位置(X_0, Y_0, Z_0)、速度$V > 0$で、かつ、θ、ψについて制御可能なとき、X,Y,Z は

$$X = X_0 + \int V_X dt$$

$$Y = Y_0 + \int V_Y dt$$

$$Z = Z_0 + \int V_Z dt$$

によって、あらたな X,Y,Z に移行可能である。(補足：機体の Z 軸 Z_aは機体のノーマルの状態に対して下向きであるが、この絶対座標系の Z 軸は、本来のオイラー座標 Z 軸の符号を反転したものと定義する。)

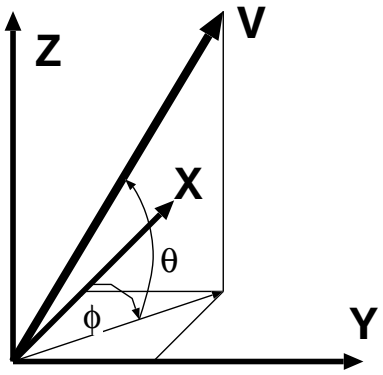


図 5.32: 航空機操縦における問題空間

図5.33は、本問題における操縦操作と状態変数の関係を示している。もしも、θ, ψ, Vが自由に設定できれば、図5.32からも分かるように、任意の状態$s_1 \in S$から任意の状態$s_2 \in S$への遷移は可能となる。このようにするには、操縦桿、スロットル及びスピードブレーキの操作だけから、ピッチ角θと方位角ψと速度Vに矛盾なく変換する方法を定義する必要がある。

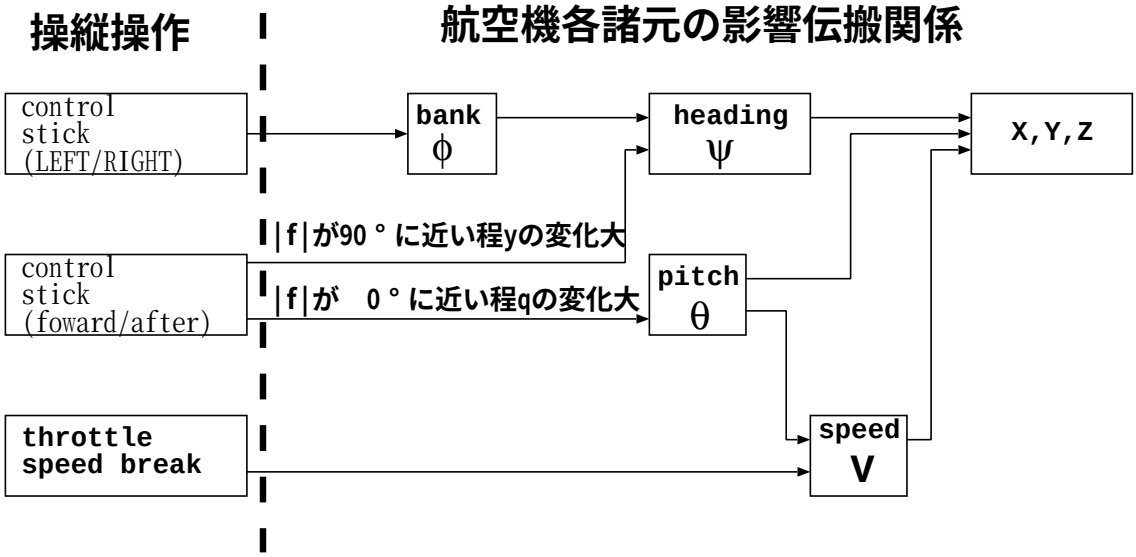


図 5.33: 航空機操縦における制御の流れ

（7）フェーズフロー

問題解決のために以下のフェーズフローを定義する。操縦操作をいくつかの部分問題に分割して実施する。姿勢に関する操作順序は、第一に、機体のバンク角ϕをゼロとする。第二に、ピッチ角をコントロールし、所望の高度に遷移を行なう。第三に、水平旋回を行い機首方位を所望へ変える。一方、速度に関する操作対象は、スロットル及びスピードブレーキによる加速度のみ考慮する。

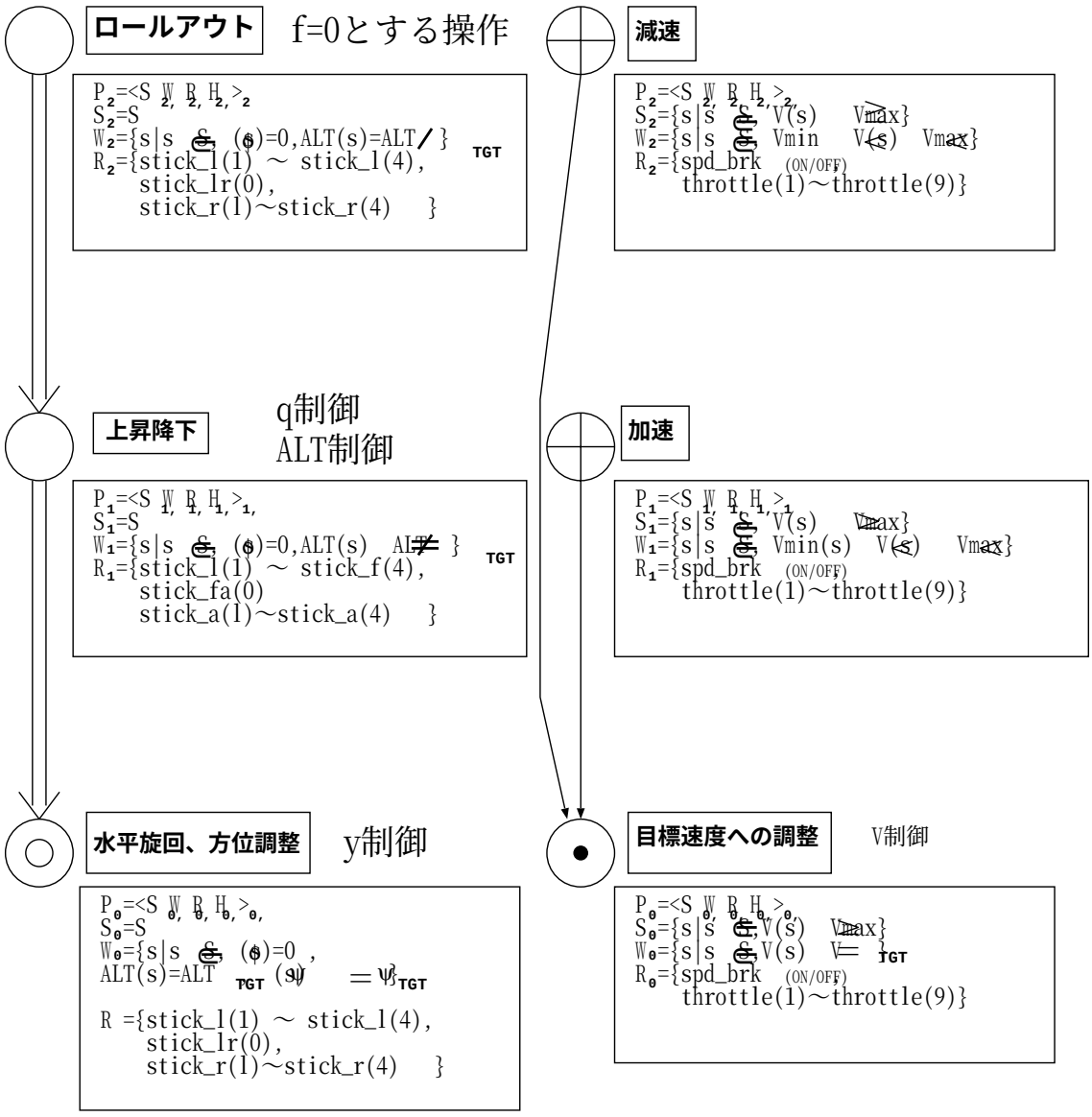


図 5.34: 操縦フェーズフロー

5.3.4 実験及び結果

実験１

図5.34のフェーズフローによる操縦システムを構築し、20 ケースの例題 (Appendix E) を与えて問題解決を実施する (TEST1)。次に、バンクが-10〜+10(deg) の範囲でも上昇及び降下を認める緩やかな条件のフェーズフローを図5.34のフェーズフローに加えて生成した新たなシステムにも同様の問題解決を実行する (TEST2)。そして、これらを比較する。なお、問題空間 S の範囲については、重力などの物理的制約条件を考慮して問題解決を定義し、以下の様に絞り込んで考えるものとする。これらは問題の定義域となる。

$$
\begin{aligned}
ALT_{TGT} &= 1000\text{〜}10000(m) \\
V_{TGT} &= 200\text{〜}600(kts) \\
\psi_{TGT} &= 0\text{〜}360(deg) \\
\theta_{TGT} &= -10\text{〜}60(deg) \\
\phi_{TGT} &= -60\text{〜}60(deg)
\end{aligned}
$$

前項にて設定した問題解決システムが、最終的に目指す諸元は以下のとおりとし実験する。

$$
\begin{aligned}
ALT_{TGT} &= 5000(m) \\
\psi_{TGT} &= 90(deg) \\
V_{TGT} &= 400(kts) \\
\theta_{TGT} &= 0(deg) \\
\phi_{TGT} &= 0(deg)
\end{aligned}
$$

なお、フェーズフローによる問題解決では各フェーズ毎にワーストケースを見積もることができる。各々の変数について目標値に対するフィティングのための所要時間のワーストケース見積もりは、

- 高度のフィティング　:　45(sec)
- 方位のフィティング　:　45(sec)
- 速度のフィティング　:　30(sec)
- ピッチのフィティング　:　30(sec)
- バンクのフィティング　:　30(sec)

よって、問題の定義域であれば、トータル解決時間 180sec 以内となるはずである。

ここで、問題空間内の変数のランダムな組み合わせについて実験を行った結果を図 5.35 に示す。当然ではあるが、最悪のケースでも、見積もり値内であった。改良版のフェーズフローでは、オリジナルに対し一様に改善されている。

TEST1:オリジナル

TEST2: 追加バージョン（上昇降下開始条件の緩いフェーズフローを追加したもの）

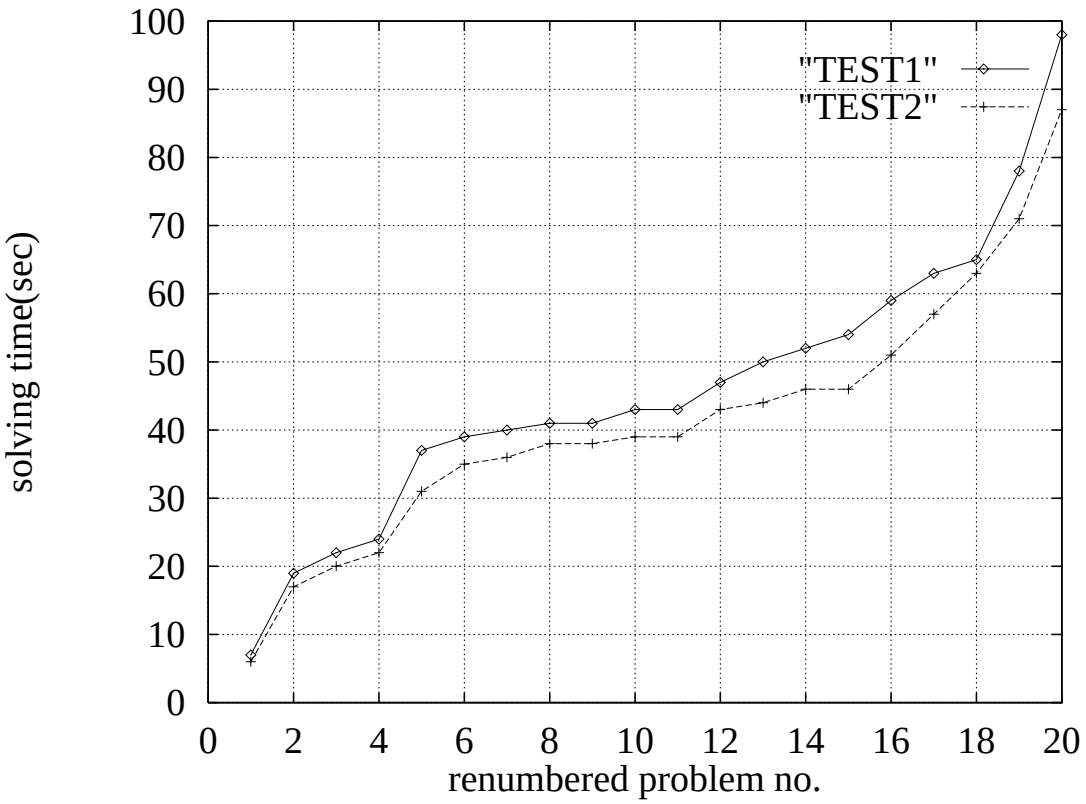


図 5.35: TEST1 および TEST2 の解決性能の比較

実験２

ルールベースシステムにて、操縦システムを構築し、比較実験を行う。本システム５秒に１回操縦のための意思決定を行うものとした。

問題の設定（その１）

初期位置を原点とし、そこから進行方向の 30000m 前方、5000m 左にずれたポイントに移動する問題を考える。また、初期状態、ゴール位置通過時の姿勢とも、$\theta = 0(deg)$、$\phi = 0(deg)$、$\psi = 90(deg)$ とする。移動に際して、実時間探索（先読み深さ１、３、５）を使い、姿勢の妥当性、経路の最短性などを評価しながら移動するものとする。

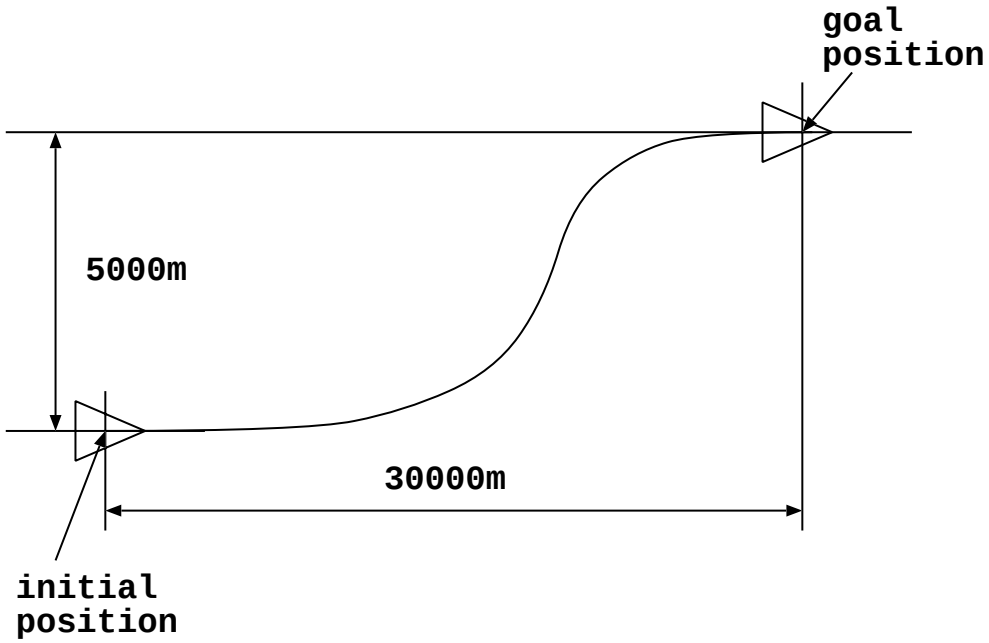
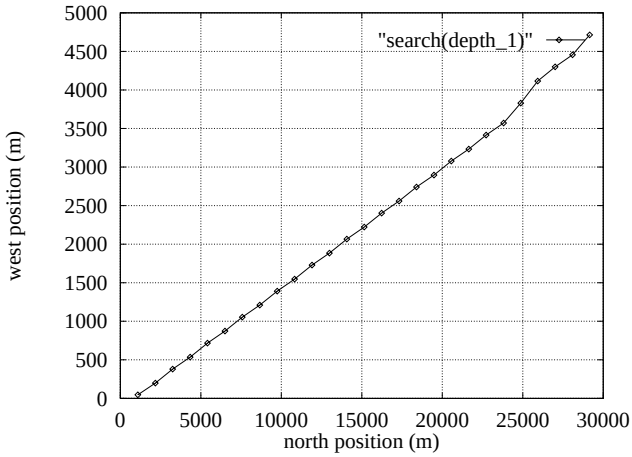
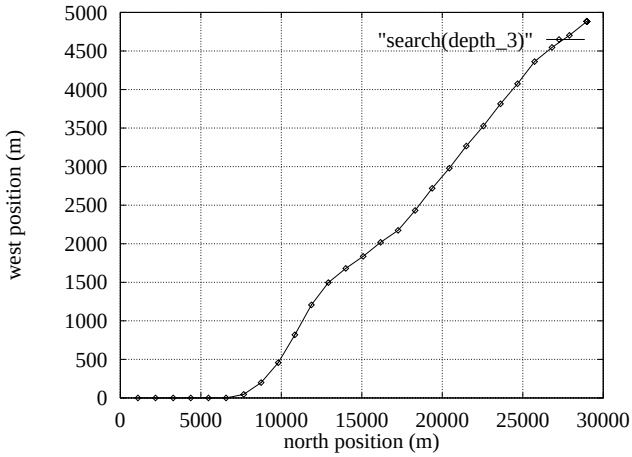


図 5.36: 実験２の設定（その１）

本問題は操作内容を細分化していくに従って計算時間が増大する。そこで、操作内容を限定し、先ずは、水平面の運動のみ確認した。その結果を、図 5.37に示す。図のように、読みの深さが深くなるに従って、ゴール時の状態が理想的な状態に変化していった。本実験では、水平面のみでも理想的な結果を得るのに、少なくとも、先読み深さ 5(25 秒先の状態) が必要であった。





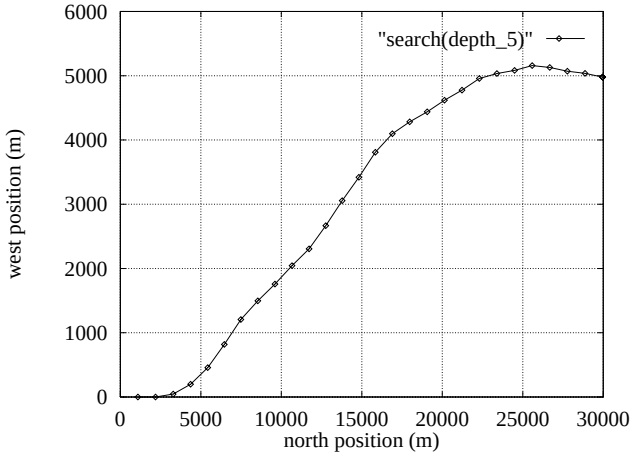


図 5.37: 航空機の実時間探索制御における航跡の変化 (先読み深さ 1,3,5)

問題の設定（その２）

上下方向の変化も含めた飛行制御問題、ここでは、先のその１より問題の難易度をあげて着陸問題を考える。

- 着陸点からの初期距離：３００００ m
- 着陸コースに対する初期のずれ：５０００ m
- 初期高度：４０００ m

結果として、図 5.38のような結果が得られた。

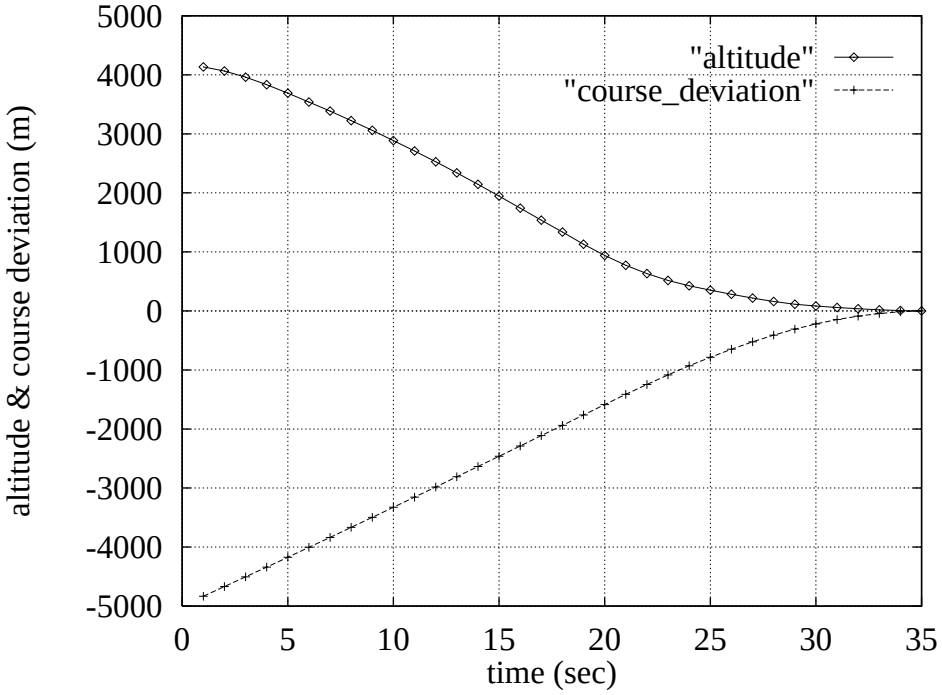


図 5.38: 提案手法による着陸までの航跡の変化

問題設定（その１）では、同一平面上の単調関数でも、妥当な解を得るためには、２５秒分の先読みが必要であったが、本提案方式では深い探索なしに安定に解決できることが示された。

5.3.5 評価

実験１で述べたように、提案方式は各フェーズ毎問題解決時間のワーストケースの見積もりが得られる。今回の実験では、実験値は見積もり値以下となっている。また、本例題でも示したとおり、より効果的なフェーズフローを追加した場合、矛盾なく問題解決能力を改善することが出来た。

実験２では、実時間探索と比較を行なった。明らかに、現実の世界で人間が行なっているような操作系列や飛行パスを実現するためには、かなりの深さの探索が必要になることが分かった。提案方式の場合、極めて少ないパラメータでのシミュレーションであったが、比較的複雑な問題を容易に扱うことが出来た。現実世界の問題では、対象となるパラメータも非常に多くなるため、結果的に、一層、本提案方式のようなフェーズの分割の考えが必要になろう。

5.3.6 まとめ

本節では、航空機の操縦の例題をとおして、乗物などの制御の問題も提案する枠組で解決できることを示した。

一方、見方を変えれば、本論文にて提案したフェーズフローを使うことで新規設計のヒューマンインターフェース、たとえば、航空機のコクピットの操作を組合わせることで所望の諸元に飛行できることを数学的に説明できる可能性がある。

次章では、本章の実験についてディスカッションする。

第 6 章

実験結果に対するディスカッション

２章の人間による実験の例、５章の計算機による実験の例について再度議論する。

6.1　人間の持つ問題解決戦略について

6.1.1　被験者の妥当性

２章においては、人間が持つ問題解決戦略の一例を示した。被験者は、小学生の子供であったが、同種のパズルに対して何度か解いた経験のある人、全く経験のない人、あるいは、パズル全般に対して得意な人、不得意な人など様々であった。多くの被験者は 10 分程度の練習で、このパズルの性質が理解できた。被験者には、上からそろえる必要のないこと、また、なるべく少ない手数でそろえることを考えてもらいながら取り組んでもらった。メンバーが特定のタイプの人間に偏っていないことは、本実験の被験者として妥当であった。また、小学生の高学年ということについては、実験を行うに当たっての思慮分別の点、思考の柔軟性及び知識量や経験の点で妥当と考えた。また、大人に比べユニークな発想が出ることを期待した。因みに、低学年でも試みたが、実験進行が困難であった。

6.1.2　問題の妥当性

特別な知識や長期の経験を必要としない問題なので適当な問題と考えられる。

6.1.3　未経験者の解決戦略

実験問題の未経験者においても、普段の生活の中に、「並べ替え」などの操作や、「スポーツ」などの中のにも「フォーメーション」といった概念は存在し、特に、予備知識など無くても、日常生活の応用にて有効な解決戦略を発見可能と考えられる。

6.2 計算機実験の結果に対する個別ディスカッション

6.2.1 可解性

今回実施した例題の中で、パズルとゲームに関しては、比較的簡単な例題であったので、各部分問題は容易に可解であることは理解できる。また、航空機の操縦についても、問題を整理してみると人間が無意識に行っている操作が飛行という問題解決を巧みに解決していることがわかる。 しかしながら、ここで、この可解性が成り立つのは、与えられた問題の状態が、問題解決システムの定義域に含まれる場合の話であり、たとえば、N2-1 パズルでは、与えられたパズルの状態において、任意の２箇所のタイルを入れ替えた場合、仮に全探索を行ったとしてもゴールへの到達は不可能であり、また、航空機の操縦においても、墜落の瞬間の状態からの回復操作は存在しない。 ここまでの議論の中で、「部分問題の系列」と言ってきたのは、あくまで、解決可能な部分問題の系列を指しており、従って、実際に、システムを設計する場合、その定義域が、ユーザーの要求を満足しているかを吟味し、満足していない部分がある場合は、その旨、明確化しておく必要があると考えらえる。

6.2.2 問題解決のための知識内容とその記述量

表6.2.2に、５章で扱った、各例題に対する問題解決のための知識例とその記述量を示す。

表 6.1: 各例題に対する解決知識と記述量

例題	フェーズ判定	オペレータ	評価	部分問題数
15 puzzle	配置データ don't care 含む	属性および位置の 入れ換え	マンハッタン距離	40 (設計１) 46 (設計２)
追跡ゲーム	エージェント相対 距離	属性および位置の 入れ換え	マンハッタン距離	6 (設計１) 33 (設計２)
航空機の操縦	例 $ALT > 4000$ & $ALT < 6000$	操作機器と操作量	目標諸元と 現諸元との差	6 (設計１) 9 (設計２)

提案方式による問題解決知識の内容は、既に述べたように、問題解決ストーリーをどのような部分問題 (つまり、S_i,W_i,R_i,H_i自体が問題解決知識) にて定義されるかによる。この点で、if〜then…をルールの基本構造として推論を行う従来のルールベースシステムとは様相が異なる。提案方式における知識の記述量は、全フェーズフローがいくつの部分問題から構成されているのか、また、個々の部分問題の定義（S_i,W_i,R_i,H_i）がどの程度の記述量で表現できるかによる。パズルにおいては特定のタイルをそのホームポジションに揃えることが個々の部分問題のゴールでり、どのような順で揃えるのかが重要であった。従って、基本的にマス目の数に比例して部分問題は増加することになる。パズルやゲーム

に比べ、航空機の操縦の方が、状態数が多く、一見複雑であるように思われる。しかし、問題を分析してみると、たとえば、高度 5000m と 10000m で操縦手順は同じであり、実際には、少ない部分問題によってシステムを実現することができた。また、木目細かいフェーズフローに改良することも可能であり、今回、各例題にて取り敢えず解決できるシステムとよりよい解を得ることが可能なシステムの２種類の例を示した。ユーザーの要求性能次第でチューニングの目標が決まり、そして記述量も決まるものと思われる。

6.2.3　処理時間（実時間性）

３つの例題は、ともに、オペレータを１回出力する時間、すなわち、全部分問題条件のチェック、部分問題の選択、および、部分問題の解決に要する時間はきわめて短時間であり、その計算時間の上限値については、既に、４章にて検討したとおりフェーズフローの大きさ（フェーズフローの数とフェーズ数）に比例する。今回の例題と実験環境下においては、数ミリ秒のオーダーであり、パズルにおいては、問題全体を解決するのに、１秒程度であり、パズルは、追跡側の指し手決定において８ノードの状態評価を実施するため数秒かかった。また、航空機制御においても、殆ど瞬時に解決可能であった。従って、十分な実時間性を有するものと考えられる。一方、比較用システムの場合では、search horizon 10 にて、１時間近くかかるケースが存在した。

6.2.4　開発及び改修の容易性

今回の 3 つの例題において、問題分析、設計、システム構築、デバッグなど含めて、それぞれ１週間程度で完了できた。　改修または改良において、一般にルールベースシステムで言われているのと同様、本提案システムにおいても、モジュール性が存在する。たとえば、航空機の操縦に関する例題において、フーズフローの追加の例において示したように、適切な追加は問題解決の性能を改善する。

6.3　計算機実験の結果に対する総括

５章におけるパズル、ゲーム、航空機の操縦という問題は、一見異質な問題のように見え、同じ枠組みで扱ってよい根拠に乏しいように思える。特に、パズルやゲームなどのトイ・プロブレムといわれるクラスの問題と乗り物の運転、操縦といったリアルワールド寄りの問題は、異質に見えるかもしれない。しかし、それは、探索重視の手法を念頭に置きすぎているためと考えられる。

ルールベースシステムにおいて、実時間探索のような機能は、リアルワールドの問題を扱う上で欠かせないツールの一つと考えられる。しかし、今回の実験でも明らかになったように、素直に人間の問題解決ストーリーを構築した提案手法と大きな差がなかった。逆に、複雑さが増すほど、探索では手に負えないくなり、本提案手法の有効性が強く現れる。

問題解決エンジンに実時間探索を使用する場合、ハードウェア一定のもとでは、解の質 と 計算時間には、反比例的な関係にあり、下図の斜線より左下の領域の性能は実現するのは困難である。現実世界の問題では、ナレッジエンジニアやシステムエンジニアは、ユーザーの要求を十分吟味して、search horizon を設定する必要がある。

しかし、今回の提案方式で利用可能な問題解決戦略があれば、不可能と思われていた問題解決の性能を実現できる可能性がある。

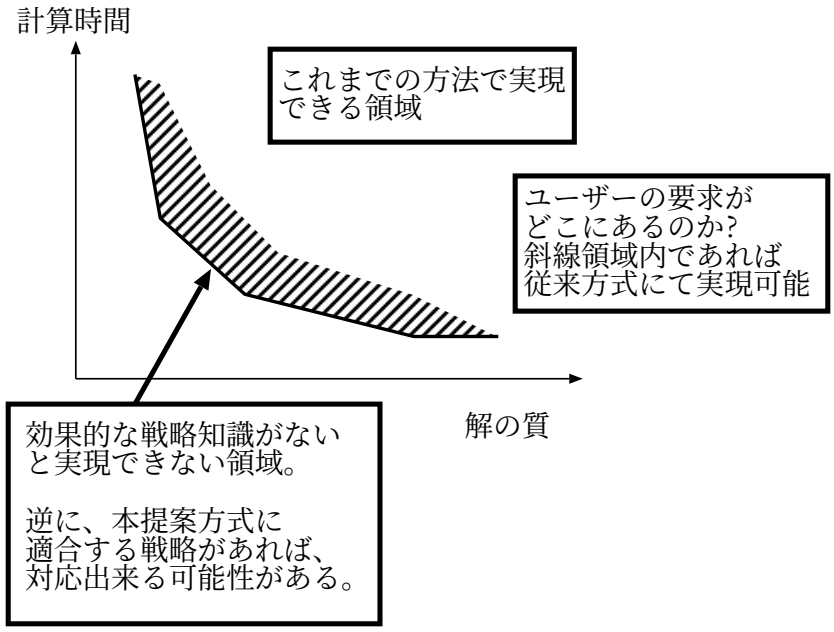


図 6.1: 解の質と計算時間

ここで、3つの例題の共通点は、パズルやゲーム盤面の配置、あるいは、物理空間の位置、姿勢、配置といった項目である。これらは、問題の区分から言えば、解析問題に属する。ある意味で、人間にとって視覚的に分かりやすい問題であり、定理の証明や法律上にの推論とはその点、異なる問題であろう。従って、小学生のように、法律や病気の診断に関する知識をもっていなくても戦略知識を構築できる。法律などの論理を扱う問題への応用については、今後の検討が必要であろう。また、合成問題に対する検討も必要であろう。

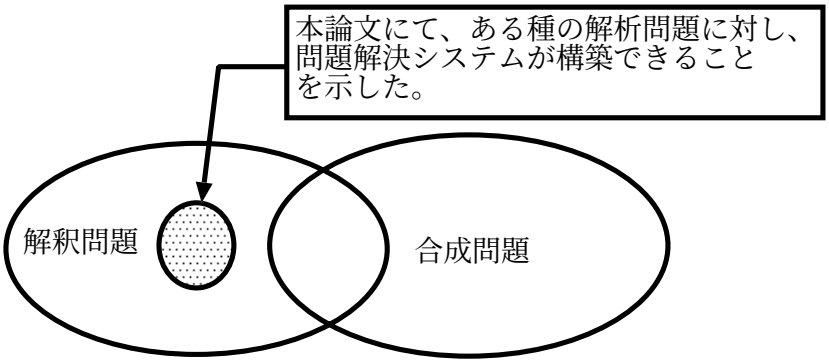


図 6.2: 適用対象

第 7 章

結論

7.1 はじめに

本研究の全般的まとめを行い、明らかにされた事実（確認された事実、予定外にわかった事実)、今後の課題、そして、今後の展望について示す。

7.2 本研究の主題

本研究は、人間の持つ問題解決戦略自体にかなりの問題解決能力が秘められているという前提に立ち、その問題解決戦略を効果的に利用する問題解決方式を提案し評価することであった。 ここで、対象にする問題は、人間がある程度その問題を理解し、ランダムな出題に対しても殆どのケースで解決できるような問題を対象にしていた。しかし、ある程度解決プロセスがわかっているといっても、本来、手続き記述型アプローチの問題解決システムでは、一般に、制御コードと知識の判別がつき難くいことや幅の狭い解決システムになる可能性が高いことといった問題があり、一方、従来のルールベースアプローチのシステムでは、実際にシステムを実行しなければ、その動作内容はわかり難く、また、問題解決時間が長いことなどの問題を伴う。 今回提案する問題解決システムは、人間が持つ問題解決のストーリー（プロセス）を、一連の単純な部分問題の系列として計算機上に知識表現することで、従来の問題点を多少なりとも解決しつつ、より良質の解を得ることを目指すものである。特に、当初から以下の検討項目に着目しながら研究を行ってきた。

1. 人間の持つ解決戦略は、どの程度の解決能力を有するのか。
2. 人間的解決戦略を用いる解決手法が従来の枠組みの延長において定式化可能か。
3. 人間的解決戦略は獲得可能か、そして、計算機上に実装可能か。
4. 人間的解決戦略を用いるシステムは、十分な解決能力を有するか。

7.3　本研究にて明らかとなった事実

7.3.1　人間の持つ解決戦略の有効性

２章では、実際に被験者を集い、15 パズルと追跡ゲームを例題として問題解決をしてもらった。そして、その結果、特に専門家でもない人間が、比較的良質な解を導くことができた。　問題自体は単純であったため、ここでの２つの問題に対し、それぞれ全く初めてという被験者においても、30 分〜1 時間の練習で問題に慣れることができ、ゴール状態まで解決できるようになった。　今回、比較のために用いたルールベース型問題解決システムは、ヒューリスティック実時間探索を解決エンジンとして用いるものを使用した。結果的に、被験者たちの解決結果 n について、比較システムの先読み深さを尺度として測定してみると、解の質という点において、パズルの場合、先読み深さ 8 程度、追跡ゲームの場合で、深さ 5 程度の解に匹敵することが分かった。　また、被験者たちの用いた解決戦略は、比較用システムの方にて分析してみた結果、先読み深さが深い程、顕著に出現するという重要な知見を得ることができた。一般に、深い探索にて現れる解決戦略の方が知的と考えるのは妥当な見方と考えられる。従って、人間（ましてや、小学生）が、初めて直面する問題に対し、１時間足らずである程度知的な解決戦略を適用することができたのは注目すべき事実であろう。　また、解決過程において、被験者は時々試行錯誤する部分も若干あったため、より洗練された解決技能を持つ被験者ならば、さらに、良い解を得ることができたと思われる。

7.3.2　人間の解決戦略を用いる問題解決システムの定式化

３章にて、システムの提案と定式化を行った。　ここでは、Barnerji が行った定式化における“ 問題 ”の基本構造 $P =< S, W, R >$をもとに、人間のエキスパートの持つ問題解決のストーリーを部分問題の系列として抽出し、さらに、部分問題を節点と見たてたとき、節点が一列に連結されたグラフ構造として扱えることを示し、さらに、解決が適切に進むための条件を明らかにした。これまで、人工知能またはその他の分野において、同様の方式及び定式化はこれまで無く、問題解決の新たな枠組みを構築できたと考える。

7.3.3　人間的な解決戦略の抽出及び実装

４章にて、システムの設計法を提案した。提案方式を構築するための解決戦略としての条件を明確にし、5 章において、実際に、提案する設計法を用いて、システム構築を行い、その構築過程と根拠を示すことができた。これにより、提案方式の適用可能な問題が存在することと同時に、それをシステムとして実装できることを示すことができた。

7.3.4　人間的な解決戦略を用いるシステムの問題解決能力

５章の計算機実験において、本提案方式に基づくシステムの問題解決能力を確認した。その結果、得られた解の質は以下のとおりである。

例題	知識源	解の質
15 puzzle	２章にて得た人間の戦略	実時間探索における先読み深さ２０に相当
追跡ゲーム	２章にて得た人間の戦略	実時間探索における先読み深さ５に相当
航空機の操縦	パイロット・コメントなど。	実時間探索における先読み深さ２０に相当

上記の表に示すように、２章で得た平均的な人間が思いつく程度の、一見、平凡な解決戦略の解決結果も、実時間探索の先読み深さの尺度に照らし合わせてみると先読み深さはかなり深いものに匹敵する。パズルの例題において、先の人間の結果よりも格段に向上しているのは、計算機処理に置き換えたことにより、迷いや試行錯誤的な手順が無くなったためである。

7.3.5　提案方式が有効に機能したことに対する理由

提案する問題解決方式は、各部分問題毎単純な評価関数と限定されたオペレータを用いて構成されている。通常、山登り法などの単純なオペレータ選択法では、問題解決は困難である。しかし、たとえば、15 パズルにおいても、問題空間の１点のサブゴールを目指すことに対し、単調関数にて状態評価関数を表すことが出来ないのは、すぐに理解できる。上記の問題解決の困難さは、各部分問題を空間結合にてリンクするいう方針を、打ち出したことで解消できた。すなわち、評価に不要な情報は、無視し必要な情報のみを評価対象にするという明確な方針が、単純で且つ単調な関数の使用を可能にしたと考えられる。

7.4　今後の課題

本研究では、人間の持つ解決戦略を利用する問題解決方式を提案し、そしてそのようなシステムの設計法を確立した。しかし、人間が効果的な問題解決ストーリーを構築出来ること、あるいは、有効なオペレータを思いつくのは、その人間自身の経験や学習などが大いに関係しているものと考えることができる。　ここで、計算機によって、自動的に部分問題の系列を見出す方法が明らかとなれば、非常に画期的なことであろうと考えられる。たとえば、極端な話として、ある問題に対し、10 才の子供の発想と同程度の部分問題の

系列を計算機にて生成することを考える場合、人間の子供が生後 10 年間に経験する知識や学習内容を何らかの形で計算機に与え、これを用いて生成する必要があるかもしれない。結果的に、このような部分問題系列の自動生成を考えるということは、特定の人間の経験にとらわれない何らかの生成方法を確立するということと言える。しかし、多様な問題に共通して適用可能な、部分問題系列の生成手法を確立するのは、容易なことではないと考えらえる。

7.4.1 現状の技術で解決可能な課題

以下の２つの機能があると、デバッグ効率が格段に向上するものと思われる。

1. 隣接する部分問題系列の可解性を検証する機能があることが望ましい。
2. 部分問題のリンク構造が図解できる機能があることが望ましい。

7.4.2 論理中心の問題と合成問題への応用

今回は、制御など目的とする解釈問題の一部の例題に対し適用した。今後、法推論などの論理的な分野への応用や合成問題（計画問題、設計問題）への応用なども検討する必要がある。

7.4.3 今後の展望

1. 未解決のゲームの必勝法やゲームプログラミングの実力向上などに利用可能である。
2. 計算機を用いた教育支援システム, 即ち、CAI は、学校教育を中心に古くから研究が行なわれているが、特に、学習者の理解状況を判断し学習者に応じた教育を行なうためのシステム (Intelegent Tutoring System : ITS) の研究に対し、 本研究のフェーズフローが応用可能と考えられる。たとえば、パイロット訓練やプラントオペレータの訓練において、予め、対象システムのフェーズフローを作成しておき、誤った操作をした際の、自動指導システムが構築可能と考えられる。
3. その他、多くの制御問題には、応用可能と考えられる。

謝辞

北陸先端科学技術大学院大学（JAIST）における研究生活を始めるにあたり、心構えと研究上の助言を頂きました大学時代の恩師 河村 龍馬 先生（東京大学宇宙航空研究所長、東京大学名誉教授)、そして、広瀬 直喜 博士（ 科学技術庁 航空宇宙技術研究所 主任研究官 ）に感謝致します。

JAIST での研究を快く承諾して頂き、そして、送り出して頂きました 所属企業（三菱プレシジョン株式会社）のフライトシミュレータ設計部門及びその関係部門の皆様に感謝致します。

JAIST での研究活動の中で、1997 年まで研究指導をして頂きました情報科学研究科木村 正行 教授（現 JAIST 副学長）には、本来の研究指導はもとより、有益な人生論までもご教授頂き感謝致します。1998 年から私の研究指導をして頂きました 下平 博助教授 には、私の個人的都合により、夕方の家族の団欒のひとときまでも、しばしば、私の研究打ち合わせのために貴重な時間を振り向けて頂きましたこと、申し訳なく思うと同時に、深く感謝致します。また、長期に渡り、有益な助言を、その溢れるようなパワーで授けて頂きました 國藤 進 教授（現 知識科学研究科）に感謝致します。木村 教授の後任として着任されました嵯峨山茂樹 教授 には、本研究をまとめるあたり、論文の隅々までコメントを頂きましたこと、また、深夜まで助言を頂きましたことなど印象深く、厚くお礼申し上げます。また、予備審査の席上で、本質的で有益なご指摘及び助言を頂きました、電気通信大学 西野 哲朗 助教授に感謝致します。その他、様々な助言を頂きました、旧 木村研究室、下平研究室、嵯峨山研究室、國藤研究室の皆様は厚くお礼申し上げます。

そして、暖かく見守ってくれました 私の両親を始め、義父、また、亡くなった義母、その他、妹夫婦、親戚の皆は、私の心の支えでありました。特に、義父、早野俊一氏には、研究上のアドバイスまでも頂き深く感謝する次第です。

最後に、私の研究を笑顔で応援してくれた三人の子供達、意己、暁子、祥子には感謝の気持ちで一杯であり、また、私と子供達の健康面・精神面を一人で支えてくれた妻、崇子 には心より感謝する次第です。

参考文献

[Araya 1996] 荒屋 真二 , 百原 武敏 , 岡本 吉弘 , 山口 建二 , “８パズルの問題空間解析とその完全解”, 人工知能学会誌, Vol.15,No.1,pp.37–46,Jan.1996.

[Athans 1974] Athans,M. et al., “System Networks and Computation : Multivariable Methods”, *New York : McGraw-Hill*,1974.

[Bachant and McDermott 1984] J.Bachant and J.McDermott, “R1 Revisited, Four Years in the Trenches”, *AI Magazine Fall*,pp.21-32,1984.

[Banerji 1969] R.B.Banerji, “問題解決の理論—人工知能の基礎”, 産業図書, 南雲 仁一, 野崎 昭弘　共訳,1969.

[Banerji 1983] R.B.Banerji, “人工知能—コンピュータによるゲーム”, 共立出版, 高原 康彦, 中野 文平, 宇治橋 義弘 共訳,1983.

[Benda et al. 1985] M. Benda, V. Jagannathan, and R.Dodhiawalla., “On Optimal Cooperation of Knowledge Sources”, *Technical Report BCS-G2010-28, Boeing AI Center*, 1985.

[Brown 1980] Brown, J.S. and VanLehn, K., “Repair Theory of bugs in procedural skills”, *Cognitive Science* ,Vol.4,pp.379-426,1980.

[Bunge 1956] Bunge,M., “Causality”, *The President & Fellows of Harvard College*,1956.

[Buchanan 1978] Buchanan, B.G. and Feigenbaum, E.A., “Dendral and Meta-Dendral: their appricationsdiamension”, *Artificial Intelligence*,11(1,2), 5-24, 1978.

[Chandrasekaran 1993] Chandrasekaran,B., Goel, A. K., Iwasaki Y., “Functional Representation as Design Rationale”, *IEEE Expert,*Vol. 8, pp.48-56, 1993.

[E.iida 1994] 飯田 栄治 , 下平 博 , 木村 正行 , “８パズルの高速解法”, 電子情報通信学会, 信学技報,COMP94-54〜66, Nov.1994.

[E.Iida 1996] 飯田栄治 “大局的戦略に基づくゲーム木探索の一提案”, 電気関係学会北陸支部大会, 論文集,pp.374,1996.

[E.iida 1997] 飯田 栄治 , 國藤 進, “エージェント指向将棋ゲームシステム”, 人工知能基礎論研究会資料,SIG-FAI-9603,pp.24-29,Mar.1997.

[E.Iida 1998] 飯田 栄治, 國藤 進, 下平 博, 木村 正行, “N^2-1 パズルのスケールダウン解法”, 電子情報通信学会, 和文論文誌Ｄ －I,pp.504-514,1998.6.

[E.Iida 1998a] 飯田 栄治, 下平 博, “実時間意思決定システムに関する一提案”, 電子情報通信学会, 知能ソフトウェア工学研究会資料信学技報 Vol.98, No.6, pp.1-8,Apr.1998.

[E.Iida 1998b] 飯田 栄治, 下平 博, “実時間意思決定システムの一提案とその応用”, 人工知能学会, 第 12 回全国大会, 講演論文集,pp.528-529,Jun.1998.

[E.Iida 1998c] 飯田 栄治, 下平 博, “航空機パイロットの知識構造に関する一考察”, 日本認知科学会, 第 15 回全国大会, 講演論文集,pp.68-69,Jun.1998.

[E.Iida 1998d] Eiji Iida, Hiroshi Shimodaira, Susumu Kunifuji, Masayuki Kimura, “A System to Perform Human Problem Solving”, *The 5th International Conference on Soft Computing and Information / Intelligent Systems (IIZUKA'98)*,pp.941-946,Oct.1998.

[Forgy and McDermott,1977] C.L.Forgy and J.McDermott, “OPS, A Domain-independent Production System Language,” *International Joint Conference on Artficial Intelligence (IJCAI-77)* ,1977.

[Forgy 1979] C.L.Forgy, “OPS4 Users Manual,” *Carnegie Mellon University Technical Report*, CS-79-132,1979.

[Forgy 1981] C.L.Forgy, “OPS5 user's manual, Technical Report”, ,CSD,CMU,1981

[Forgy 1981] C.L.Forgy, “OPS5 User's Manual, CS-81-135” *Carnegie-Mellon University,*1981.

[Forgy 1985] C.L.Forgy, “OPS83 User's Manual and Report,” *Production Systems Technologies, Inc.*,1985

[Fujii 1996] 藤井 勝之, “ゲーム木分割探索におけるプロセッサ木の動的な組み替えの効果”, *Game Programming Workshop 96*, 論文集,pp.94–103,1996.

[Fikes 1971] Fikes,R.E. and Nilsson,N.J., “STRIPS:a new approach to the application of the theorem proving to problem solving” *Artificial Intelligence*,2(3/4),189-208,1971.

[Fukutomi 1992] Fukutomi ,S. ,Naito, N., Takizawa, Y., “An Integrated Operator Decision Aid System for Boiling Water Reactor Power Plants”, *Nuclear Technolgy*,Vol.99,pp.120-132,1992

[Furuta 1998] 古田一雄, “ヒューマンモデリングの現状と課題”, 人工知能学会誌 ,vol.13,No.3,May.1998.

[Griesmer et al. 1984] J. H. Griesmer, S. J. Hong, M. Karnaugh, J.K. Kastner, M. I. Schor, R. Ennis, D. Klein, K. Miliken, and H. M. Van Woerkom, “YES/MVS, A Continuous Real Time Expert system”, *National Conference on Artificial Inteligence (AAAI-84)*, pp.130-136,1984.

[Hart 1968] Hart,P.E. , Nilsson N.J. and Raphael,B., “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”, *IEEE Trans. on Systems Science and Cybernetics* ,Vol.SSC-4,No.2,pp.100-107,1968.

[Hirose 1996] 広瀬 正幸, 伊藤 琢巳 , 松原 仁 , “逆算法による詰め将棋の自動創作”, *Game Programming Workshop 96*, 論文集,pp.34–43,1996.

[H.Iida et al. 1994] 飯田 弘之, 乾 伸雄, 吉村 信宏, “コンピュータチェス”, サイエンス社 ,1994.

[H.Iida 1994] 飯田 弘之 , 小谷 善行 , “ゲームプレイングにおける教授戦略” , *Game Programming Workshop 94*, 論文集,pp.148–157,1994.

[Ishida 1991] 石田 亨 , “プロダクションシステムの高速化に関する研究”, 京都大学 *(*博士論文*)*,1991.

[Ishida 1992] 石田 亨, “実時間両方向探索”, *MACC'92*, 1992.

[Ishida 1996] 石田 亨, 新保 仁, “実時間探索による経路学習”, 人工知能学会誌,pp.411-419,Jan.1997.

[Ito 1995] 笠田 洋和, 山田 雅之, 松波 功二, 世木 博久, 伊藤 英則, “詰め将棋におけるゲーム木の並列探索とその評価”, 情報処理学会論文誌 ,Vol.36,No.11,pp.2531–2539,Nov.1995.

[Ito 1996] 立松 靖朗, 山田 雅之, 世木 博久, 伊藤 英則, 詰め将棋におけるプロセッサ稼働率を考慮したゲーム木並列探索 , 情報処理学会論文誌, Vol.37.No.9,pp.1745–1748,Sep.1996.

[JAL 1995] 日本航空（株）技術研究所ヒューマンファクターグループ, “ヒューマンファクターガイドブック”, 日本航空, Dec.1995.

[Johnson et al. 1879] W.W.Johnson, W.E.Storey, “Notes on the ‘15’Puzzle” *Am.J.Mathematics*, 1879.

[Katou 1982] 加藤 寛一郎, 大屋 昭男, 柄沢 研治, “航空機力学入門”, 東京大学出版会 ,1982.

[Katou 1988] 加藤 寛一郎, 大屋 昭男, 柄沢 研治, “工学的最適制御”, 東京大学出版会 ,1988.

[Kitamura 1996] 北村 泰彦, 寺西 憲一, 辰巳 昭治, “マルチエージェント実時間探索における組織化とその評価” 人工知能学会誌,pp.470-477,May,1996.

[Kobayashi 1990] 小林 重信他, “知識システムハンドブック”, オーム社,1990.

[Kolf 1990] R.E. Kolf, “Real-Time-Heuristic Search”, *Artificial Intelligence*,Vol.42, No.2-3,pp.189-211,1990.

[Kolf 1985a] R.E. Korf, “Depth-First Iternative-deepening An Admissible Tree Search”, *Artificial Intelligence*, Vol.27, No.1, pp.97-109, 1985.

[Kolf 1985b] R.E. Korf, “Learning to Solve Probrems by Searching for Macro-Operators”, *Pitman Advanced Publishing Program*, 1985.

[Kotani 1990] 小谷 善行, 吉川 竹四朗, 柿木 義一, 森田 和朗
“ コンピュータ将棋”, サイエンス社, 1990.

[Kowalski and Thomas,1983] T. Kowalski and D. Thmas, “The VLSI Design Automation Assistant, Prototype System”, *Design Automation Conference*,1983.

[KOUKUU 1983] 日本航空宇宙学会編, “航空宇宙工学便覧”, 日本航空宇宙学会 ,1983.

[KOUKUU 1989] (社) 日本航空宇宙工業会, “機上装置への人工知能の応用研究 vol.1”, 日本航空宇宙工業会, 革新航空機技術開発に関する研究調査, 1989.

[KOUKUU 1990] (社) 日本航空宇宙工業会, “機上装置への人工知能の応用研究 vol.2”, 日本航空宇宙工業会, 革新航空機技術開発に関する研究調査, 1990.

[Kuipers 1984] Benjamin J. Kuipers “Commonsense Reasoning about Causality : Deriving Behaviour from Structure”, *Artificial Intelligence*,Vol.24,No.1-3,pp.169-203,1984.

[Kuipers 1986] Benjamin J. Kuipers “Causal simulation”, *Artificial Intelligence*,Vol.26,pp.289-338,1986.

[Lind 1990] Lind,M. “Representation Goal And Functions of Complex System - An Introduction to Multilevel Flow Modeling” *Institute of Automatic Control Systems, Technical University of Denmark* , Report No.90-D-381,1990.

[Lind 1994] Lind,M., “Lind,M.: Modeling Goals and Functions of Complex Industrial Plants” *Applied Artificial Intelligence* ,Vol.8,No.2,pp.259-283,1994.

[Matsubara 1995a] 松原 仁 他 “小特集：ゲームプログラミング”, 人工知能学会誌, Vol.10.No.6,pp.835–859,Nov.1995.

[Matsubara 1995b] 松原 仁, “将棋の次の一手問題に関する考察”, *Game Programming Workshop 95*, 論文集, pp.66–74,1996.

[Matsubara 1996] 松原 仁, “コンピュータ将棋の進歩”, 共立出版,1996.

[Matsubara 1994] 松原仁, “将棋とコンピュータ” 共立出版,1994.

[Matsubara et al. 1993] 松原 仁, 橋田 浩一, “フレーム問題の疑似解決のためのヒューリスティックスとしての因果律” 講談社サイエンティフィック , 認知科学の発展 , 第 6 巻,Apr.,1993.

[Matsubara et al. 1990] 松原 仁, 橋田 浩一, “フレーム問題をどうとらえるか” 講談社サイエンティフィック , 認知科学の発展 , 第２巻,March.1990.

[McCarthy 1969] McCarthy ,J. and Hayes, P.J., “Some philosophical problems from the standpoint of artificial intelligence”, *Machine Intelligence*,vol.4,pp.463-502, 1969.

[McDermott 1982a] J.McDermott, “R1: A Rule-Based Configurer of Computer Systems”, *Artificial Intelligence*,Vol.19,pp.39-88,1982.

[McDermott 1982b] J.McDermott, “XSEL: A Computer Sales Person’s Assistant”, *Machine Intellegence*,Vol.10,pp.325-337,1982

[Mizoguchi 1997] 溝口, 池田, “オントロジー工学序説—内容指向研究の基礎技術　と理論の確立を目指して”, 人工知能学会誌, Vol.12,No.4,pp.559-570,1997.

[Newell 1980] Newell, A. “ Reasoning, problem solving and decision processes : The problem space as a fundamental category, R. Nickerson (ed)” *Erlbaum* , Attention and Performance VIII , 1980

[Newell 1969] Newell, A., “Heuristic Programming: III-Structured Problems, J. Aronofsky (ed)”, *Progress in Operations Research III*,pp.360-414,Wiley,1969.

[Newell 1973] Newell, A., “Production system : Models of control structures, W. C. Chase (ed) ”, *Visual Information Processing* , pp.463-526, Academic Press,1973.

[Newell 1981] Newell, A. and Rosenbloom, P., “Mechanisms of skill acquisition and Cognition and the law of practice, J. R. Anderson (ed) ”, *Learning and Cognition* , Erlbaum, 1981.

[Newell,Shaw,Simon 1963] Newell, Shaw, Simon , “GPS, a program that simulates human though” In CT, pp.279-293

[Niwa 1996] 丹羽, 鷲尾, “原子力プラントにおける安全機能の構造と定義”, *J.Nuclear Safety System*,No.3, pp.230-247,1997.

[Nilsson 1973] NIS J.NILSSON, “人工知能—問題解決のシステム論”, 合田 周平, 増田 一比古 共訳, May.1973.

[Nilsson 1983] NIS J.NILSSON, “人工知能の原理”, 白井 良明, 辻井 潤一, 佐藤 泰介 共訳,Jan.1983

[Nishioka 1998] 西岡 靖之 他, “製造を考慮した対話型設計支援システム”, 人工知能学会誌 ,vol.13,No.2,pp300-311,1998

[Nozaki 1990] 野崎 昭弘 , “ロジカルな将棋入門” 筑摩書房 ,1990.

[Ohga 1995] Ohga, Y., Seki,H., “An Information Offering System for Operation Support Based on Plant Functional Structure”, *J. Nuclear Science and Technology* ,Vol.32, No.8,pp. 727-739, 1995.

[Pell 1994] Barney Pell, 松原 仁 , “Metagamer の将棋への適用”, *Game Programming Workshop 94*, 論文集, pp.4–11,1994.

[Ratner et al. 1986] Daniel Ratner , Manfred Warmuth , “Finding A Shortest Solution For the N × N Extension of the 15-puzzle is Intractable” *AAAI-86,* pp.168–172,1986.

[Reinefeld1993] Alexander Reinefeld, “Complete Solution Of THE Eight-Puzzle and Benefit of Ordering in IDA” *IJICAI-93,* pp.248-353,1993.

[Sakuma 1996] 佐久間 秀武, “今なぜ人間中心の自動化か－その概念と実現性について”, 日本航空宇宙学会 , 第 44 巻, 第 504 号,Jan.1996.

[Seo 1995] 背尾 昌宏, “共謀数を応用した詰め将棋プログラムについて”, *Game Programming Workshop 95*, 論文集, pp.128–137,1995.

[Shofield1967] P.D.A.Shofield, “Complete Solution Of THE Eight-Puzzle”, *Machine Intelligence*, pp.125-133,1967.

[Shortliffe 1976] E. Shortliffe, “Computer-Based Medical Consultings, MYCIN”, *American Elsevier*, 1976.

[Shannon 1950] Shannon,C.E., “Programming a computer for playing chess”, *Philosophical Magazine(Series 7)* ,vol.41,pp.256-275.1950.

[S.Matsubara1997] 松原 繁夫, 石田 亨, “実時間探索探索に目標生成機構を組み込んだ実時間プランニング”, 人工知能学会誌,pp.90-99,Jan.1997.

[Soar 1994] 堀　雅洋, 池田　満　他, “小特集「Soar プロジェクト」”, 人工知能学会誌,pp.478-504,July.1994.

[Tokuda 1994] 徳田 浩　飯田 弘之 小谷 善行, “相手モデルを考慮したゲーム木探索法の実験と検証”, *Game Programming Workshop 94*, 論文集, pp.139–147,1994.

[Tokuda 1995] 徳田 浩, 小谷 善行 , “将棋におけるニューラルネットワークを用いた評価関数の生成実験”, *Game Programming Workshop 95*, 論文集, pp.47–56,1995.

本研究に関する発表論文

口頭発表・投稿中も含む。なお、査読付き論文には○印、国際会議での口頭発表には◎印を付ける。

一般講演

- 飯田栄治, 下平博, 木村正行, “ 整列オペレータによる８パズルの高速解法 ”, 電気関係学会北陸支部連合大会, 講演論文集 pp.452, 1994.10, 金沢大学
- 飯田栄治, 下平博, 木村正行,“ 強化学習機能を有するエージェントの衝突回避シミュレーション”, 電気関係学会北陸支部連合大会, 講演論文集 pp.451,1994.10, 金沢大学
- 飯田栄治, 下平博, 木村正行,“ 大局的戦略に基づくゲーム木探索の一提案 ”, 電気関係学会北陸支部連合大会, 講演論文集 pp.4374,1996.10, 富山高等専門学校
- 飯田栄治, 下平博,“ 実時間意思決定システムの一提案とその応用 ”, 人工知能学会第 12 回全国大会, 講演論文集 pp.528-529,1998.6.16 - 19, 早稲田大学国際会議場
- 飯田栄治, 下平博,“ 航空機パイロットの知識構造に関する一考察 ”, 日本認知科学会第 15 回全国大会, 講演論文集 pp.68-69,1998.6.25 - 27, 名古屋大学シンポジオンホール
- 飯田栄治, 下平博, “フェーズフローに基づく問題解決”, 数理モデル化と問題解決 第 6 回シンポジウム,2000.3.2 - 3.3, 同志社大学田辺キャンパス（発表予定）

研究会テクニカルリポート

- 飯田栄治, 下平博, 木村正行,“ ８パズルの高速解法 ”, 電子情報通信学会, コンピュテーション研究会資料 COMP94 54〜66 , 信学技報 Vol.94 ,No354,pp.51-59 ,11.18.94, 奈良先端科学技術大学院大学
- 飯田栄治, 國藤進, 下平博, 木村正行,“ エージェント指向将棋ゲームシステム ”, 人工知能学会, 人工知能基礎論研究会資料 SIG-FAI-9603,pp.24-29,1997.3.27〜28, 名古屋工業大学

- 飯田栄治, 下平博,“ 実時間意思決定システムに関する一提案 ”,　電子情報通信学会, 知能ソフトウェア工学研究会資料信学技報 Vol.98, No.6, pp.1-8,1998.4.18 , 東京電気大学

国際会議及び Proceedings

- ○ ◎ Eiji Iida, Hiroshi Shimodaira, Susumu Kunifuji, Masayuki Kimura “A System to Perform Human Problem Solving ”, The 5th International Conference on Soft Computing and Information / Intelligent Systems (IIZUKA'98) ,Fukuoka, Japan, 1998.10.16-20

- ○ ◎ Eiji Iida, Hiroshi Shimodaira, Susumu Kunifuji, Masayuki Kimura “A Problem Solving System based on Phase Flow ”, The 6th International Conference on Soft Computing and Information / Intelligent Systems (IIZUKA2000) ,Fukuoka, Japan, 2000.10.1 - 4 （発表予定）

国内ジャーナル

- ○ 飯田栄治, 國藤進, 下平博, 木村正行,“ N2 － 1 パズルのスケールダウン解法 ”,　電子情報通信学会, 和文論文誌Ｄ－ I pp.504-514,1998.6, 計算量とアルゴリズム小特集

- ○ 飯田栄治, 下平博,“ フェーズフローに基づく問題解決 ”, 電子情報通信学会, 和文論文誌Ｄ－ II　（投稿中）

Appendix

A. スマートな解の証明

B. $N^2 - 1$ パズルの計算量分析

C. $N^2 - 1$ パズルの追加実験

D. 追跡ゲームの実験に関する初期データ

E. 航空機操縦問題の初期設定データ

F. 駒落ち将棋システムへの応用

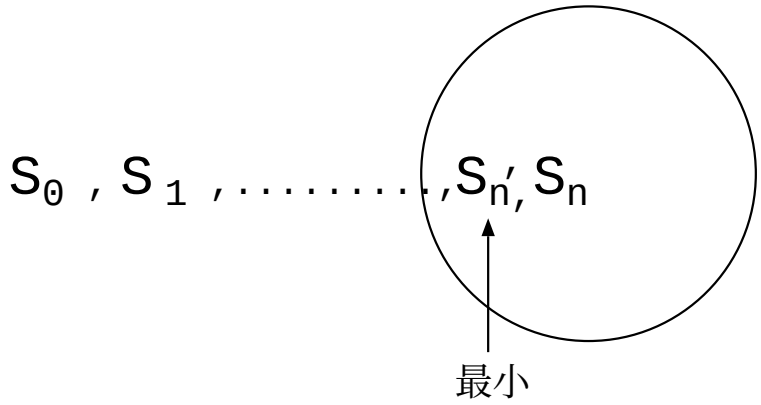
第 A 章

スマートな解の証明

定理 5 スマートな解 [Banerji 1983]
$s_0 \in S$に対する $P =< S, W, R >$ の解が存在するとき、必ず s_0に対する Pの「スマートな解」が存在する。

証明

$s_0 \in S$に対する問題 P の解を、$(s_0, s_1, ..., s_n)$ とする。非負整数の集合 $N = \{i | S_i \in W\}$ とし、また、$n \in N$とすれば、$N = \phi$である。N は、非負整数から構成されるので、$S_{n'} \in W$ なる最小 n' が存在するはずである。

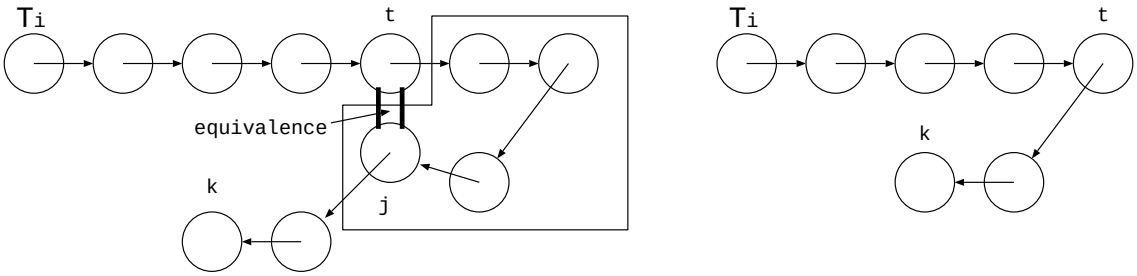


故に、$(\forall i)(i < n' \rightarrow not(S_i \in W))$ が成り立つ。

解 $T = (s_0, s_1, ..., s_{n'})$ とし、$T_0 = T$とすれば、少なくとも１つの解が存在することになる。解を一般化し、$T_i = (s_0^i, s_1^i, ..., s_k^i)$ とする。ここで、$\{t|j < t \& s_j^i \neq s_t^i\}$ where $(\exists j)(j > t \& s_j^i = s_t^i)$

は、解の状態列において同じものが繰り返し出現する場合における、状態番号の集合である。

t は、s_t^iがあとになって再現する最小の整数。仮に、j を $s_j^i = s_t^i$なる最大の整数とする。T_{i+1}を以下のように定める。0 ≦ p ≦ t なるすべての p に対して $s_p^{i+1} = s_p^i$ 0¡m ≦ k-j に対し $s_{t+m}^{i+1} = s_{j+m}^i$ ここで、T_{i+1}は、T_iから数えて t+1 番目の状態から j 番目までの状態を除いた状態列である。$s_{t+k-j}^{i+1} = s_k^i \in W$ よって、T_{i+1}も解である。



$s_p^{i+1} = s_p^i$

よって、すべての $p < t$ に対して、$s_p^{i+1} R s_{p+1}^{i+1}$ である。p=t に対して、$s_{t+1}^{i+1} = s_{j+1}^i$& $s_t^{i+1} = s_t^i = s_j^i$ だから、$s_t^{i+1} R s_{t+1}^{i+1}$が成り立つ。全ての $p < t$ に対しては、$s_{t+m}^{i+1} = s_{j+m}^i R s_{j+m+1}^i = s_{t+m+1}^{i+1}$ である。

従って、各T_iは、解である。また、T_{i+1}の項数はT_iより小さい。T_0が有限個の項しか持たないから、T_{p+1}が定義できないようなT_pが存在する。即ち、T_pは、すべての異なった項からなっており、「スマートな解」である。

第 B 章

$N^2 - 1$ パズルの計算量分析

B.1　アルゴリズムとその分析

B.1.1　アルゴリズムの比較

実時間探索

実時間探索の一般的なアルゴリズムは図 B.1のとおりである。
実時間探索の問題解決に要するコスト C_{RT} を考える。

N : パズルのサイズ
N_{GOAL} : ゴールへ至る手数
C_1 : 1 ノードの 1 回当たりの評価コスト
C_2 : 1 ノード当たりの展開コスト
D : 実時間探索深さ
B : 分岐因子　　とすれば、

特に、$N_{GOAL} \geq D$のときに注目して考えると、

$$C_{RT} = (C_1 + C_2)B^D(N_{GOAL} - D + 1) \tag{B.1}$$

となる。

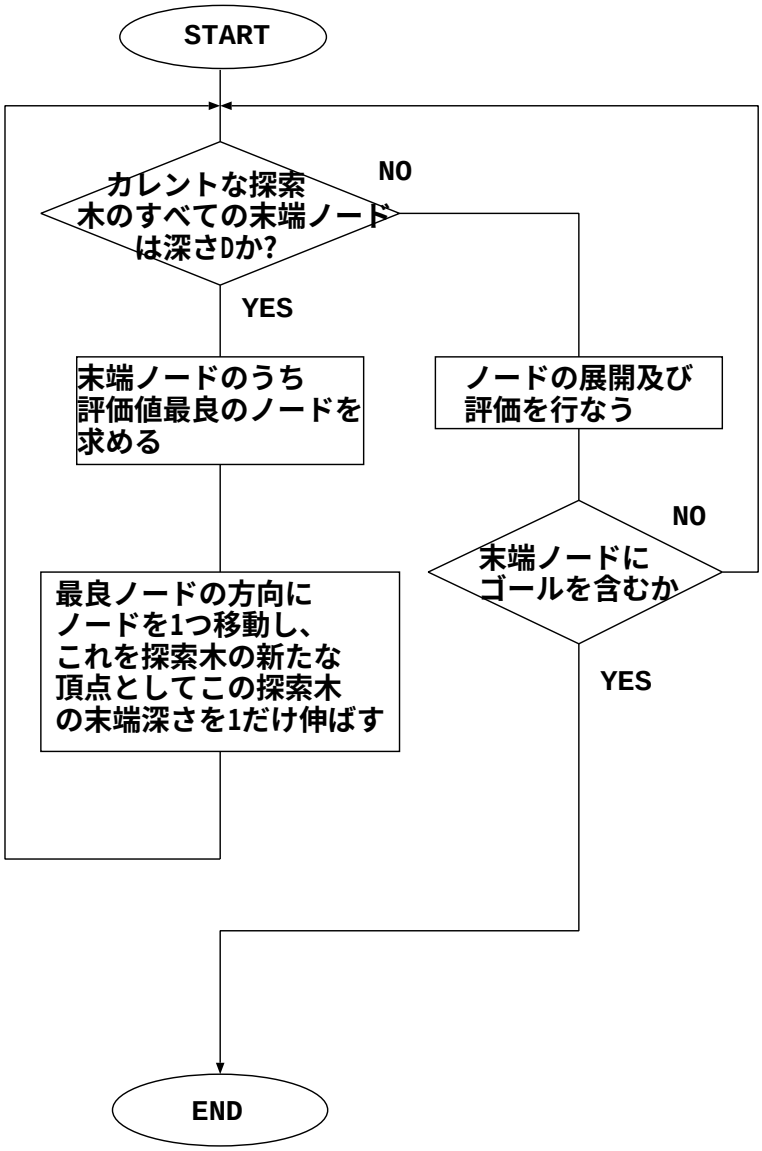


図 B.1: 実時間探索アルゴリズム

提案解法

提案解法をアルゴリズム化すると図B.2のようになる。

ここで、問題解決に要するコスト C_{SD} を考える。

N：パズルのサイズ

N_{GOAL}：ゴールへ至る手数

C_1：タイル１つ当たりの評価コスト

C_2：最小値決定コスト

C_3：オペレータの１回当たりの適用コスト

C_4：ゴール変換コスト

C_5：特殊手順を要する部分の解決コスト

N_{op}：オペレータの適用回数　　とすれば、

•コスト最小のL字形領域を選択するコストは、４つのL字形を評価しなければならないので、$4C_1N^2 + C_2$であり、問題サイズ１～Nまでの総コスト C_L は、

$$C_L \leq \sum_{M=1}^{N} (4C_1M^2 + C_2) = \frac{2C_1N(N+1)(2N+1)}{3} + C_2N \tag{B.2}$$

•各オペレータは複数手順からなる $(N_{op} \leq N_{GOAL})$ ので、オペレータの全適用コスト C_{op} は

$$C_{op} \leq C_3N_{GOAL} \tag{B.3}$$

• C_4,C_5 はスケールダウンごとに発生するものとして、それらの総コスト C_{45} は、

$$C_{45} \leq C_4N + 2C_5N \tag{B.4}$$

よって、C_{SD} は、(B.2)+(B.3)+(B.4) により

$$C_{SD} \leq \frac{2C_1N(N+1)(2N+1)}{3} + C_2N + C_3N_{GOAL} + C_4N + 2C_5N \tag{B.5}$$

となる。

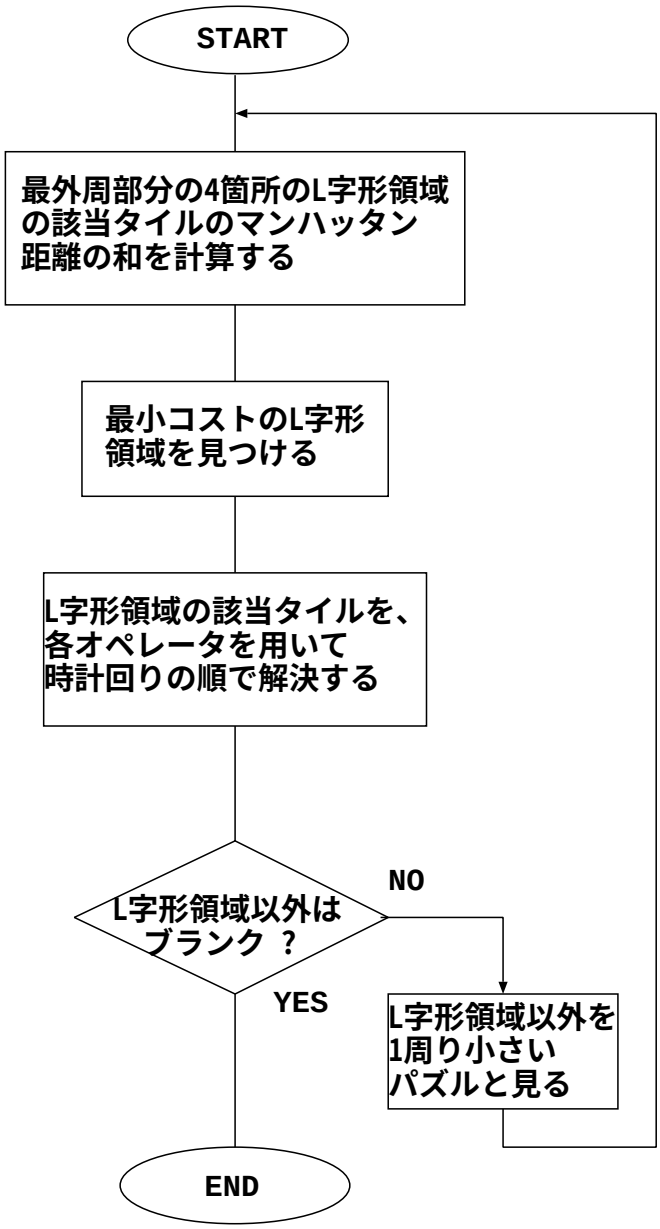


図 B.2: 提案手法のアルゴリズム

計算量の分析

実時間探索では、一般に得られる解の長さ (N_{GOAL}) は、探索深さが深くなるほど短くなり、最終的に最適解に落ち着く。しかし、(B.1) 式から分かるように、１回当たりの探索深さに対して計算コストが指数オーダーで上昇する。提案手法では、探索木を生成しないことから、計算コストは (B.5) 式より、問題サイズNの３乗のオーダーでおさえるこ

とができる。

B.1.2 解の長さに関する分析

探索による解法では、ゴール状態に到達するまで、解の長さがわからないが、提案手法では、未知のサイズ (N) の問題の解の長さの上限値 (upper bound) を知ることができる。以下、ワーストケースを想定した解の長さを考える。

a. 移動対象タイル T にブランクが隣接している場合、図 5.4 (a) のオペレータのように、T が１マス動くのに最大で５手要する。また、最大移動距離は、$2N-2$ なので、移動手数 N_T は

$$N_T \leq 10(N-1) \tag{B.6}$$

b. ブランクが T から離れている場合、ブランクを T に隣接させるために要する手数 N_B は、

$$N_B \leq 2N-2 \tag{B.7}$$

c. a,b より１つのタイルのホームポジションまでの最悪の移動コスト N_H は

$$N_H = N_T + N_B \leq 12(N-1) \tag{B.8}$$

d. Ｌ字部分の整列手数 N_L は、対象となるタイルの総数が $2N-1$ なので、(B.8) 式から、

$$N_L \leq 12(N-1)(2N-1) \tag{B.9}$$

e. サイズ N から 1 までの N_L の和Ｓを考える。これが、解長さの上限値となる。

$$\begin{aligned} S &\leq \sum_{m=1}^{N} 12(m-1)(2m-1) \\ &= 8N^3 - 6N^2 - 2N \end{aligned} \tag{B.10}$$

第 C 章

$N^2 - 1$ パズルの追加実験

C.1 実験及び結果

C.1.1 アルゴリズムの比較

実時間探索

8 パズル、15 パズル 100 題、24 パズル 40 題について、問題を解かせた。15 パズルの例題は Korf の IDA^* の例題と同一であり、8 パズルと 24 パズルについては、今回、新たにランダムなタイル配置の例題を作成した。問題解決知識を、$SUN\ SPARCstation$ 上の $JPS1$ に実装し、実験を行なった。表 C.1に解長さの平均を示す。また、参考として、実験に用いた例題および実験結果を、表 C.2表 C.3表 C.4に示す。

表 C.1: 平均の解長さ

問題	提案手法	備考
8puzzle	28.7	
15puzzle	135	←最適解の解の長さの平均:53
24puzzle	297	

C.1.2　評価

Korf の実時間探索 (RTA^*) の実験結果と提案方式の C.1 の結果を重ね合わせてて比較した。得られた解の長さを解の“ 質 ”ということにし、最適解に近いほど質の高い解（良質）であるものとする。実時間探索では、一般に、先読み探索の１回当たりの深さが深くなるほど、コストが指数的に上昇し解の質も高くなる。 逆に、先読み深さが浅ければ、計算負荷が軽減（コスト低下）するが、解の質も低下する。

我々の解は、15 パズルについては、実時間探索の探索深さ 20 以上に設定した場合と同等の解の質であった。また、24 パズルでは提案手法の解が予想以上に質の高い解であり、Korf の実験の範囲を超えていた。データから推定してみると、深さ 30 以上に相当することが分かった。因みに、提案方式では、24 パズルでも、ほとんど 30 秒以下で解くことができたが、RTA^*で置き換えて考えてみると、我々のマシンと同等性能のマシンで数時間（計算量は、付録にて検討する）を要する。

つぎに、解長さの予測式と、実験結果および最適解の関係について見てみる。ここで、対象とするサイズは、３、４および５であり、サイズ３については、すでに、我々が８パズルについて報告 [E.Iida1994] した実験値と荒屋らによって報告 [Araya1996] された最適解長さを用いている。これらを同一グラフ上にプロットすると、図 C.2のようになる。

24 パズルの最適解は分かっていないので記入していない。提案手法の解の質については、更にサイズの大きい問題に対しても、このグラフの傾向からある程度推定できる。今回の範囲では、提案手法の実験値は上限値（試算式は付録に示す）の約 1/3、逆に最適解の 2 倍前後であった。

C.1.3　まとめ

ここで、用いたフェーズフローの流れは、２４パズル程度のものであれば、人間においても目分量で利用できる単純な方針である。にも関わらず、探索計算を行なうと数時間を要する。これは、人間的な問題解決戦略がかなり強力なツールとなり得ることを示しているのではないかと思われる。

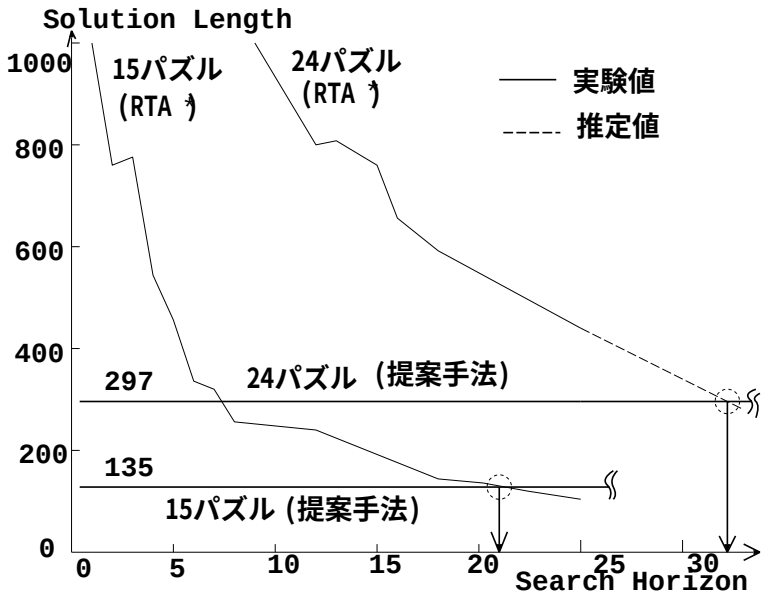


図 C.1: RTA^*と提案手法

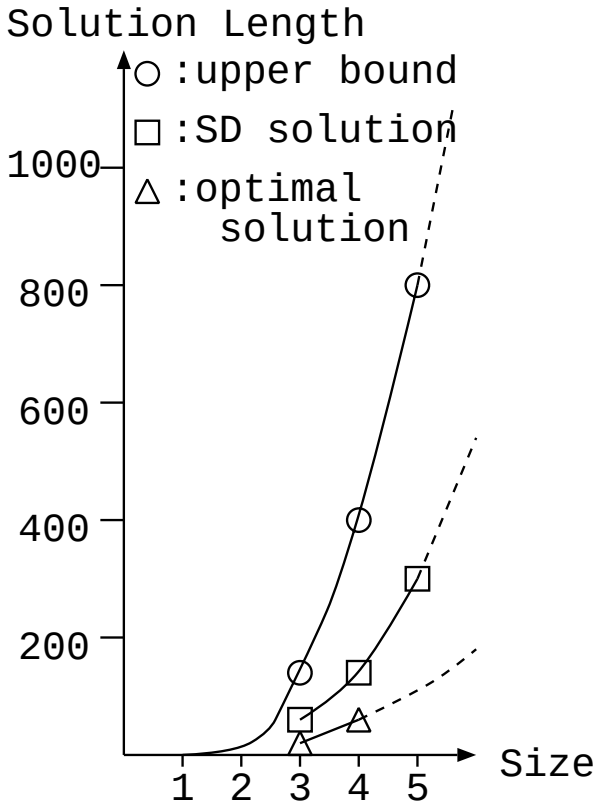


図 C.2: 上限値、提案手法 および 最適解の解長さ

表 C.2: 8 パズルに関する実験例題及び結果

番号	問題	A アルゴリズムによる結果（参考）			今回提案する高速解法
		手数	展開回数	run time	手数
1	[2,6,3,7,5,8,4,1,0]	16	51	30.73	24
2	[3,8,7,2,5,4,6,1,0]	26	67	43.64	32
3	[2,6,4,5,7,1,8,3,0]	22	23	7.89	22
4	[3,1,6,2,7,5,4,8,0]	26	76	61.92	36
5	[6,1,3,2,4,7,8,5,0]	24	76	59.49	24
6	[8,6,7,1,2,4,3,5,0]	28	29	11.69	38
7	[6,5,1,2,7,3,8,4,0]	32	59	38.45	22
8	[1,8,3,5,6,2,7,4,0]	16	19	5.75	24
9	[4,2,6,8,7,5,1,3,0]	24	30	12.41	38
10	[3,5,8,4,7,2,1,6,0]	26	38	15.11	36
11	[5,8,2,7,6,1,3,4,0]	24	30	11.63	32
12	[7,5,8,3,1,4,2,6,0]	20	25	8.83	22
13	[1,5,7,4,6,3,2,8,0]	28	44	21.76	36
14	[1,5,8,6,4,2,7,3,0]	24	30	11.67	26
15	[2,8,5,1,6,4,7,3,0]	26	38	18.61	28
16	[3,8,5,1,6,4,2,7,0]	30	48	24.27	32
17	[4,7,2,5,6,3,1,8,0]	30	72	57.04	32
18	[6,5,8,3,4,7,1,2,0]	16	18	5.34	36
19	[4,7,5,6,2,8,3,1,0]	20	25	9.01	26
20	[8,1,6,2,3,7,5,4,0]	28	99	107.46	20
21	[4,6,5,8,1,7,2,3,0]	24	74	56.96	32
22	[5,4,2,1,6,3,8,7,0]	34	132	159.72	30
23	[1,5,7,3,8,2,6,4,0]	30	74	60.57	20
24	[4,6,5,1,2,3,7,8,0]	24	43	21.35	28
25	[1,7,2,5,4,6,8,3,0]	20	70	47.38	26
26	[5,7,2,4,8,1,6,3,0]	32	110	131.24	36
27	[7,1,2,5,4,3,8,6,0]	16	20	6.28	32
28	[4,1,5,2,6,8,3,7,0]	20	38	18.09	24
29	[5,3,6,1,2,4,8,7,0]	28	73	55.74	28
30	[6,1,5,3,8,7,2,4,0]	24	29	12.49	22
31	[7,4,8,2,1,5,3,6,0]	30	145	224.78	36
32	[8,1,5,3,2,7,6,4,0]	22	27	10.64	28
33	[1,3,6,5,8,4,2,7,0]	18	23	8.36	22
34	[7,5,8,2,1,4,6,3,0]	22	29	10.80	32
35	[5,4,3,1,8,7,2,6,0]	28	51	27.98	34
36	[4,5,1,2,8,6,3,7,0]	32	165	275.64	30
37	[2,6,8,1,4,3,7,5,0]	34	35	78.12	38
38	[8,1,3,7,5,2,4,6,0]	28	126	160.26	20
39	[4,6,1,3,5,8,7,2,0]	26	63	43.45	18
40	[6,1,7,4,8,5,2,3,0]	34	129	174.66	30
	合 計	1012	2353	2147.45	1150
	平 均	25.3	58.825	53.687	28.75

表 C.3: 15 パズル実験例題

No.	Problem	No.	Problem	Remarks
1	[e,d,f,7,b,c,9,5,6,0,2,1,4,8,a,3]	51	[a,2,8,4,f,0,1,e,b,d,3,6,9,7,5,c]	
2	[d,5,4,a,9,c,8,e,2,3,7,1,0,f,b,6]	52	[a,8,0,c,3,7,6,2,1,e,4,b,f,d,9,5]	
3	[e,7,8,2,d,b,a,4,9,c,5,0,3,6,1,f]	53	[e,9,c,d,f,4,8,a,0,2,1,7,3,b,5,6]	
4	[5,c,a,7,f,b,e,0,8,2,1,d,3,4,9,6]	54	[c,b,0,8,a,2,d,f,5,4,7,3,6,9,e,1]	
5	[4,7,e,d,a,3,9,c,b,5,6,f,1,2,8,0]	55	[d,8,e,3,9,1,0,7,f,5,4,a,c,2,6,b]	
6	[e,7,1,9,c,3,6,f,8,b,2,5,a,0,4,d]	56	[3,f,2,5,b,6,4,7,c,9,1,0,d,e,a,8]	
7	[2,b,f,5,d,4,6,7,c,8,a,1,9,3,e,0]	57	[5,b,6,9,4,d,c,0,8,2,f,a,1,7,3,e]	
8	[c,b,f,3,8,0,4,2,6,d,9,5,e,1,a,7]	58	[5,0,f,8,4,6,1,e,a,b,3,9,7,c,2,d]	
9	[3,e,9,b,5,4,8,2,d,c,6,7,a,1,f,0]	59	[f,e,6,7,a,1,0,b,c,8,4,9,2,5,d,3]	
10	[d,b,8,9,0,f,7,a,4,3,6,e,5,c,2,1]	60	[b,e,d,1,2,3,c,4,f,7,9,5,a,6,8,0]	
11	[5,9,d,e,6,3,7,c,a,8,4,0,f,2,b,1]	61	[6,d,3,2,b,9,5,a,1,7,c,e,8,4,0,f]	
12	[e,1,9,6,4,8,c,5,7,2,3,0,a,b,d,f]	62	[4,6,c,0,e,2,9,d,b,8,3,f,7,a,1,5]	
13	[3,6,5,2,a,0,f,e,1,4,d,c,9,8,b,7]	63	[8,a,9,b,e,1,7,f,d,4,0,c,6,2,5,3]	
14	[7,6,8,1,b,5,e,a,3,4,9,d,f,2,0,c]	64	[5,2,e,0,7,8,6,3,b,c,d,f,4,a,9,1]	
15	[d,b,4,c,1,8,9,f,6,5,e,2,7,3,a,0]	65	[7,8,3,2,a,c,4,6,b,d,5,f,0,1,9,e]	
16	[1,3,2,5,a,9,f,6,8,e,d,b,c,4,7,0]	66	[b,6,e,c,3,5,1,f,8,0,a,d,9,7,4,2]	
17	[f,e,0,4,b,1,6,d,7,5,8,9,3,2,a,c]	67	[7,1,2,4,8,3,6,b,a,f,0,5,e,c,d,9]	
18	[6,0,e,c,1,f,9,a,b,4,7,2,8,3,5,d]	68	[7,3,1,d,c,a,5,2,8,0,6,b,e,f,4,9]	
19	[7,b,8,3,e,0,6,f,1,4,d,9,5,c,2,a]	69	[6,0,5,f,1,e,4,9,2,d,8,a,b,c,7,3]	
20	[6,c,b,3,d,7,9,f,2,e,8,a,4,1,5,0]	70	[f,1,3,c,4,0,6,5,2,8,e,9,d,a,7,b]	
21	[c,8,e,6,b,4,7,0,5,1,a,f,3,d,9,2]	71	[5,7,0,b,c,1,9,a,f,6,2,3,8,4,d,e]	
22	[e,3,9,1,f,8,4,5,b,7,a,d,0,2,c,6]	72	[c,f,b,a,4,5,e,0,d,7,1,2,9,8,3,6]	
23	[a,9,3,b,0,d,2,e,5,6,4,7,8,f,1,c]	73	[6,e,a,5,f,8,7,1,3,4,2,0,c,9,b,d]	
24	[7,3,e,d,4,1,a,8,5,c,9,b,2,f,6,0]	74	[e,d,4,b,f,8,6,9,0,7,3,1,2,a,c,5]	
25	[b,4,2,7,1,0,a,f,6,9,e,8,3,d,5,c]	75	[e,4,0,a,6,5,1,3,9,2,d,f,c,7,8,b]	
26	[5,7,3,c,f,d,e,8,0,a,9,6,1,4,2,b]	76	[f,a,8,3,0,6,9,5,1,e,d,b,7,2,c,4]	
27	[e,1,8,f,2,6,0,3,9,c,a,d,4,7,5,b]	77	[0,d,2,4,c,e,6,9,f,1,a,3,b,5,8,7]	
28	[d,e,6,c,4,5,1,0,9,3,a,2,f,b,8,7]	78	[3,e,d,6,4,f,8,9,5,c,a,0,2,7,1,b]	
29	[9,8,0,2,f,1,4,e,3,a,7,5,b,d,6,c]	79	[0,1,9,7,b,d,5,3,e,c,4,2,8,6,a,f]	
30	[c,f,2,6,1,e,4,8,5,3,7,0,a,d,9,b]	80	[b,0,f,8,d,c,3,5,a,1,4,6,e,9,7,2]	
31	[c,8,f,d,1,0,5,4,6,3,2,b,9,7,e,a]	81	[d,0,9,c,b,6,3,5,f,8,1,a,4,e,2,7]	
32	[e,a,9,4,d,6,5,8,2,c,7,0,1,3,b,f]	82	[e,a,2,1,d,9,8,b,7,3,6,c,f,5,4,0]	
33	[e,3,5,f,b,6,d,9,0,a,2,c,4,1,7,8]	83	[c,3,9,1,4,5,a,2,6,b,f,0,e,7,d,8]	
34	[6,b,7,8,d,2,5,4,1,a,3,9,e,0,c,f]	84	[f,8,a,7,0,c,e,1,5,9,6,3,d,b,4,2]	
35	[1,6,c,e,3,2,f,8,4,5,d,9,0,7,b,a]	85	[4,7,d,a,1,2,9,6,c,8,e,5,3,0,b,f]	
36	[c,6,0,4,7,3,f,1,d,9,8,b,2,e,5,a]	86	[6,0,5,a,b,c,9,2,1,7,4,3,e,8,d,f]	
37	[8,1,7,c,b,0,a,5,9,f,6,d,e,2,3,4]	87	[9,5,b,a,d,0,2,1,8,6,e,c,4,4,3,f]	
38	[7,f,8,2,d,6,3,c,b,0,4,a,9,5,b,4]	88	[f,2,c,b,e,d,9,5,1,3,8,7,0,a,6,4]	
39	[9,0,4,a,1,e,f,3,c,6,5,7,b,d,8,2]	89	[b,1,7,4,a,d,3,8,9,e,0,f,6,5,2,c]	
40	[b,5,1,e,4,c,a,0,2,7,d,3,9,f,6,8]	90	[5,4,7,1,b,c,e,f,a,d,8,6,2,0,9,3]	
41	[8,d,a,9,b,3,f,6,0,1,2,e,c,5,4,7]	91	[9,7,5,2,e,f,c,a,b,3,6,1,8,d,0,4]	
42	[4,5,7,2,9,e,c,d,0,3,6,b,8,1,f,a]	92	[3,2,7,9,0,f,c,4,6,b,5,e,8,d,a,1]	
43	[b,f,e,d,1,9,a,4,3,6,2,c,7,5,8,0]	93	[d,9,e,6,c,8,1,2,3,4,0,7,5,a,b,f]	
44	[c,9,0,6,8,3,5,e,2,4,b,7,a,1,f,d]	94	[5,7,b,8,0,e,9,d,a,c,3,f,6,1,4,2]	
45	[3,e,9,7,c,f,0,4,1,8,5,6,b,a,2,d]	95	[4,3,6,d,7,f,9,0,a,5,8,b,2,c,1,e]	
46	[8,4,6,1,e,c,2,f,d,a,9,5,3,7,0,b]	96	[1,7,f,e,2,6,4,9,c,b,d,3,0,8,5,a]	
47	[6,a,1,e,f,8,3,5,d,0,2,7,4,9,b,c]	97	[9,e,5,7,8,f,1,2,a,4,d,6,c,0,b,3]	
48	[8,b,4,6,7,3,a,9,2,c,f,d,0,1,5,e]	98	[0,b,3,c,5,2,1,9,8,a,e,f,7,4,d,6]	
49	[a,0,2,4,5,1,6,c,b,d,9,7,f,3,e,8]	99	[7,f,4,0,a,9,2,5,c,b,d,6,1,3,e,8]	
50	[c,5,d,b,2,a,0,9,7,8,4,3,e,6,f,1]	100	[b,4,0,8,6,a,5,d,c,7,e,3,1,2,9,f]	

表 C.4: 24 パズル実験結果（SD 解法）

番号	問題	タイル移動回数
1	[5,1,g,6,l,a,b,2,m,f,7,n,e,8,4,h,3,j,c,k,d,o,0,9,i]	260
2	[k,3,b,g,n,o,f,m,1,4,a,d,7,0,e,l,8,j,6,5,2,c,i,9,h]	329
3	[b,5,f,j,7,m,a,l,1,c,6,g,4,k,e,i,2,n,3,o,9,h,d,0,8]	279
4	[a,j,9,k,8,i,3,f,4,g,6,b,l,c,7,h,0,2,m,n,1,d,o,e,5]	284
5	[6,d,j,e,1,f,3,4,g,k,8,9,n,7,b,l,h,2,i,o,c,0,a,m,5]	305
6	[h,4,j,g,9,5,l,f,3,i,k,b,2,m,a,6,n,d,8,0,c,7,e,o,1]	238
7	[e,o,3,n,c,l,6,5,h,m,d,1,0,4,b,7,f,8,k,2,g,9,j,a,i]	318
8	[3,o,a,n,g,4,f,k,5,l,0,2,b,c,1,8,j,6,h,m,e,7,d,i,9]	308
9	[6,0,7,l,c,o,f,m,2,h,1,a,b,k,3,e,n,4,g,d,5,9,j,8,i]	293
10	[j,a,9,g,4,5,d,i,8,c,0,1,n,e,l,7,o,k,6,3,h,2,b,f,m]	259
11	[o,a,3,8,l,5,n,b,9,g,m,4,h,k,2,1,c,6,e,0,f,7,j,d,i]	271
12	[9,h,5,n,6,g,3,d,j,o,4,8,i,2,e,f,b,m,c,0,a,1,k,7,l]	245
13	[6,0,c,i,7,h,9,l,g,o,3,m,k,4,1,f,a,2,d,b,j,e,n,8,5]	343
14	[l,g,n,1,0,o,j,b,a,d,9,k,5,7,b,3,i,h,f,4,c,m,2,e,8]	344
15	[6,h,o,d,5,n,8,0,g,l,3,1,b,j,4,i,m,e,9,c,f,7,k,2,a]	305
16	[f,4,d,m,7,l,c,6,e,o,h,j,k,2,8,5,g,a,n,i,0,1,9,b,3]	300
17	[8,l,n,3,j,o,2,b,a,i,9,f,0,m,6,4,1,g,7,k,d,5,c,h,e]	276
18	[b,9,4,f,k,j,a,2,8,3,7,n,g,i,h,1,6,o,c,e,d,0,m,5,l]	311
19	[2,k,8,j,a,l,c,o,4,h,f,5,g,9,n,3,i,6,0,d,7,b,m,e,1]	242
20	[m,9,n,3,o,1,k,d,6,c,g,a,4,j,2,7,0,e,8,f,i,5,b,l,h]	272
21	[e,n,3,j,c,8,m,2,b,a,7,d,h,o,0,9,i,k,4,g,l,5,1,f,6]	326
22	[0,1,m,f,k,a,l,2,c,6,9,h,5,e,j,n,g,b,3,8,7,d,4,i,o]	300
23	[6,0,k,l,1,a,h,8,o,e,9,i,g,4,f,7,m,3,d,5,j,c,n,2,b]	295
24	[e,6,c,4,f,3,d,7,n,5,h,1,l,b,0,2,m,i,o,g,a,j,9,k,8]	296
25	[3,f,8,l,c,n,9,h,2,i,7,g,4,0,a,m,5,o,b,k,1,e,j,d,6]	292
26	[2,b,n,d,8,m,3,f,l,5,k,6,c,7,e,o,1,0,a,i,9,g,j,4,h]	261
27	[k,4,h,5,6,l,d,7,g,9,c,a,i,2,b,m,j,n,3,o,e,0,1,f,8]	341
28	[5,k,f,7,1,d,l,3,o,m,4,0,c,a,9,h,i,g,n,8,6,j,2,b,e]	278
29	[b,a,n,4,g,k,2,m,f,o,1,e,5,9,j,8,i,d,3,h,c,l,7,0,6]	287
30	[d,i,8,1,g,7,h,2,b,m,a,0,5,f,6,j,e,n,4,k,c,l,3,9,o]	304
31	[h,4,c,6,e,k,9,b,n,5,m,1,f,7,3,d,i,0,2,8,a,o,g,j,l]	279
32	[i,8,b,m,c,2,j,d,3,g,4,f,a,1,k,7,l,0,e,o,6,5,n,h,9]	332
33	[d,4,g,l,9,2,a,k,m,e,j,8,5,b,1,f,7,i,c,0,3,h,n,o,6]	296
34	[j,m,b,g,h,4,n,1,0,i,c,a,6,e,d,3,7,f,8,2,9,l,5,k,o]	309
35	[9,e,m,1,b,a,c,j,g,5,d,l,6,0,2,o,k,n,8,3,h,4,7,f,i]	359
36	[2,e,6,c,l,h,5,0,1,9,b,3,i,8,o,7,m,d,4,f,k,j,g,a,n]	288
37	[c,i,8,l,n,k,e,4,d,3,6,7,o,0,j,5,g,1,f,h,2,b,a,m,9]	294
38	[f,l,e,i,8,n,g,k,5,2,6,h,4,7,a,o,0,d,9,j,c,b,3,1,m]	305
39	[1,4,0,c,b,m,h,2,k,n,l,f,9,8,d,o,e,a,j,3,6,i,5,g,7]	289
40	[6,d,k,5,2,b,c,m,1,f,9,g,j,n,h,8,l,o,3,7,e,0,a,4,i]	366

第 D 章

追跡ゲームの実験に関する初期データ

D.1　データの定義

本実験に使用した初期配置は、ランダム設定された４０ケースのデータである。ゲーム実行に際しの初期配置データは、プログラムにてメモリーにロードされる。次頁以降、-1,0,1,2,3,4 の６種類の数字にて盤面の状態を表す。それぞれの意味は既に述べたとおりである。なお、データの座標は、以下のように定義した。

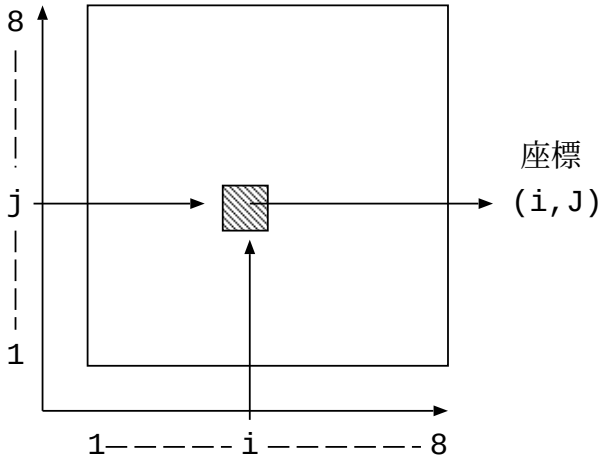


図 D.1: 追跡ゲームデータ定義

次頁以降データを示す。

D.2 データ

```
data 1      1, 0, 0, 0, 0, 0, 0, 2,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0,-1, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            4, 0, 0, 0, 0, 0, 0, 3,

data 2      0, 0, 0, 0, 0, 0, 0, 0,
            0,-1, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 2, 0, 0,
            0, 0, 0, 0, 1, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 3, 0, 0, 0, 4, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,

data 3      0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 1, 0, 0, 0, 0,
            0,-1, 0, 0, 0, 0, 4, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 3, 0,
            2, 0, 0, 0, 0, 0, 0, 0,

data 4      0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0,-1, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 1, 0, 0, 0, 0,
            0, 2, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 3, 0, 0, 0, 4, 0, 0,

data 5      0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0,-1, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 2, 0, 0, 0, 0, 0, 3,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 4, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 1, 0, 0,

data 6      0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 1, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 2, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 3, 0, 0, 0, 0, 0, 4,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0,-1, 0, 0, 0, 0,

data 7      0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 4, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 1, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 3, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0,-1, 0, 0, 0, 0, 2, 0,

data 8      0, 0, 0, 0, 0,-1, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 4, 0, 0,
            0, 2, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            3, 0, 0, 0, 0, 1, 0, 0,

data 9      0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 1, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 4, 0, 0,
            0, 0, 0, 0, 0, 0, 0,-1,
            0, 0, 2, 0, 3, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,

data 10     0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 1, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0,-1, 2, 0, 0, 0,
            0, 0, 3, 4, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
            0, 0, 0, 0, 0, 0, 0, 0,
```

```
data 11         0, 0, 0, 0, 0, 0, 0, 2,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0,-1, 0, 0,
                0, 1, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 4, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 3, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 12         0, 0, 0, 0, 0, 0, 0, 0,
                0,-1, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 0,
                0, 3, 0, 0, 0, 4, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 13         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0,-1, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 4, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 3, 0,
                2, 0, 0, 0, 0, 0, 0, 0,

data 14         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 0,
                0, 2, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0,-1,
                0, 3, 0, 0, 0, 4, 0, 0,

data 15         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0,-1, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 2, 0, 0, 0, 0, 0, 3,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 4, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 0,

data 16         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 3, 0, 0,-1, 0, 0, 4,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 17         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 4, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 3, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
               -1, 0, 0, 0, 0, 0, 2, 0,

data 18         0, 3, 0, 0, 0,-1, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 2, 0, 4, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 0,

data 19         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 4,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0,-1,
                0, 0, 2, 0, 3, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 20         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 4, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 2, 0,-1, 0, 3,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 1, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
```

```
data 21         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0,-1, 0, 0, 0,
                0, 3, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 1, 0, 2, 0,
                0, 4, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 22         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 4, 0,-1, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 3, 0, 0, 0,

data 23         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 4, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 3, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 2, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 0,-1, 0, 0,

data 24         0, 0, 0, 0,-1, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 1, 0, 2, 0, 3, 0, 4,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 25         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 1, 0, 0, 0,
                0, 0, 0, 4,-1, 3, 0, 0,
                0, 0, 0, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 26         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 0, 0, 0, 2,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0,-1, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 4, 0, 0, 0, 3,
                0, 0, 0, 0, 0, 0, 0, 0,

data 27         4, 0, 0, 0, 3, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0,-1, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                1, 0, 0, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 28         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 4, 0, 0, 0,
                0, 0, 0, 3, 0, 0, 0, 0,
                0, 0, 2, 0, 0, 0, 0, 0,
                0, 1, 0,-1, 0, 0, 0, 0,

data 29         0, 0, 0, 0, 3, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 0,-1, 0, 0, 0,
                0, 0, 0, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 4, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 30         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 4, 0, 0, 0,
                0, 0, 0, 3, 0, 1, 0, 0,
                0, 0, 2, 0, 0, 0, 0, 0,
                0, 0, 0,-1, 0, 0, 0, 0,
```

```
data 31         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 1, 0, 0,-1, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 3, 0, 2, 0,
                0, 0, 4, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 32         0, 0, 0, 0, 0, 0, 0,-1,
                0, 0, 0, 0, 4, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 1, 0, 3, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 33         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 3, 0, 1, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 2, 0, 0, 0, 0,
                0, 0, 0, 0, 0,-1, 0, 0,
                0, 0, 0, 4, 0, 0, 0, 0,

data 34         0, 4, 0, 0,-1, 0, 0, 1,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 2, 0, 3, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 35         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 1,-1, 0, 0,
                0, 0, 0, 4, 0, 3, 0, 0,
                0, 0, 0, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 36         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 1, 0,-1, 0, 2,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 4, 0, 3, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 37         4, 0, 0, 0, 3, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0,-1, 0, 0, 0, 0, 0,
                0, 1, 0, 2, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 38         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 4, 0, 0, 0,
                0, 0, 0, 0, 0, 3, 0, 0,
                0, 2, 0, 0, 0, 0, 0, 0,
                0, 1, 0,-1, 0, 0, 0, 0,

data 39         0, 0, 0, 0, 3, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 1, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0,-1, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 4, 0, 2, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,

data 40         0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0,-1, 0, 0,
                0, 0, 0, 0, 4, 0, 0, 0,
                0, 0, 0, 3, 0, 1, 0, 0,
                0, 0, 2, 0, 0, 0, 0, 0,
                0, 0, 0, 0, 0, 0, 0, 0,
```

第 E 章

航空機操縦問題の初期設定データ

E.1　データの定義

航空機操縦問題の初期設定データ２０ケースのパターンから構成されている。目標は、以下の値であり、値の許容誤差内に入るように制御する問題である。

- 目標高度=5000(m)
- 目標方位=90(　°).....（東）
- 速度=400(kts)

E.2　実験における初期データ

次頁に一覧にて示す。

表 E.1: 実験における初期データ

No.	pitch(°)	bank(°)	heading(°)	throttle(%)	speed(kts)	altitude(m)
1	10	0	60	80	218	10000
2	20	20	180	80	300	10000
3	10	10	210	80	150	7000
4	20	0	330	80	250	10
5	0	0	270	80	200	5000
6	-10	10	0	80	218	3000
7	0	-10	80	80	231	6000
8	0	0	200	80	218	3000
9	20	-10	150	80	218	3000
10	15	30	170	80	218	2000
11	10	0	300	80	218	10000
12	20	20	180	80	300	7000
13	10	-10	210	80	150	7000
14	20	10	330	80	250	10
15	0	40	270	80	200	5000
16	-10	-10	0	80	218	3000
17	0	-50	80	80	231	6000
18	0	50	200	80	218	3000
19	30	-10	150	80	218	3000
20	-30	30	170	80	218	10000

第 F 章

駒落ち将棋システムへの応用

F.1　はじめに

F.1.1　将棋プログラムとその研究

現在まで、チェスをはじめとして、様々なゲームが計算機上に実装されてきた。特に、欧米では古くからチェスの研究が盛んであり、その実力は、人間の世界チャンピオンのレベルに達している。　我が国には、チェスと起源が同一と考えられている将棋がある。将棋は、サイズ的にチェスより一回り大きく、取り込んだ相手駒を再利用できるという特徴を持っており、チェスに比べ各局面当たりの可能指し手の数が多い。　通常、チェス、将棋などのプログラムは、ミニマクッス法や$\alpha-\beta$法に代表される探索手法が指手決定の道具となっている。原理的には、広い手を深く読めるほど強くなるわけであるが、実際には、深く読むほど生成される局面の数が指数関数的に増大し、莫大なメモリと処理時間が必要となる。そこで、探索木の有効な枝のみを刈る操作が行われているが、重要な枝と不要な枝を見分けるのは難しい。　これまでの将棋システムの大部分は、序盤は定跡データベース、中盤以降は、一定の評価関数と探索により、指し手決定を行っているが、局面当たりの指し手の多さから、先読みについては、十分な深さまで読めないことと、重要な手を読み落としてしまうことにより、チェスのようには強くなっていない。現在のところ、将棋プログラムは研究用プログラムよりも商用の娯楽ソフトウェアの方が実力が上である。

最近日本においても欧米のコンピュータチェスのようにコンピュータ将棋に関する関心が高まり、一般の人々においても、解説書 [Kotani1990] [Matsubara94] [Matsubara96] や解説記事 [Matsubara1995a] がしばしば目に触れるようになって来た。将棋プログラムの研究には様々な視点が存在し、詰め将棋と指し将棋に関する研究に大別される。詰め将棋に関する研究は、探索手法に関する研究として、一時、最良優先探索に基づくアプローチが全盛であったが、最近は共謀数に関する多重反復深化に基づくアプローチ [Seo1995] により、より長手数の問題が解けるようになった。また、コンピュータによる詰め将棋創作の研究 [Hirose1996] なども行われている。並列化による高速解法の研究 [Ito1995] [Ito1996] の研究なども盛んになって来た。指し将棋に関する研究は、相手モデルを考慮した探索法 [Tokuda1994]、評価関数の生成や分析に関する研究 [Pell1994]、[Tokuda1995], 相手に故意に勝つチャンスを与える教授戦略 [H.Iida1994]、並列化に関する研究 [Fujii1996] などがある。市販ソフトのつぎの一手問題回答能力に関する分析 [Matsubara1995b] などがある。また、近年、指し将棋に対する最適手の発見において、人間の持つ戦略的知識の重要性が指摘されている。[E.Iida1996][E.Iida1997]

F.1.2 駒落ち将棋

駒落ち将棋は、古くからプレーヤー同士の戦力（戦略知識、使える駒）の点で、バランスをとるために、実力の上位者（より戦略知識の豊富なプレーヤー）側の一部の駒をゲーム開始に先だって盤上から除いておくものである。実力が同じならば、当然、ハンディのある方が不利である。

コンピュータサイエンスの面から考察する。もし、同一の静的評価関数でこの局面を評価したならば、ゲーム開始の時点において、差が生じている。ゲーム開始後、駒の取り合いが発生し、上位者に徐々に駒が渡ると評価関数の差が縮まる。実力下位の者にとっては、最終局面までこの差を維持できるかどうかで勝敗が決まる。

実力の差に応じて、下表の様なハンディキャップがある。

表 F.1: 駒落ちのパターン

落す駒→ ↓駒落ちパターン	飛車	角	金	銀	桂	香	備考
10 枚	1	1	2	2	2	2	差大
8 枚	1	1		2	2	2	↑
6 枚	1	1			2	2	↑
5 枚	1	1			1	2	↑
4 枚	1	1				2	↑
3 枚	1	1				1	↑
2 枚	1	1					標準的駒落ちパターン
飛香	1					1	↓
飛	1						↓
角		1					↓
香						1	差小

F.2 問題の定義

ここでは、最も、実力差の大きい場合の１０枚落ち将棋の必勝将棋プレー・システムを構築する。即ち上手は、玉と歩のみである。

初期配置は図 F.1のとおりであり、上側が実力上位者である。一般には、上手、下手、双方の目的は、自分の王より、先に、相手の玉を詰めることである。ルールは、上手先攻で。また、予め除外した駒は再利用できない。その他は、平手（ハンディキャップなし）将棋と同じである。なお、ここでは、平手将棋のルールには立ち入らない。

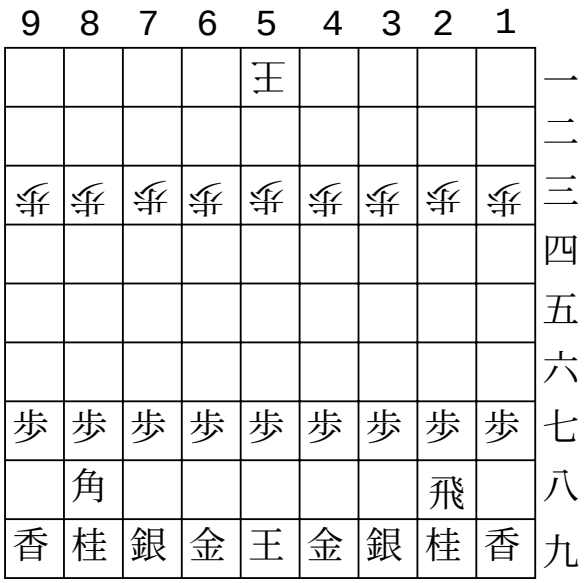


図 F.1: １０枚落ち将棋の初期配置

たとえ１０枚落ちでも、初心者には実力上位の者になかなか勝つことが難しい。また、従来の枠組で必勝プログラムを組むこともは、記述量が増える、デバッグが難しい、必勝であることの保証など、の理由から難しい側面があった。

F.3　フェーズフロー構築のための知識表現

状態の知識表現

本問題空間の状態 s は、以下のように表現するものとする。

$$s = (x_1, x_2, ..., x_{81}) \tag{F.1}$$

ここで、各状態変数は、図 F.2 の位置に対応しており、変数には、その位置に存在する駒の識別番号が与えられる。また、計算上、座標（X、Y）を使用する。ただし、この場合、X=1〜9、Y=一〜九。

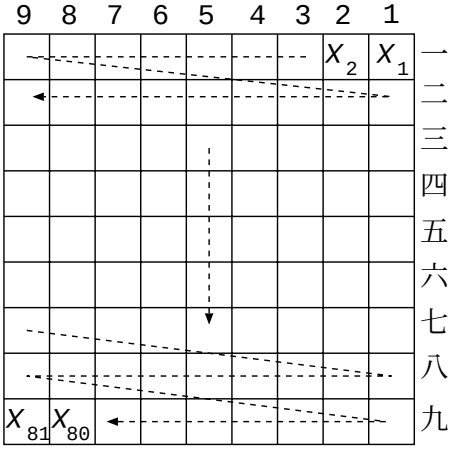


図 F.2: １０枚落ち将棋の知識表現

問題分割に関する検討

ここでの方針は、初期状態における優位を最後まで保つことを目指す。即ち、最適な問題解決アプローチではない。ゆっくりとした解決速度ではあるが、確実に解決できるアプローチをとる。その理由は、フェーズフローを用いた問題解決のアプローチ自体が “危険を冒しつつも迅速に” というよりは “ゆっくりではあるが確実に” ということにウエートが置かれているためである。以下、１つの解決可能なプロセスを述べ、フェーズフローにて記述する。

F.3.1　メインフェーズフロー

- **Phase1**(P_1)　**〜　1 筋の歩をつき捨てて飛車をなり込む**
  この目的は、図 **F.3A** の様に１六龍とするためである。単純にこのプロセスだけを実行してみると、自分の駒だけで最悪、8 回動かすことが必要となる。この間、他の筋において相手の歩の突進を受ける場合がある。この場合のフェーズフローＦ１はサブフェーズフローのとこで述べる。

- **Phase2**(P_2)**:１二歩と打って “と金[1]” を生成**
  図 **F.3B** のように、１二歩と打った時点で相手は、“ と金 ”生成の阻止が出来ない。

- **Phase3**(P_3) **〜更に、と金を作る。**
  “と金” と “龍” を使って、相手の歩を取る。 このフェーズフローＦ２は、あとで

[1]歩の成りで、金と同じ能力をもつ

述べる。

- **Phase4(P_4)　～　更に、1 筋に “と金” ４枚縦に連ねる。**
  図 **F.3C,D** のように、更に、“ と金 ”を４枚縦に連ねる。この “と金” を４枚連ねるプロセスは、**Phase2,Phase3** を用いることとする。通常、この時点で、下手の勝ちとなる。なぜならば、“ と金列 ”と“ 龍 ”を順次横にシフトしながら上手の玉の存在 領域を狭めていくからである。このプロセスは、Ｆ３として、後で述べる。

$$F_0 : P_0 \Rightarrow P_1 \Rightarrow, ..., \Rightarrow P_4 \tag{F.2}$$

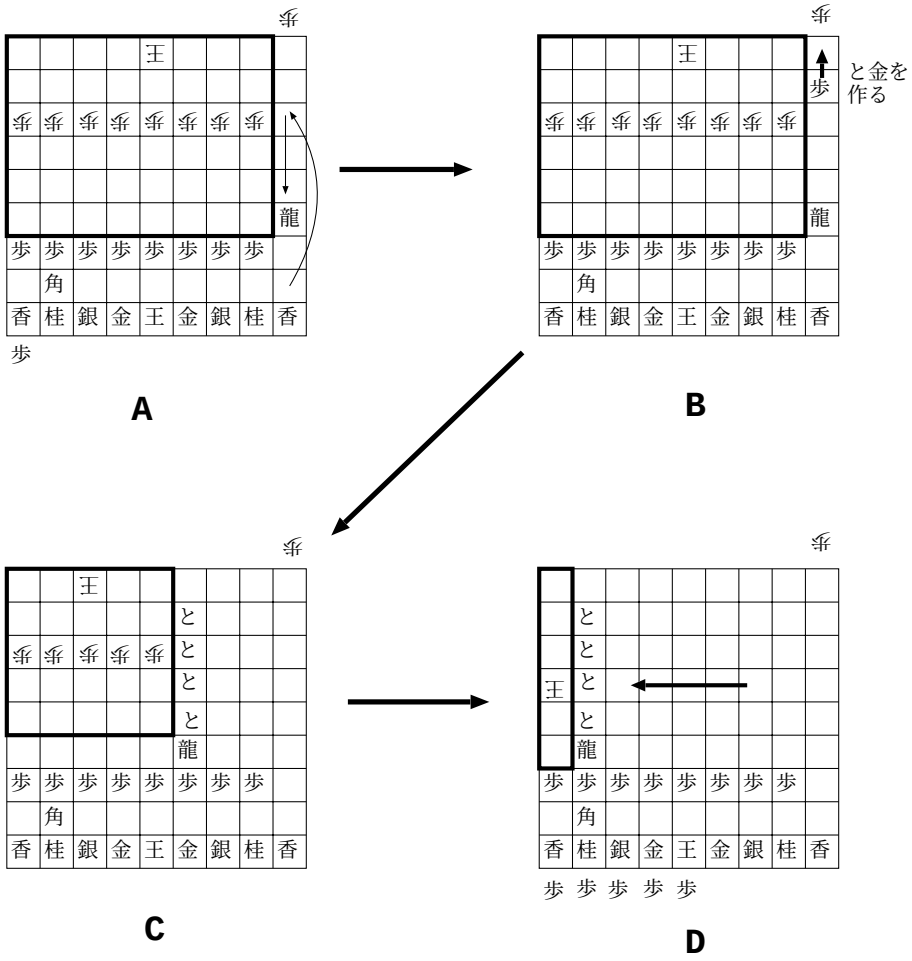


図 F.3: 解決プロセス概要

F.3.2　サブのフェーズフロー

- **サブフェーズフロー** $F1$

本フェーズフローは、龍がＰ１の定位置に着く前に、相手の歩の突進を受けた場合の対処方法である。龍が定位置のくるまで最悪８手かかるのに対し、相手の歩は、３手で六段目に到達する。ここでは、駒がぶつかってすぐに取るのではなく、到達してから始めて、下段から受け駒上がって対処するものとする。この理由は、１六龍となったとき、自陣に死角を作らないためである、従って、下段の駒のバックアップ体制に関して規定する。以下のように、細部フェーズフローを作るものとする。これらは、あくまで状態集合で扱う。たとえば、△８六歩はこの位置に相手の歩が存在する、あらゆる、局面を含む。

f_1:△９七と⇒ ▲９七角
f_2:△８六歩⇒ ▲７八銀⇒ △８七と⇒ ▲同銀
f_3:△７六歩⇒ ▲７八銀⇒ △８七と⇒ ▲同角
f_4:△６六歩⇒ ▲６八金⇒ △８七と⇒ ▲同金
f_5:△５六歩⇒ ▲５八玉⇒ △８七と⇒ ▲同玉
f_6:△４六歩⇒ ▲５八玉⇒ △８七と⇒ ▲同玉
f_7:△３六歩⇒ ▲２八銀⇒ △８七と⇒ ▲同銀
f_8:△２六歩⇒ ▲３八金⇒ △８七と⇒ ▲同金

- **サブフェーズフロー** $F2$

“と金” と龍で相手歩を捕獲するプロセスを示す。龍の先の“ と金 ”を左となり列の相手歩を捕獲する。相手歩座標 (X_F, Y_F)、“と金” の座標 (X_T, Y_T)、龍の座標 (X_R, Y_R) とする。

Phase0 : $Y_T \leq Y_F$ ならば “と金” さがる

Phase1 : $(Y_T = (Y_F + 1)) \wedge (Y_R = 6) \wedge (Y_F \leq 4)$ ならば 龍は捕獲対象の歩の存在する筋に移動する。

Phase2 : 自分の手番にて “と金” で歩を捕獲する。

$$F_2 : P_0 \Rightarrow P_1 \Rightarrow P_2 \tag{F.3}$$

- **サブフェーズフロー** $F3$

図 F.3C のように、 “と金” が４枚一列にならび龍がその下にいるものとする。これを詰むまで繰り返す。

Phase0 : すべて１列

Phase1 : 2 段目の “と金” を左１マスシフト。

Phase2 : 3 段目の “と金” を左１マスシフト。

Phase3 : 4 段目の “と金” を左１マスシフト。

Phase4 : 5 段目の “と金” を左１マスシフト。
Phase5 : 龍を左１マスシフト。

$$F_3 : P_0 \Rightarrow P_1 \Rightarrow P_2, ..., \Rightarrow P_5 \tag{F.4}$$

F.3.3　フェーズフローチャート

全体の構造は以下のとおりである。

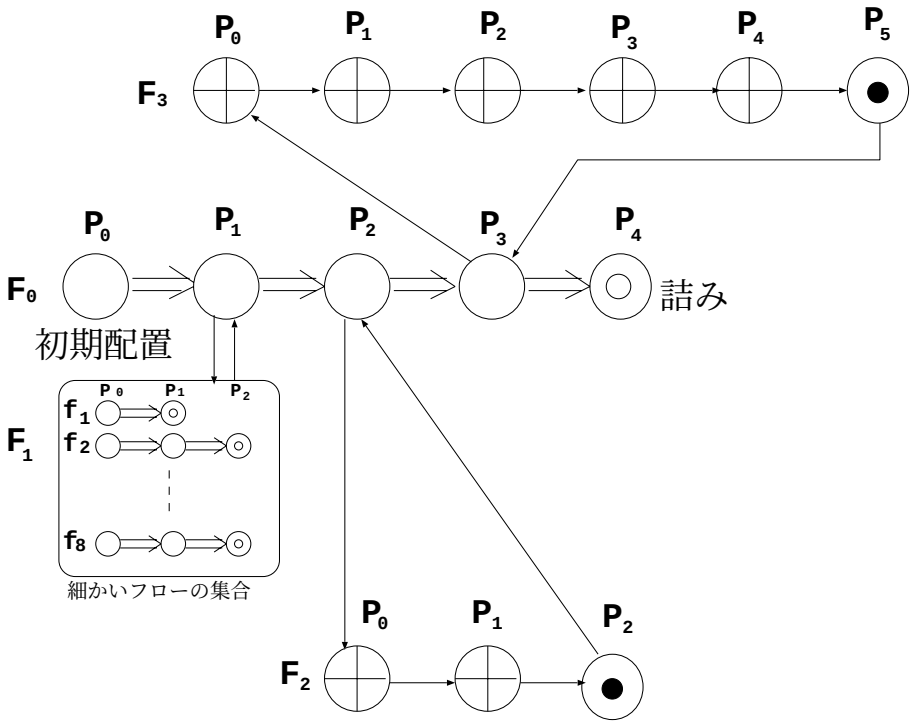


図 F.4: 解決プロセス概要