学位論文

マルチエージェント協調作業のための経路プランニングに関する研究
(Path Planning Algorithm for Multi-agent Cooperation Tasks)

平成 18 年 12 月 15 日　博士 (科学) 申請

指導教員　　伊庭　斉志　教授

東京大学大学院新領域創成科学研究科 基盤情報学専攻

神尾　正太郎

Abstract

The cooperation of multiple robots enables complex tasks. If the task is complicated, an operator will be forced to send many instructions to the robots in order for the task to be achieved. It is necessary to be able to control the robots easily. Path planning algorithms will reduce the burden for this purpose.

This paper proposes a path planning framework for the sake of the multi-agent cooperation tasks. In the multi-agent cooperation tasks, some robots are assigned a sub-task and its sub-goal. But the other robots cannot be assigned the sub-task nor the sub-goal. Therefore, the path planning algorithms with a sub-goal and without a sub-goal are proposed in this paper. In addition, sub-goal generation algorithm is presented. These algorithms are based on the Rapidly-exploring Random Trees (RRT) method. The algorithms are realized with a centralized planner. Its applicability is shown in two tasks: a multi-agent object rearrangement task and a cooperative object transportation task. Moreover, the cooperative object transportation task is realized using two humanoid robots in a real world environment.

The multi-agent object rearrangement problem is the task that multiple robots carry objects from their initial positions to their goal positions. A robot transfers one object at a time. The path planning for multiple robots requires large amount of computation. The proposed algorithms reduce the complexity. Therefore, the computation time will gradually raise even if the number of robots increases. The result is compared with the dynamic programming (DP) approach. The proposed method can acquire comparable result in shorter execution time than that of the DP approach.

The cooperative object transportation task is the task of having robots cooperate in carrying an object from a start position to the specified goal position. In this task, each robot is in separated rooms with windows. Two robots have to cooperate and transfer the object from one to the other through the windows. The proposed algorithm can produce robots' paths by using small amount of information. The effectiveness of the algorithm is shown by simulation results. The result is compared with the normal RRT method. The proposed algorithm is more faster than the normal method in complex problems.

In addition, the passing order searching is investigated in the cooperative

object transportation task. The objective is to determine the order in which the object passing should be executed. This is a difficult task to find an optimum solution. Two algorithms are discussed that use the best-first search with a cost to go to the goal position. The algorithms reduce the amount of the required information that the operator has to give to the robots and it makes robot operations easy. The effectiveness of the algorithms is verified through simulation.

The application to a real world problem is also presented. When realizing our path planning using actual robots, noise coming from the environment needs to be dealt with. The position of a robot is affected by the noise arising from slipping and deviation of robot movements. As a result, the robots cannot achieve the task by simply executing the planned path. Therefore, the re-planning algorithm is necessary for accomplishing the task. This paper describes the algorithm and the control system. When the robot's location significantly differs from the intended path, the planning is redone based on the current location. The robot acquires their locations using the cameras with Monte Carlo Localization. The experiment is performed with the cooperative object transportation task using two humanoid robots. The result confirms the applicability of the proposed method.

内容便概

近年，ロボットが多数開発され，産業分野や日常生活の場などに普及すると期待されている．その際，ロボットを複数台で協調動作させて，複雑なタスクを効率的に行うことが期待される．複数のロボットをタスクに応じて適切に協調制御することは難しい問題である．この問題を解決するには，複数ロボットのための経路プランニングが必要である．しかし複数ロボットを扱うには計算量が多く，これまで有効なアルゴリズムがほとんどなかった．

本論文では，ランダム探索に基づいた，マルチエージェント環境のための経路プランニングアルゴリズムを提案する．このアルゴリズムはマルチエージェント環境でも効率的にプランを探索することができる．そして，提案アルゴリズムは様々なタスクに応用可能であることを２つの実例を通じて示す．

その１つ目の例は物体再配置問題である．この問題はこれまでマルチエージェント環境で性能の良いアルゴリズムはほとんどなかった．提案手法を用いることで，マルチエージェント環境での物体再配置問題を効率的に解けることを示す．

２つ目の例は協調荷物搬送問題である．これはロボットが荷物を受け渡しながら作業を行うため，荷物を受け渡す場所（サブゴール）が必要となるが，提案アルゴリズムを応用することで，サブゴールを自動的に探索することができる．その上，環境モデルだけを使って，どのような順序で受け渡しをすればよいかを探索できる．

さらに，提案手法の実環境への応用を想定して，ヒューマノイドロボットでの協調荷物搬送作業を実現する．単純にプランニング結果をロボットに適用するだけではシミュレーションと実環境のずれが大きいために失敗することが多い．そのため再プランニング手法を提案する．これを用いることで，実環境で生じるずれに応じてプランを修正できることを示す．

目次

図目次

表目次

第1章

序論

1.1 本研究の背景

近年実用的なロボットの研究開発 [55, 62, 59] が盛んとなり，産業分野やその他の分野における実用システムの開発推進事業も行われている [53]．将来的には，環境中にロボットが複数配置される状況も期待できる．複数のロボットが協調作業を行うことで，より複雑な作業の実現が期待される．例えば，ロボットよりも大きい物体の操りなどでは複数の移動ロボットが必要である [49]．また，Shinozawa らが提案するように，人間サイズのロボットと小型のロボットで協調したり作業領域ごとに役割分担する [35] という異種ロボット同士での協調作業も考えられる．

ヒューマノイドロボットのように，ロボットは人間の生活する環境での活躍も期待されている．しかし人間の生活空間で活動するには，活動環境のロバストな認識や人間などにとって安全な行動アルゴリズムなどが実現できなければならず，まだ課題も多い．このようなことから，ヒューマノイドロボットの活躍の場としてはまずは工場など産業分野での利用が期待される．産業分野であれば，工場など，ロボットが認識しやすい環境を作り上げやすい．また，作業用途に必要な知識だけを理解できれば良く，ロボットにとって対応困難な状況も少なくできる．

ロボットにとって分かりやすい環境を用意したとしても，タスクのために複数のロボットを制御することは容易ではない．それには複数のロボットがどのように行動すれば良いかをあらかじめ計画しなければならないからである．この計画を行うのがプランニングである．本研究では特にロボットをどのような経路で移動させるかという経路プランニングに着目する．

複数ロボットが存在する状況での経路プランニングは困難な問題である．複数のロボットが同時に行動するため，環境中の障害物だけでなく，他のロボットも移動する障害物となってしまうからである．このようにマルチエージェント環境はシングルエージェント環境に比べて動的で複雑である．

シングルエージェント環境で用いられる経路プランニングアルゴリズムは静的な環境を想定することが多く，そのままマルチエージェント環境に適用した場合には計算量が膨大

となり，実用的でない．そこで，マルチエージェント環境に適した経路プランニングアルゴリズムを実現することが重要である．

1.2　本研究の目的

本論文では，マルチエージェント環境に適した経路プランニングアルゴリズムを提案する．提案アルゴリズムはマルチエージェント環境でも効率的に探索できる．また様々なタスクに応用可能であることを実例を通じて示す．さらに，シミュレーション実験で確認するだけでなく，実環境での適用可能性を確認する．

経路プランニングではタスクに必要な情報以外に，ロボットのゴール状態を必要とする手法も多い．しかしあらかじめゴール状態を与えることは，それだけで十分に複雑な問題である．なぜならば初期状態からゴール状態までに障害物などとの衝突がなく，タスクを達成できるようなゴール状態を考えて指定しなければならないからである．そこで操作者の負担を減らす観点から，必要最低限以上の情報を必要としないプランニング手法が有用である．提案手法はその負担を減らすアルゴリズムも提供するため，あらかじめ必要な情報が少なくてすむ．

本論文の構成は次のようになっている．以下ではまず従来の経路プランニングアルゴリズムについてまとめる．第2章では，マルチエージェント環境のための提案経路プランニングアルゴリズムについて詳述する．第3章では，提案アルゴリズムを物体再配置問題に適用する．第4章では提案アルゴリズムを協調荷物搬送問題に応用し，第5章では協調搬送問題での協調手順を自動的に探索するアルゴリズムを示す．さらに第6章では協調荷物搬送アルゴリズムをヒューマノイドロボットで検証し，再プランニングアルゴリズムを提案する．最後に第7章で結論を述べる．

1.3　マルチエージェント制御

複数ロボットの制御システムを実現するには中央集権的システムによる実現方法と分散システムによるものとがある．本研究では中央集権システムを用いて実現する．

複数エージェントの協調作業を対象とする場合には必然的に，ある程度の情報を共有しなければならない．エージェント間で共有する情報量の多少に応じて，制御システムのアルゴリズムは完全に中央集権的なものから完全に分散的なものまでのバリエーションがありうる [14]．しかし分散システムによる実現方法は対象とする問題に依存しがちである．そのため汎用的な方法論へと拡張することが困難である [41]．それに対し，中央集権的システムでは汎用的なシステムを比較的構築しやすい．中央集権的な制御システムを分散システムへと変換する手法の研究も行われており [41]，中央集権的なシステムを研究することには意義がある．

ロボットの中央集権制御を行うには，各ロボットの状態に関する情報を収集する必要がある．この状態とはロボットの内部状態や環境中での位置，方向などである．内部状態は

ロボット自身で把握することができるが，環境中の位置，方向を知るには外部のセンサが必要である．近年では GPS やレーザーレンジファインダ，カメラなどからの情報を用いて環境中のロボット位置を推定する Localization 手法 [38, 61] の研究や，さらに環境の地図も同時に獲得する SLAM（simultaneous localization and mapping）[1] などの研究が盛んとなっている．これら手法により環境中でのロボット位置を実用的な精度で得られる．これらのことから，中央集権的な制御によるシステムは実現可能であり，汎用的なシステムを研究することが重要であると考えられる．これが可能であることは第 6 章でヒューマノイドロボットを用いた制御システムの実現を通じて示す．

本論文では中央集権的システムによるアルゴリズムで，複数ロボットが同じゴールへ向けて協調するために必要な経路プランニングアルゴリズム群を示す．これらで実現される機能は以下のようなものである．

1. マルチエージェント環境でも衝突のない，目標位置までの経路を速く生成する．
2. 目標位置を定めなくとも，他のロボットと衝突しない経路を生成する．
3. 2 台のロボットが協力する場所を決定する．

これらアルゴリズムの詳細は第 2 章で説明する．

1.4　経路プランニング

プランニングに関してはこれまで人工知能分野でさまざまな研究が行われてきた．プランニングの扱う問題範囲は広いが，ここでは本研究に関係する，経路プランニングまたは行動プランニング（Motion planning）に関する研究を扱うにとどめる [*1]．経路プランニングをまとめたものとしては，文献 [23] が詳しい．

経路プランニングはこれまでに多数の研究が行われている分野である．古くは，ピアノを障害物にぶつからないように目的位置まで運ぶ経路を見つける Piano Mover's Problem [26] などが研究された．経路プランニングにはさまざまなアルゴリズムを適用することができる．例えば，離散空間で A*などのヒューリスティック関数に基づいて探索を行うものや，環境の幾何学形状を分解して経路をプランニングする cell decomposition という呼ばれる手法など多く研究されている [23]．いずれの手法も計算量や得られる経路の形状，精度などにそれぞれ長所，短所を持っている．

1.4.1　プランニングの困難さ

一般化された行動プランニング問題では，PSPACE 困難と呼ばれる理論的困難さが証明されている [23]．PSPACE 困難とは，アルゴリズムにおける NP 困難と似た概念で，

[*1] 経路プランニングはロボットの行動する経路をプランニングすることであり，行動プランニングはその行動系列をプランニングすることとして区別できる．しかしアルゴリズムを得る上では同じであり，ここでは経路プランニングと行動プランニングを同義として扱う．

アルゴリズムが使用するメモリ量が多項式では表現できないことを表す [*2]．そのため，どのような問題に対しても最適解を求められるアルゴリズムは実現不可能である．

そこで，現在研究されている行動プランニングのアルゴリズムは，大きく分けて 2 つのアプローチがある [23]:

- 限定した問題のみ扱う
- アルゴリズムに対する要求を減らす．（例えば，アルゴリズムの完全性をあきらめ確率的完全性を持つアルゴリズムで実現する，など．）

前者は制限されたタスクを対象とすることで，完全なアルゴリズムを実現させる．例えば，静的な環境だけを扱ったり，扱う物体の数を制限するなどが挙げられる．例えば，Wilfong [39] は，1 つの可動物体を目的位置まで移動する問題に対する完全なアルゴリズムとその計算量を明らかにした．しかし可動物体が複数の場合は複雑であるとして扱われていない．

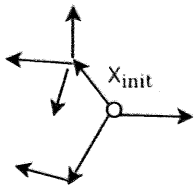
後者の例としては，ヒューリスティクス関数を用いるものや確率的な探索を行う手法がある．近年，経路プランニングで用いられるアルゴリズムとして Rapidly-exploring Random Trees（RRT）と呼ばれる手法がある [24]．これは確率的な探索を行うアルゴリズムである．このアルゴリズムは確率的完全な（probabilistic complete）アルゴリズムと呼ばれる．その意味は，十分な探索時間を与えれば，解を得られる確率が 1 に近づく（どのような問題に対しても解を得られる）ということである．また，前処理を必要とせず，高速なプランニングが可能である．

1.5　Rapidly-exploring Random Trees (RRT)

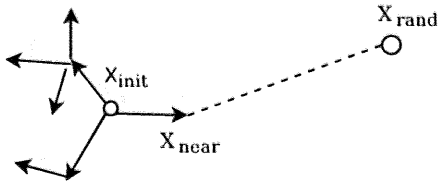
RRT は状態空間内でランダムサンプリング（乱数）を用いて探索木を成長させる手法である [24, 21]．この探索木によってプランニングを行なう．ここで状態空間とは，ロボットの各関節の角度指令値や環境内の位置などの状態変数によって表現される空間で，その状態空間中の一点（状態）を指定すると，環境内でのロボットの位置，姿勢が一意に定まるものをいう．

ロボットの状態空間を $\boldsymbol{C}$ とし，そのうち，環境中の物体と衝突しない状態のなす空間を $\boldsymbol{C}_{\mathrm{free}}$ とする．すると経路プランニング問題とは，初期状態 $\boldsymbol{x}_{\mathrm{init}}(\in C_{\mathrm{free}})$ から最終状態 $\boldsymbol{x}_{\mathrm{goal}}(\in C_{\mathrm{free}})$ までの $\boldsymbol{C}_{\mathrm{free}}$ 内の経路を探索する問題と言い替えることができる．$\boldsymbol{C}_{\mathrm{free}}$ は環境中の物体やロボットの形状によって決まるが，その相互作用が複雑なために，明示的に書くことは一般に難しい．しかし状態が 1 つ与えられたときにそれが $\boldsymbol{C}_{\mathrm{free}}$ 上にあるか判定することは比較的易しく，その状態でのロボットと環境との衝突判定をすれば良い．RRT はランダムサンプリングされた状態が C_{free} 上かどうかの判定が主な処理であ

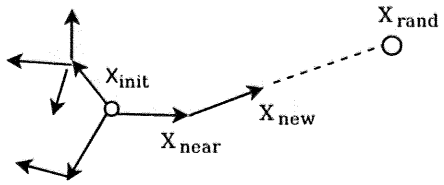
[*2] プランニング問題の一種と考えられる倉庫番として知られるパズルは PSPACE 困難と証明されている [10].



(a) 探索木 T



(b) ランダムサンプリングで得られた $\boldsymbol{x}_{\mathrm{rand}}$



(c) 探索木から枝を伸ばす

図 1.1 RRT 探索の手順

るため，高速な経路プランニングを実現できる．

RRT は多次元空間においても高速なプランニングが可能である．これはヒューマノイドロボットのような自由度の大きいロボットにも適用可能であるため，近年盛んに研究されている（これら研究については第 1.6.1 節で述べる）．

1.5.1 アルゴリズム

RRT のアルゴリズムは以下の手続きにより，初期状態 $\boldsymbol{x}_{\mathrm{init}}$ を根とする探索木 T を構成する [24]．

1. 初期状態 $\boldsymbol{x}_{\mathrm{init}}$ のみを含む探索木 T を生成する（図 1.1(a)）．
2. 探索すべき状態空間からランダムサンプリングにより（ランダムに）状態を選び $\boldsymbol{x}_{\mathrm{rand}}$ とする（図 1.1(b)）．
3. 探索木 T 中で $\boldsymbol{x}_{\mathrm{rand}}$ に最も近い状態 $\boldsymbol{x}_{\mathrm{near}}$ から，$\boldsymbol{x}_{\mathrm{rand}}$ に向けて一定距離進めた状態を $\boldsymbol{x}_{\mathrm{new}}$ とする（図 1.1(c)）．
4. 状態 $\boldsymbol{x}_{\mathrm{new}}$ が環境や障害物との衝突を起こさなければ，探索木 T に $\boldsymbol{x}_{\mathrm{near}}$ から $\boldsymbol{x}_{\mathrm{new}}$

への枝を追加する．（ステップ3.，ステップ4. の操作をまとめてEXTEND操作と呼ぶ.）

5. $\boldsymbol{x}_{\mathrm{new}}$ が最終状態 $\boldsymbol{x}_{\mathrm{goal}}$ に到達していれば，$\boldsymbol{x}_{\mathrm{init}}$ から $\boldsymbol{x}_{\mathrm{goal}}$ までの状態列を出力として返し終了．そうでなければ，ステップ2. へ戻って繰り返す．

探索には実数値空間を用いるため，$\boldsymbol{x}_{\mathrm{new}}$ が最終状態 $\boldsymbol{x}_{\mathrm{goal}}$ に到達したかどうかは $\boldsymbol{x}_{\mathrm{goal}}$ から一定距離内まで近づいたかどうかで判定する．探索木が $\boldsymbol{x}_{\mathrm{goal}}$ に到達すれば，$\boldsymbol{x}_{\mathrm{init}}$ から $\boldsymbol{x}_{\mathrm{goal}}$ までの状態の系列，すなわち経路プランニングの結果が得られる．

このアルゴリズムは，解が存在しても必ず解を見つけるとは限らない．そして解が存在しない場合には，アルゴリズムが停止しない．そのためループを一定回数繰り返しても探索木が $\boldsymbol{x}_{\mathrm{goal}}$ に達しない場合にはプランニング失敗とする．RRTは扱う問題に対する仮定がほとんどないため，さまざまな問題への適用が容易である．必要となるのは状態間の距離尺度だけである．

1.5.2 拡張手法

Kuffnerらは RRT-Connect を提案している [21]．これは初期状態と最終状態の双方向から探索木を成長させる手法である．双方向から探索することで RRT の探索効率が向上する．さらに，上述の EXTEND 操作を拡張した CONNECT 操作を提案している．CONNECT 操作は，探索木が成長する限り EXTEND 操作を繰り返す操作である．これにより，探索空間で環境との衝突が発生しない限り，サンプル点まで探索木を一度に成長させることができる．これによって探索効率が向上する．

双方向探索には CONNECT 操作と EXTEND 操作を組み合わせることも可能である．しかし LaValle らによれば，ホロノミック環境で最も良い性能を示すのは，CONNECT 操作で双方向に探索を行う RRT-ConCon である [24]．

RRT-ConCon のアルゴリズムは以下のとおりである：

1. 初期状態 $\boldsymbol{x}_{\mathrm{init}}$ と最終状態 $\boldsymbol{x}_{\mathrm{goal}}$ を根とする2つの探索木をそれぞれ T_a, T_b とする．
2. ランダムサンプリングにより状態 $\boldsymbol{x}_{\mathrm{rand}}$ を選択する．
3. 探索木 T_a から $\boldsymbol{x}_{\mathrm{rand}}$ に向けて CONNECT 操作を実行する．T_a から $\boldsymbol{x}_{\mathrm{rand}}$ に向けて枝を伸ばすことができたならば，その枝の末端状態を $\boldsymbol{x}_{\mathrm{new}}$ とする．枝を伸ばすことができなければ，ステップ6. に進む．
4. 探索木 T_b から $\boldsymbol{x}_{\mathrm{new}}$ に向けて CONNECT 操作を実行する．T_b から $\boldsymbol{x}_{\mathrm{new}}$ に向けて枝を伸ばすことができたならば，その枝の末端状態を $\boldsymbol{x}'_{\mathrm{new}}$ とする．枝を伸ばすことができなければ，ステップ6. に進む．
5. $\boldsymbol{x}_{\mathrm{new}}$ と $\boldsymbol{x}'_{\mathrm{new}}$ が等しければ探索成功である．初期状態 $\boldsymbol{x}_{\mathrm{init}}$ から $\boldsymbol{x}_{\mathrm{new}}$ を経由して最終状態 $\boldsymbol{x}_{\mathrm{goal}}$ までの状態列を出力として返し終了．
6. 探索木 T_a と T_b を入れ替えて，ステップ2. から繰り返す．

この手続きは Kuffner の提案する RRT-Connect と等しい.

1.5.3 従来の RRT を適用する場合の問題点

従来の RRT 手法でも，複数のロボットを一度に表現する状態空間を設定することで，マルチエージェント環境での経路プランニングに適用できる．しかしこの場合には，以下のような問題点がある．

1. 各ロボットのゴール状態を明示的に与える必要がある．
2. タスクに関係のない無駄な動きが生じる．

1 つ目の問題点は，双方向探索（1.5.2 節）を行なうための必要条件である．各ロボットにゴール状態を明示的に与えることは，それだけで問題をある程度解くことになる．つまり経路プランニングアルゴリズムの利用者側からすれば，ゴール状態を与えるのは煩雑である．逆に双方向探索を行なわず，ゴール状態となる条件だけ与えて，その状態に達するまで探索することも可能である．しかし探索時間は非常に長くなるため，現実的でない．

2 つ目の問題点は，各ロボットにゴール状態を明示的に与えたとしても生じる．単純な RRT では複数ロボットのプランニングを同時に実行する．そのため，あるロボットがゴール状態に到達しても，もう一体がゴール状態に達するまでに時間があるため，その状態から離れたり近付いたりという無意味な動作を示す．これはそれぞれのロボットを個別に経路プランニングするのでない限り避けることのできない問題である．

1.5.4 時刻を含めた双方向プランニングの問題点

無駄な動きを減らすには各ロボットで個別に経路プランニングする必要がある．それには各ロボットの状態に時刻パラメータを含めて RRT で探索を行なえば良い．すなわち，時刻 t とその時刻のロボットの状態 $\boldsymbol{x}$ からなる組 $(t, \boldsymbol{x})$ を状態空間 $\boldsymbol{C}$ として改めて定義して探索を行う．これを RRT で実現するアルゴリズムは，以下のようになる：

1. 初期状態 $(t_{\text{init}}, \boldsymbol{x}_{\text{init}})$ を根とする探索木を T とする．
2. ロボットの状態 $\boldsymbol{x}$ に関してランダムサンプリングを行い，サンプリング点 $\boldsymbol{x}_{\text{rand}}$ を得る．
3. 探索木 T 中で $\boldsymbol{x}_{\text{rand}}$ にロボットの状態が最も近い状態 $(t_{\text{near}}, \boldsymbol{x}_{\text{near}})$ から $\boldsymbol{x}_{\text{rand}}$ に向かって枝を伸ばす．このとき移動距離に応じて時刻を進めることで，新たな状態 $(t_{\text{new}}, \boldsymbol{x}_{\text{new}})$ を求める．
4. $(t_{\text{new}}, \boldsymbol{x}_{\text{new}})$ が衝突を起こさない状態であれば探索木 T に追加する．
5. $(t_{\text{new}}, \boldsymbol{x}_{\text{new}})$ が目的とする最終状態であれば，出力として初期状態から最終状態までの状態列を返して終了．そうでなければステップ 2. に戻って繰り返す．

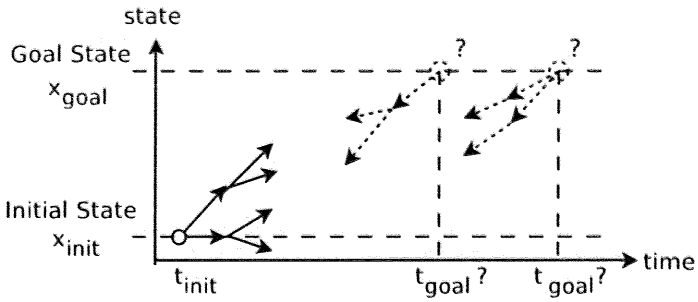


図 1.2　時刻を含めて探索を行なう際には，ゴール状態 x_{goal} の時刻 t_{goal} が決まらない限り双方向探索ができない（横軸に時刻 t を，縦軸に状態変数 x をとる，状態空間 (t, x) での例）.

この手続きにより，RRT で時刻を含めた状態空間の探索ができる．探索木中で $\boldsymbol{x}_{\mathrm{rand}}$ にロボットの状態が最も近い状態 $(t_{\mathrm{near}}, \boldsymbol{x}_{\mathrm{near}})$ を探す場合に，時刻だけが異なる複数の状態があり得る．その場合，単純な方法としては，もっとも時刻の早い状態を選択する．

しかしこのように時刻を含めた探索を行なう場合に，双方向探索（1.5.2 節）を行うのは困難である．双方向探索では探索木を成長させるためにあらかじめ初期状態 $(t_{\mathrm{init}}, \boldsymbol{x}_{\mathrm{init}})$ と最終状態 $(t_{\mathrm{goal}}, \boldsymbol{x}_{\mathrm{goal}})$ を定めなければならない．ところがロボットが最終状態 $\boldsymbol{x}_{\mathrm{goal}}$ に達する時刻 t_{goal} を前もって決定するのは一般に困難である．そのため時刻を含めた状態空間での双方向探索は困難である（図 1.2 参照）．双方向探索を使わない場合には探索効率が非常に悪くなり，実用上問題がある．

1.6　関連研究

1.6.1　RRT を用いた経路プランニング

RRT はロボットの行動生成 [22, 19] や経路計画 [7, 28] に応用されている．Kuffner らは，RRT を用いてヒューマノイドロボットの行動生成（物をつかむ，片足を上げるなど）を行なった [22]．彼らの手法では初期姿勢，最終姿勢，環境のモデル，ロボットのモデルが与えられれば，初期姿勢から最終姿勢までの（環境物体と衝突しないような）行動系列をプランニングすることが可能である．Okada らは，ステレオビジョンを用いたヒューマノイドロボットの経路計画に RRT を用いた [28]．

これらの従来手法でロボットに一連のタスクを行なわせる場合には，タスクを達成できるような目標姿勢や位置を与えなければならない．複数ロボットに協調作業の指令を出すには，指令として与えるべき情報も多くなり，操作が煩雑である．

1.6.2 マルチエージェント環境における経路プランニング

マルチエージェント環境での経路プランニングを扱った研究がこれまでに多数行われている．大きく分けて2通りの方法に分けられる．

- ヒューリスティクス関数を用いるもの [50].
- 限定されたタスクに適用できるもの [33, 9, 11, 5].

ヒューリスティクス関数を用いるものは，どのように関数を設計するかが問題となる．また，対象とするタスクごとに関数を再設計しなければならないなど，利用者の負担も大きい．

限定されたタスクに適用できるアルゴリズムは，タスクに制限を加えることでプランニング計算量を抑えている．そのため異なるタスクに応用できるものは少ない．Parsons らは複数のロボットの経路プランニングを行うアルゴリズムを示した [33]．彼らのアルゴリズムは完全なアルゴリズムであり，ロボット数が2の場合について実装を示している．しかし完全なアルゴリズムであるがゆえに，ロボット数の指数関数で計算量が増大するため，現実的な手法ではない．

Curtiss [11] は，優先度をつけた RRT を用いて複数エージェントのプランニングを実現した．この手法では従来の RRT を用いて初期状態から探索木を伸ばす．優先度の高いロボットからプランニングし，衝突が起こらない経路が得られるまでランダムに点を選ぶ．彼の研究では非常に大きな2次元空間でのプランニング問題であるため衝突もあまり起こらない．十分な移動スペースが存在するならばゴールまで到達できるが，衝突が多くなるとゴールに到達するまで時間がかかるという問題がある．

Bruce らは複数ロボットの行動プランニングを RRT で実現した [5]．しかし各々のロボットが個別に，決められた目標位置まで障害物や他のロボットに衝突しない経路をプランニングするだけであり，協調しているとは言えない．

このようにマルチエージェント環境では計算量が問題となる手法が多い．また，特定のタスクにしか適用できない手法が多く，協調タスクを扱っているものは少ない．さらに，協調作業のためのゴール位置，サブゴール位置などの自動生成に焦点を当てた研究はほとんど行われていない．

次章ではこれら課題を解決した経路プランニングアルゴリズムを提案する．

第 2 章

マルチエージェント環境のための RRT 経路プランニングアルゴリズムの提案

本章ではマルチエージェント環境での経路プランニングに必要とされる機能を議論し，そのためのアルゴリズムを示す．これらアルゴリズムには RRT を応用するため，マルチエージェント環境でも計算量が指数関数的に増加することがない．さらに，さまざまなマルチエージェント問題に適用可能である．

2.1 マルチエージェント環境のための RRT 経路プランニングアルゴリズム

2.1.1 前提条件

提案アルゴリズムが仮定する前提条件を説明する．

- 環境中の障害物は固定されており，その形状モデルは既知である．
- ロボットは自由に移動でき，その形状モデルは既知である．
- ロボットの状態（位置，姿勢情報）はすべてプランニングアルゴリズムが把握している．
- 経路プランニングを用いて経路を生成した後にロボットが移動を開始する．

これらは最低限の前提条件である．このほか，環境中に移動可能な物体がある場合（第 3 章）や，協調作業の手順を指定する場合（第 4 章）などタスクによって前提条件が増えることもある．

2.1.2　プランニング方針

前章で述べたような従来手法の持つ限界を克服するには，以下の方針による経路プランニングが必要である．

1. 個別にプランニングする．
2. 優先度付きプランニングを採用する．
3. 効率的な探索アルゴリズムを採用する．

1 点目の個別のプランニングは，1.5.3 節で示したロボットの無駄な動きを減らすことにつながる．

2 点目の優先度付きプランニングは，複数ロボットのプランニングを扱うアプローチの一つである [23]．ロボットごとに優先度を割り当て，優先度の高いロボットから順にプランニングを行うアプローチである．先にプランニングを行ったロボットの経路を固定して，そのあとにプランニングするロボットは先のロボットと衝突を起こさない経路を生成する．これは各ロボットを個々にプランニングすれば良く，方針の 1 点目と親和性が良い．このアプローチでは，その性質上，解くことのできない問題が存在する．しかし多くの問題においては実用的な解が得られると考えられる．

3 点目はマルチエージェントでの効率的な経路プランニングを行う上で必要である．RRT 手法は高次元でも探索できる有望な手法である．しかし 1.5.4 節で示したように，時刻を含めたプランニングで双方向探索を行うためにはゴール状態とともに，ゴール状態に到達する時刻も指定しなければならず，これは実際上不可能である．そこで最終時刻が指定されなくとも効率的な探索手法が必要である．これを実現するアルゴリズムは後述する．

2.1.3　必要とされる機能

これまで多くの経路プランニング問題が研究されたが，目的地までの経路プランニングを行うことが焦点であった．しかし複雑なタスクを実行するには，同じ経路プランニングでも目的や状況に応じて使い分けがされることがある．物体搬送問題 (manipulation planning) では経路プランニングを 2 種類に分ける．それは，物体の近くまでロボットが移動する Transit 経路と，物体を把持したロボットが物体を運搬する Transfer 経路である [12]．ロボットが物体を把持していない状況では Transit 経路を生成し，物体を把持してから Transfer 経路を生成するよう使い分ける．これは物体搬送問題に依存した分類である．

マルチエージェント環境でのタスク遂行を考えると，新たな分類が考えられる．複数のロボットが存在する場合にはサブタスクが割り当てられたロボットとそうでないロボットがあると考えられる．サブタスクは最終目標の前に達成すべき，より小さなタスクであ

る．一般的に，サブタスクには実行順序に関する依存関係があるために，場合によっては他のロボットが作業を終えるまで待機しなければならない．移動をともなうタスクの場合，これには他のロボットの邪魔をしないことも必要である．このように振る舞うには，他のロボットとの衝突を回避する経路上を移動することになる．これには目的地の定まっていない経路プランニングが必要である．この経路プランニングは従来の経路プランニングとは異なる．そこで，経路プランニングをサブタスク割り当てがある場合とない場合で分けて扱う必要がある．本研究ではより具体的に，サブタスクで指定される到達位置（サブゴール）を分類対象とし，このサブゴールの割り当ての有無で分類する．このそれぞれで異なる経路プランニングを行う必要がある．

さらに協調作業タスクではサブタスクと，サブタスクのゴールであるサブゴールを決定することも重要である．例えば，サッカーの最終目標は敵チームに勝つことであるが，そのためには，敵チームのゴールにシュートを決めなければならない．これがサブタスクである．このサブタスクはさらに細かく分けることができる．ゴールにシュートを決めるためには，味方ロボットにパスを出したり，ドリブルをして敵をかわしゴールまで近づく，などのさらに小さなサブタスクも必要となる．本論文ではサブタスクを「2台のロボットで行う協調作業」と定義する．このため，上の例では，味方ロボットにパスを出すなどのレベルがサブタスクとなる．このサブタスクのためのゴール（サブゴール）を人手で指定するのは煩雑である．また適切なサブゴールはそれが解を持つことを保証されているべきだが，適切なサブゴールを人手で探すこと自体が困難なことも多い．このため，サブゴールを自動的に生成できることが望ましい．

以上のことから，マルチエージェント環境でのタスク遂行には，サブゴール割り当ての有無に応じた経路プランニングアルゴリズム，そしてサブゴールの生成アルゴリズムが必要であると考える．

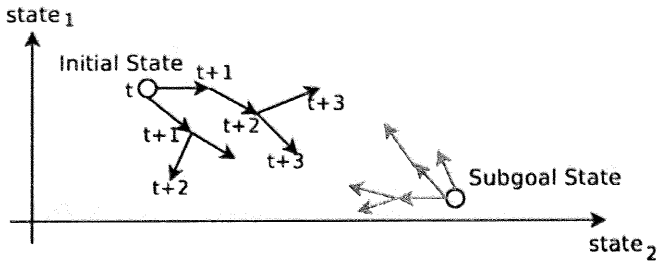
2.2　提案するアルゴリズムの実装

ここでは前節で議論した機能を実現するアルゴリズム群を提案する．それらは，サブゴール割り当ての有無に応じた経路プランニングと，サブゴール生成アルゴリズムである．そして補助的な手続きとして，ゴール状態までの到達可能性検査アルゴリズムも示す．これらはすべてRRT手法を利用して実現されている．

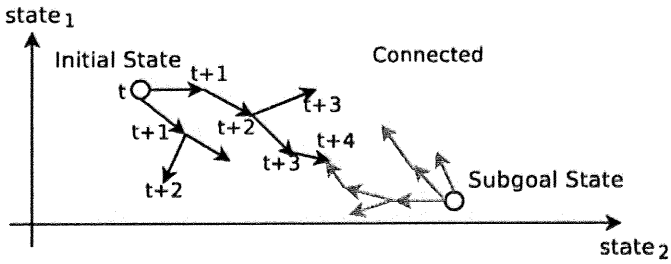
2.2.1　サブゴールが与えられた場合の効率的な経路プランニングアルゴリズム

本節では，サブゴール状態が割り当てられた場合に，ロボットの現在位置からサブゴール状態までの経路をプランニングするアルゴリズムを提案する．以下ではこのルーチンのことをPlanTowardSubgoalルーチンと呼ぶ．

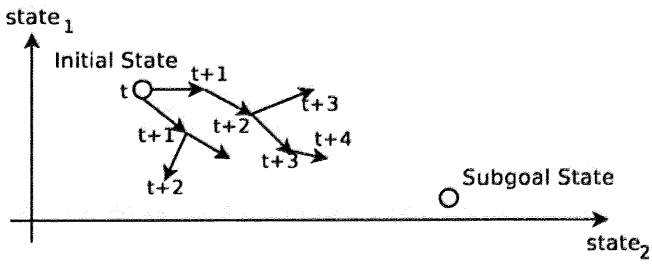
2.1.2節の方針で述べた個別のプランニングを実現するために，時刻パラメータを含め



(a) 両側から探索木を伸ばす



(b) 両探索木が，ある状態でつながる



(c) 両探索木がつながったので，時刻を含まない探索木
を破棄する．その後，探索を続行する．

図 2.1　時刻を含めた探索木と時刻を含めない探索木による双方向探索（時刻軸は紙面に垂直で，2 次元の状態空間に投影した例）

た状態空間内でプランニングする．この場合には 1.5.4 節で述べたように，従来の RRT による双方向探索手法は現実的な手法ではなかった．その理由はゴール状態に到達する最終時刻が事前に必要となってしまうためである．提案アルゴリズムではこの問題を解決する．これは最終時刻を必要としない RRT 双方向探索アルゴリズムである．

提案手法では，時刻パラメータを含めた探索木と時刻パラメータを含めない探索木を用

いる．ロボットのプランニング開始状態 $\boldsymbol{x}_{\mathrm{init}}$ からは時刻を含めた探索木 $T_{w/time}$（すなわち，他のロボットと衝突が発生しない状態を探索する）を成長させる．それと同時に，そのロボットのサブゴール状態 $\boldsymbol{x}_{\mathrm{subgoal}}$ からは時刻を含めない探索木 T_{w/o_time}（すなわち，ほかのロボットが存在しないとして可動領域を探索する）を成長させる（図2.1(a)）．これらを，双方向探索のRRTと同様に，ランダムサンプリングした状態に向けて成長させる．両探索木がつながった場合には（図2.1(b)），T_{w/o_time} を再初期化（すなわち途中結果を破棄）し，探索を続ける（図2.1(c)）．このとき，$T_{w/time}$ は保持したままである．このようにすると，時刻を含めた探索木 $T_{w/time}$ の成長が，時刻を含めない探索木 T_{w/o_time} によって誘導されるので，探索の効率が向上する．

PlanTowardSubgoal ルーチンの具体的な手順は以下のようになる．

1. 初期状態を根とする探索木 $T_{w/time}$（時刻パラメータを含めた探索を行う）とゴール状態を根とする探索木 T_{w/o_time}（時刻パラメータを含めない探索を行う）を生成する．
2. ランダムに状態 $\boldsymbol{x}_{\mathrm{rand}}$ を選ぶ．
3. 探索木 $T_{w/time}$ 中で $\boldsymbol{x}_{\mathrm{rand}}$ に最も距離の近い状態から $\boldsymbol{x}_{\mathrm{rand}}$ に向けて枝を伸ばす．衝突がなく枝を伸ばすことができたなら，その枝の末端状態を $\boldsymbol{x}_{\mathrm{new}}$ とする．この際，$\boldsymbol{x}_{\mathrm{new}}$ は移動する距離分の時刻を進める．枝を伸ばすことができなければ，ステップ5. に進む．
4. 探索木 T_{w/o_time} から $\boldsymbol{x}_{\mathrm{new}}$ に向けて枝を伸ばす．T_{w/o_time} から $\boldsymbol{x}_{\mathrm{new}}$ に向けて枝を伸ばすことができたならば，その枝の末端状態を $\boldsymbol{x}'_{\mathrm{new}}$ とする．枝を伸ばすことができなければ，ステップ5. に進む．
5. $\boldsymbol{x}_{\mathrm{new}}$ がゴール状態に等しいならば，探索成功．初期状態からゴール状態までの経路を返して終了．
6. $\boldsymbol{x}_{\mathrm{new}}$ と $\boldsymbol{x}'_{\mathrm{new}}$ が等しいならば，探索木 T_{w/o_time} の結果を破棄する．
7. 探索木 $T_{w/time}$ と T_{w/o_time} の枝を伸ばす順序を逆にして，ステップ2. から繰り返す．

時刻パラメータを含めた探索では，他のロボットが経路プランを持っている場合にはそのプランを使用し，経路プランがない場合にはその場に静止しているとして，衝突検査を行う．このアルゴリズムを用いることで，最終状態に達する時刻をあらかじめ定めなくとも，RRTで双方向探索ができる．

本研究では，双方向探索以外にサブゴール状態に向けたEXTEND操作も用いた．これにより，サブゴール状態に直進できる場合にはより効率的な探索が可能である．

2.2.2　サブゴールが指定されない場合の経路プランニングアルゴリズム

ロボットにサブタスクが割り当てられずサブゴールが指定されない場合には，サブタスクが割り当てられるまで他のロボットの邪魔とならない行動系列が必要である．ここでは

行動系列を得るべき時刻（開始時刻と終了時刻）が指定されるものとする．

本節で提案するアルゴリズムでは，指定された時刻まで他のロボットと衝突せずにいる経路を探索する（以下では PlanWithoutSubgoal ルーチンと呼ぶ）．通常，目的地の定まっていない経路生成は難しい．しかし RRT を応用することで以下のように実現できる：現在衝突のない状態にいるなら，次の時刻でも同じ状態にとどまってみる．もし衝突が起きるならば，衝突を回避する経路をプランニングする．ここで RRT 探索を用いる．これを繰り返し，必要とする終了時刻 t_{end} まで経路プランができたところで終了する．RRT 探索は乱数を用いるので，目的地が定まっていなくとも環境中を移動する経路を得られる．

PlanWithoutSubgoal ルーチンは具体的に以下のアルゴリズムで実現できる．

1. 初期状態を $\boldsymbol{x}_0$，そのときの時刻を t_0 とする．$(\boldsymbol{x} \leftarrow \boldsymbol{x}_0, t \leftarrow t_0)$．
2. もし状態 $\boldsymbol{x}$ がほかのロボットと衝突するならば，衝突がなくなるまで時刻 $t-1$ からランダムな状態に向けて RRT 探索をする．衝突がなくなるまでの経路を記録し，最後の状態 $\boldsymbol{x}'$ と時刻 t' を $\boldsymbol{x}, t$ に設定し，ステップ 4. に進む $(\boldsymbol{x} \leftarrow \boldsymbol{x}',\ t \leftarrow t')$．
3. もし状態 $\boldsymbol{x}$ が衝突のない状態なら，その状態を経路に追加し，時刻を進める $(t \leftarrow t+1)$．ステップ 4. に進む．
4. $t \geq t_{\mathrm{end}}$ ならば，探索成功である．経路を返して終了．
5. ステップ 2. に戻る．

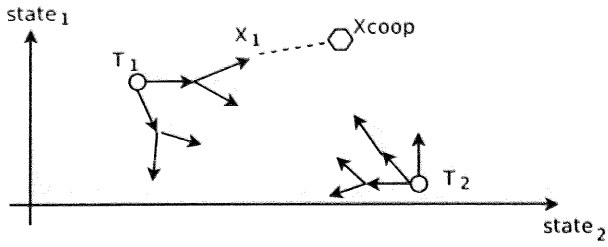
ここで，RRT 探索に用いるのは時刻パラメータを含めた探索木であり，初期状態 $\boldsymbol{x}_0$ は衝突を持たない状態であるとする．これによって時刻 t_0 から t_{end} までの衝突のない経路が得られる．

2.2.3　サブゴールの生成アルゴリズム

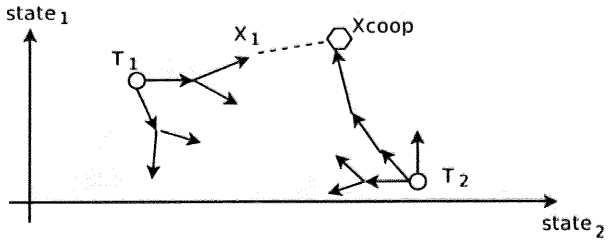
ここではサブゴールを「2 台のロボットが協調作業のための条件を同時に満たす状態ペア」と定義する．例えば第 4 章で説明するような 2 台のロボットによる荷物の受け渡しのような作業を行う場合には，ロボット同士が向き合って一定距離まで近づかなければならない．この受け渡し可能な状態のことをサブゴール状態とする．これは 2 台のロボットそれぞれに状態が指定される必要があるため，一組の状態ペアとして扱う．

この協調作業のために状態ペアが満たすべき条件を「サブゴール条件」と定義する．この条件は例えば，ロボット同士がある一定距離離れた位置で互いに向き合う状態などと定量的に定義できる．一般的に，この条件は一定範囲の余裕があると考えられる．例えば荷物受け渡しの例で言えば，ロボットの腕が届く範囲内ならば，位置が前後したり向き合う角度がずれても受け渡しを行うことができるはずである．つまり，ペアの一方の状態を定めたとしても，条件を満たすもう一方の状態は複数ありうるとして扱う．そこでサブゴール条件は，状態ペアが協調作業可能となる条件として扱う．

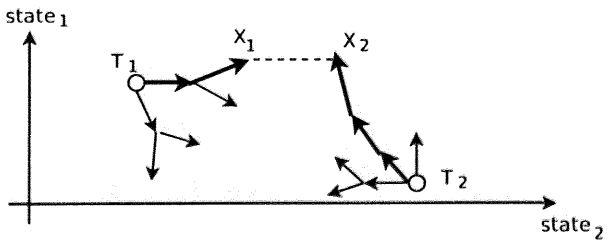
協調作業を自動化するためには，サブゴール条件を満たす状態ペアを環境中から見つけ



(a) $\boldsymbol{x}_1$ と受け渡し作業を行う標準状態 $\boldsymbol{x}_{\mathrm{coop}}$ を求める



(b) 探索木 T_2 から $\boldsymbol{x}_{\mathrm{coop}}$ に向けて枝を伸ばす



(c) $\boldsymbol{x}_1$ とのサブゴール条件を満たす状態 $\boldsymbol{x}_2$ を見つける

図 2.2　サブゴール生成の手順（状態空間が 2 次元での例．点線は両端の状態がサブゴール条件を満たしていることを表す．）

る必要がある．ペアの一方の状態を定めても，もう一方の状態を見つけ出すことは難しい．複数の可能性があるためである．しかもサブゴール条件を満たす状態ペアであればどのような状態でも良いわけではない．なぜなら，ロボットの現在状態からサブゴール条件を満たす状態まで到達できない（可能な状態空間 $\boldsymbol{C}_{\mathrm{free}}$ 中に経路がない）場合があるからである．そこで，サブゴールまでの経路も保証できることが望ましい．以下で示すアルゴリズムはサブゴールまでの経路もプランニングしながらサブゴールを決定する．また，サブゴールの生成では，協調する二体のロボット以外のロボットを無視してプランニングを行なう．これはプランニングに関して，協調行動を行なう二体のロボットに，その他のロボットに対する優先度を持たせることに相当する．

以下のアルゴリズムは双方向探索を行う RRT（1.5.2 節）を応用している．二つの RRT は協調する二体のロボットの現在の状態からそれぞれ成長させる．そして，双方の木から

互いに「サブゴール条件を満たす標準状態 $\boldsymbol{x}_{\mathrm{coop}}$」に向けて探索木を伸ばす．標準状態とは，一方の状態を定めたときにもう一方がサブゴール条件を満たすような状態のうちの具体的な一つの状態をさす．例えば，ロボット同士の距離が Lcm（L は定数），向き合った角度のずれがゼロ，などである．伸ばした枝の状態が環境との衝突を起こさなければ探索木に追加する．条件を満たす状態のペアが見つかればそれをサブゴールとして返す．

サブゴールの生成は以下のようなアルゴリズムで行う（GenerateSubgoal ルーチンと呼ぶ）：

1. ロボット r_1, r_2 の現在状態を根とする探索木をそれぞれ T_1, T_2 とする．
2. ランダムに状態を選び $\boldsymbol{x}_{\mathrm{rand}}$ とする．
3. T_1 から $\boldsymbol{x}_{\mathrm{rand}}$ に向けて CONNECT 操作を実行する．T_1 から $\boldsymbol{x}_{\mathrm{rand}}$ に向けて枝を伸ばすことができたならば，その枝の末端状態を $\boldsymbol{x}_1$ とする．枝を伸ばすことができないならば，ステップ 7. へ進む．
4. ロボット r_1 の状態を $\boldsymbol{x}_1$ と仮定して，それと協調するロボット r_2 の標準状態 $\boldsymbol{x}_{\mathrm{coop}}$ を求める（図 2.2(a)）．
5. T_2 から $\boldsymbol{x}_{\mathrm{coop}}$ に向けて CONNECT 操作を実行する．T_2 から $\boldsymbol{x}_{\mathrm{coop}}$ に向けて枝を伸ばすことができたならば，その状態と枝を T_2 に追加する（図 2.2(b)）．
6. T_2 に含まれる状態の中から，$\boldsymbol{x}_1$ とのサブゴール条件を満たす状態 $\boldsymbol{x}_2$ を探す（図 2.2(c)）．$\boldsymbol{x}_2$ が見つかれば，サブゴール生成は成功であり，二体のロボット r_1, r_2 のサブゴールとして $\boldsymbol{x}_1, \boldsymbol{x}_2$ をそれぞれ出力として返して終了．
7. T_1 と T_2 の役割を入れ替えて，ステップ 2. から繰り返す．

探索終了時の $\boldsymbol{x}_1$，$\boldsymbol{x}_2$ がサブゴール状態ペアとなる．このサブゴール生成では状態に時刻を含めずにプランニングする．時刻を含めずプランニングを行なうため，高速なプランニングが可能である．

このアルゴリズムでは状態ペア（$\boldsymbol{x}_1$，$\boldsymbol{x}_2$）を求めると同時に，サブゴール状態に至るまでの経路も確認されている．しかしここで得られた経路プランは時刻が考慮されていないため，他のロボットが存在する環境では必ずしも実現可能ではない．これを保証するには第 4 章で示すアルゴリズムのように，サブゴールを生成した後に，時刻を含めた経路プランニングを改めて実行する．これはサブゴールが定まっていれば PlanTowardSubgoal ルーチン（2.2.1 節）で効率的な経路プランニングができるためである．

このアルゴリズムはサブゴール条件をタスクに応じて変更することで，ほかのサブゴール探索に応用可能である．例えばロボットサッカーでのパス行動（一方がパスを出し，もう一方がパスを受ける）をサブゴールとしてサブゴール状態を生成するには，パスを受けるロボットがパスされるボールの進行方向直線上にあることをサブゴール条件として定義できるはずである．

2.2.4　到達可能性検査アルゴリズム

このルーチンでは，指定されたロボットが現在状態から目標状態まで到達できるかどうかを検査する．以下ではTestReachableルーチンと呼ぶ．これはロボットの現在状態から目標状態まで障害物と衝突を起こさずに移動する経路が存在すればよい．

このルーチンでは，目標状態への到達可能性をチェックするだけであり，厳密なプランニングまで行う必要はない．さらに，高速にチェックするために計算量が少ない方が良い．そこで時刻パラメータを含めないプランニングを利用する．これはRRTを用いた双方向探索手法であるRRT-ConCon（1.5.2節）で実現できる．アルゴリズムが必要とするのはロボットの初期状態と目標状態である．目的状態までの経路が存在すればtrueを返し，経路が見つけられなければfalseを返す．

このアルゴリズムでは，時刻パラメータを含めた厳密なプランが得られない．厳密なプランを得るには，前記したPlanTowardSubgoalルーチンと併用する必要がある．

2.2.5　探索の打ち切り条件

RRTを用いた探索は確率的な探索であるため，初期状態とゴール状態をつなぐ経路の存在が確率的にしか判定できない．つまり探索を何度か繰り返して失敗しても，それが経路が存在しないためなのか，それともさらに繰り返せば経路が得られるのかは分からない．

本研究ではヒューリスティクスをもちいて探索打ち切りを決定する．探索を一定回数繰り返して得られた最短距離を終了判定の指標として用いる．一定回数のうちにゴール位置までの距離が閾値以下にならなければ，そのような経路は存在しないとみなし，探索を打ち切る．閾値よりも小さくなれば，さらに長時間かけて探索する．

第3章

マルチエージェント環境での物体再配置問題への適用

物体の再配置問題（rearrangement problem）[3, 32] はロボットの行動プランニングにおける重要な問題である．この問題ではロボットを用いて物体の移動を効率的に行う必要がある．再配置問題を解くアルゴリズムはこれまでにいくつか研究されている [3, 31, 32]．マルチエージェント環境では，適切な作業の割り当てを行うことで，ロボットが1台だけの場合よりも短時間でタスクを完了させられると期待できる．その代わりに，複数のロボットが同時に行動するために問題の複雑さが増大する．環境中の障害物だけでなく，他のロボットが移動する障害物となり，シングルエージェント環境に比べて動的で複雑になるためである．この困難さを解決するには特にロボット1台あたりの行動プランニングの計算量が問題となる．そこで行動プランニングのための計算量をできるだけ抑える必要がある[*1]．

再配置問題に関連して，物体を移動する行動を得るプランナがこれまでにいくつか提案されている．Wright [12] らは物体を移動する行動プランニングを扱った．そこで示されたアルゴリズムはロボット1台が1つの物体を移動する場合しか扱えない．Ben-Shahar ら [3] の行動プランナは状態空間でのダイナミックプログラミングに基づいている [4]．しかし複数のロボットが存在する空間では，状態空間がロボット数の指数関数でふくれあがるため，彼らのプランナをマルチエージェント環境に拡張するのは現実的ではない．Ota も再配置問題のために行動プランナを用いているが，ロボット1台の場合しか扱っていない [31]．Gravot らはシンボルによるプランニングと経路プランニング手法を組み合わせて複数ロボットによる再配置問題を実現した [16]．しかし彼らの手法ではロボットは同時に一台が行動するだけであり，実応用を前提とした場合には非効率的である．

本章では，前章で示したマルチエージェント環境のための経路プランニングアルゴリズムを用いて，マルチエージェント環境（ロボットが1台の場合も含める）での再配置問題

[*1] 本章では経路プランニングと行動プランニングをほぼ同義として扱う．なぜならばホロノミック環境を仮定したシミュレーションでは経路プランとそれを実現する行動プランは同時に得られるからである．

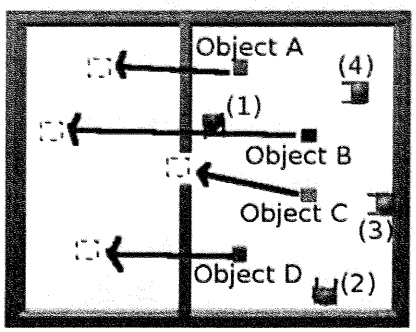


図 3.1　問題 P-2 （括弧内の数字はロボットの番号を示す.）

を解決する．提案した経路プランニングアルゴリズムを用いることで，ロボット数を増やしても計算量が爆発することなくプランを得られることを示す．

マルチエージェント物体再配置問題では，マルチエージェント環境であることによる難点も解決しなければならない．それはロボットに複数の行動プランをどのように割り当てるかという方針や，物体の搬送作業を割り当てるべきロボットの決定方法である．これら問題は最適化問題として定式化することが困難なため，本研究では最適解を追究するのではなく，可能な限りロボットを同時並行して行動させることを目的とする．そのために，ロボット行動の無駄時間（アイドル時間）を少なくする行動割り当ての方法を示す．実応用を考えた場合には，ロボットのアイドル時間が減るほうが有用である．さらに，再配置問題に用いられる優先度グラフ（precedence graph）への拡張も提案する．

3.1　マルチエージェント物体再配置問題

物体の再配置問題とは，1 つ以上の物体を初期状態からそれぞれのゴール位置まで適切に移動するようなロボットの行動系列を得る問題である．本研究では複数のロボットが存在しうる環境を考える．ロボットはどの物体を運んでも良く，ロボットのゴール状態は与えない．本研究ではロボットが物体をつかむことを想定しているため，ロボットが物体と接触しつかんだときには，ロボットと物体を一体として扱う．環境には実数値 2 次元空間を用いる．複数ロボットによる再配置問題は最適化問題としての定式化が困難である．そのため本研究では解の最適性（最短でタスクを終了する経路プランを得ることなど）にはこだわらない．

有効性を確認するために，本研究では文献 [3] で用いられている問題 P-2, P-4, P-5 を解く（図 3.1, 3.2, 3.3）．ただし，文献 [3] ではロボット 1 台について扱っていたが，本研究ではロボットを 4 台にまで拡張して用いる．いずれの問題も荷物のゴール位置は指定されているが，ロボットのゴール位置や行動順序などは指定されていない．

物体の再配置問題では，優先度グラフ（precedence graph） [3] を用いて，物体の移動順序を決定できる．ただし優先度グラフがサイクルを持つ場合には，物体同士が移動を妨

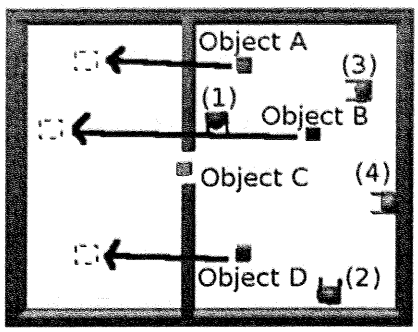


図 3.2　問題 P-4 （括弧内の数字はロボットの番号を示す.）

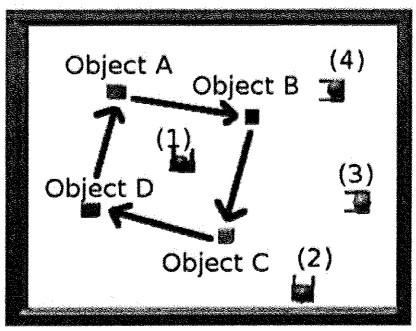


図 3.3　問題 P-5 （括弧内の数字はロボットの番号を示す.）

げており，単純に目的位置まで移動できない．その場合には，妨げとなっている物体を一時的な場所に待避したうえで，目的位置まで運ぶ必要がある．Ben-Shahar らは，優先度グラフがサイクルをもつ場合にはヒューリスティクスを用いて，一時待避する物体を決定した [3]．彼らの手法ではグラフにサイクルがなくなるまでこのヒューリスティクスを繰り返す必要がある．

以下では，拡張した優先度グラフを提案する．このグラフを用いることで，特定の条件を満たす場合にヒューリスティクスに頼らずに一時待避すべき物体を決定できる．

3.1.1　拡張優先度グラフ

従来の優先度グラフでは物体それぞれを一つのノードとして表現し，その順序関係をグラフで表現していた [3]．

われわれが提案する拡張優先度グラフでは，1 つの物体を表現するために 2 つのノードを用いる．一つは物体を初期位置から移動する（取り除く）作業を示すノードであり，もう一つはその物体をゴール位置に置く作業を示すノードである．以下では便宜上，物体 X に関して前者のノードを init ノードと呼び，X(init) と表し，後者のノードを goal ノードと呼び X(goal) と表すことにする．この拡張優先度グラフは以下のアルゴリズムで作成できる．

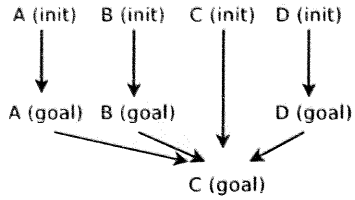


図 3.4　問題 P-2 の拡張優先度グラフ

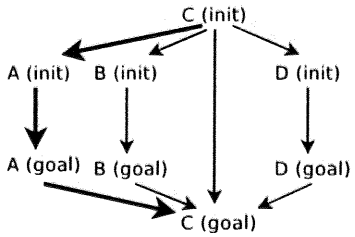


図 3.5　問題 P-4 の拡張優先度グラフ

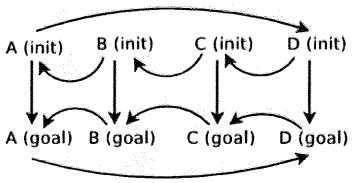


図 3.6　問題 P-5 の拡張優先度グラフ

1. すべての物体 X について，init ノードから goal ノードに向かう辺をつなぐ．これは物体を初期位置から取り除いた後にゴール位置に配置することを意味し，自明な順序である．(X(init) → X(goal))
2. 以下のステップ (a), (b) をすべての物体 X と物体 Y (Y ≠ X) の組み合わせについて繰り返す．
   (a) ある物体 X が初期位置にいる状態で，物体 Y が初期位置からゴール位置まで移動できない場合には，X(init) → Y(init) という辺を生成する．(これは物体 X をその初期位置から取り除いた後で物体 Y を移動するという順序関係を示す．)
   (b) ある物体 X がゴール位置にいる状態で，物体 Y が初期位置からゴール位置まで移動できない場合には，Y(goal) → X(goal) という辺を生成する．(物体 Y をそのゴール位置まで移動してから物体 X をゴール位置に移動するという順序関係を意味する．)

例えば，問題 P-2, P-4, P-5 の優先度グラフを生成するとそれぞれ図 3.4，図 3.5，図 3.6 のようになる．

3.1.2　一時待避が必要な場合

拡張優先度グラフでは，物体の一時待避が必要な場合は以下の 2 通りに分類できる．

1. ある物体の移動が他の物体を移動する障害となる場合．
2. 拡張優先度グラフがサイクルを持っている場合．

1. の場合は，ある物体 X が初期位置にあると他の物体 Y が移動できず，しかも物体 Y を移動せずには物体 X をゴール位置まで運べない場合である．この場合，物体 Y を移動する前に物体 X を一時待避する必要があるのは自明である．拡張優先度グラフで説明すると，ある物体 X の init ノードからリンクをたどって別の物体のノード（init ノード，goal ノードのいずれでも良い）を経由し，物体 X の goal ノードに到達できる場合に相当する．

図 3.5 が 1. の場合に相当する．図中の太い矢印で示したリンクをたどると，C(init) → A(init) → A(goal) → C(goal) となり，物体 C の init ノードから物体 A のノードを経由して，物体 C の goal ノードまで達することができる．このことから，物体 A を移動する前に物体 C の一時待避が必要であることが判定できる．

図 3.6 は 2. の場合に相当する．この場合には A(init)~D(init), そして A(goal)~D(goal) の中でサイクルができている．この 2. の場合には，どの物体を一時待避すべきかは自明ではない．そこで，Ben-Shahar らのヒューリスティクス [3] を用いて待避すべき物体を決定する．

1., 2. のいずれの場合でも，対象物体のための中間状態（待避場所）を生成し，その状態まで物体を移動する必要がある．どこに待避すれば良いかは，その後の物体とロボットの移動経路に依存するため，容易には求まらない．

本研究では一時待避の場所を初期位置から一定距離離れた場所と定め，その距離だけをあらかじめ指定することとした．この待避場所は，RRT による経路プランニングを応用し，待避すべき物体とそれを運ぶロボットの移動経路をプランニングすると同時に決定する．詳細は 3.3.1 節で説明する．

3.2　行動計画アルゴリズム

提案手法のアルゴリズムの流れは以下のようになる（以下では PlanRearrangement ルーチンと呼ぶ）．

1. 拡張優先度グラフを生成する．
2. グラフがサイクルを持つ場合には，一時待避を行い，再帰的に PlanRearrangement ルーチンを呼ぶ．
3. グラフを深さ優先探索でたどりながら，それぞれの物体について再帰的に経路プランを生成する（3.2.1 節）．

4. 経路プランが生成できていれば，すべてのロボットの行動プランを同じ長さにする．

物体の移動順序は文献 [3] にもとづき，優先度グラフをたどりながら解決する．ステップ 2. の深さ優先探索では，順序関係を崩さない限りにおいて，移動経路長の短いものから順に実行する．この移動経路長は予備的な経路プランニングを行い，自動的に求める．もし一時待避の場所が適切でなく，その後の経路プランが生成できない場合には，バックトラックし，別の場所に一時待避を行い，経路プランニングをやり直す．

3.2.1　優先度グラフの解決

物体移動の順序関係は，優先度グラフでノードから出る矢印と逆向き（子ノードから親ノードの向き）の深さ優先探索でたどることで満足される．この探索の正しさは，先行する物体（親ノード）が処理されてから子ノードが処理されるという順序関係から明らかである．

深さ優先探索がたどるノードごとに，以下のルーチン（DetermineObjectToMove ルーチンと呼ぶ）を利用して，経路生成を行うべき物体を決定する．そして必要に応じて経路生成を行う．

1. 物体の init ノードであれば，物体の一時待避が必要かどうか（3.1.2 節の 1. の場合）判定する．
   (a) 一時待避が必要ならば一時待避ルーチンを実行して，再帰的に PlanRearrangement ルーチンを呼ぶ．
   (b) 一時待避が必要なければ，この物体の goal ノードに遷移する．
2. 物体の goal ノードであれば，先行物体が移動済みか検査する．
   (a) 先行物体が移動済みでなければ，それらを移動するために再帰的に DetermineObjectToMove ルーチンを実行する．
   (b) 先行物体が移動済みならば，その物体をゴール位置まで移動する（次節で説明する）．

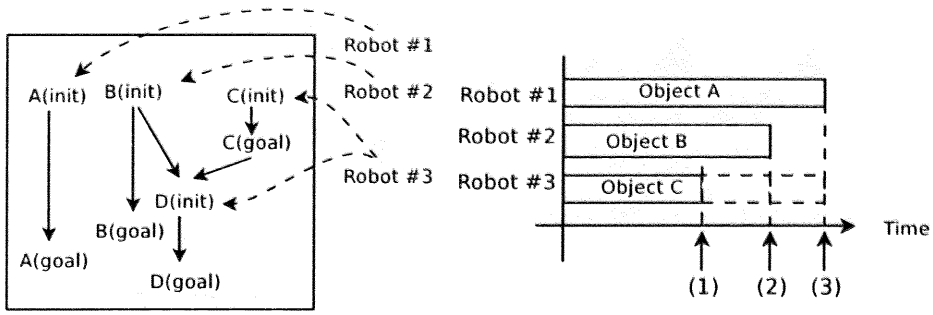
このようにしてそれぞれの物体について経路を生成していく．

一時待避が必要な場合には，物体の一時的な待避位置を決定して，ロボットで物体を移動する．そして移動後の環境に対して PlanRearrangement ルーチンを呼び出す．もし一時待避の場所が適切でなく，その後の経路プランが生成できない場合には，バックトラックし，別の場所に一時待避を行い，経路プランニングをやり直す．

3.2.2　物体の移動経路生成

物体を移動するためには，まずロボットが物体の位置まで移動し，次に荷物を持ったロボットが物体をゴール位置（もしくは一時待避位置）まで移動する．

ただし複数のロボットが存在するために，どのタイミングでも物体の移動ができるわけ



(a) 拡張優先度グラフと物体に割り当てたロボット

(b) ロボット#3 に物体 D までの経路プランを割り当てる場合の作業開始時刻の候補 (1), (2), (3)

図 3.7　ロボットに経路プランを割り当てるための開始時刻候補の例

ではない．ロボットが物体を移動する前に以下のような場合がありうる．

- 優先度グラフ中で先行する物体が移動完了するまでロボットが移動できない場合．
- 優先度グラフ中で先行する物体が移動完了するまで物体が移動できない場合．
- すべてのロボットが移動完了するまで物体が移動できない場合．

これら条件が当てはまるか判定しながらプランニングを行う必要がある．実際には，これらから導出された以下の場合分けに沿って処理を行う．

まず，ロボットが物体の現在位置まで移動するために，次の 3 つのタイミングを開始時点として物体までの経路生成を順に試みる．

(R1) ロボットが前の作業を終えた直後（図 3.7(b) 中の (1)）．
(R2) 優先度グラフでの先行する物体（とそれを移動するロボット）の移動が終わった直後（図 3.7(b) 中の (2)）．
(R3) これまでに経路プランのあるすべてのロボットと物体が移動を終えた直後（図 3.7(b) 中の (3)）．

上の 2 つの場合（(R1), (R2)）は他に物体を移動しているロボットがいて干渉する場合に経路を生成できないことがある．それらによる待ち時間がある場合には，PlanWithoutSubgoal ルーチン（2.2.2 節）を使って，開始時刻まで他のロボットと干渉しないような経路プランで埋める．ここで，(R1) の場合には優先度グラフの順序関係を無視しており，一見すると不要な場合分けに見える．しかし複数のロボットを同時並列的に行動させていると，必ずしも移動できないわけではなく，さらに全体のタスク実行時間を減らすことができる．ロボットのアイドル時間が減らせる可能性があるため，(R1) の場合分けも有用である．これが有用な事例は 3.6.1 節で示す．

図 3.7 に場合分けの例を示した．図 3.7(a) のように，物体 A~D の 4 つの物体が拡張

優先度グラフで表されており，そのそれぞれの移動にロボット#1~#3 が割り当てられたとする．図 3.7(b) は，物体 A~C を移動するためにロボットに割り当てられた経路プランの開始時刻と終了時刻を表した図である．この状況からロボット#3 で 物体 D を動かすプランを作るには，先ほどの場合分けを順に試みる．まずロボット#3 が物体 C を運び終わった時点 (1) で試みる．この例の場合には優先度グラフを見ると，物体 B を運び終わってからでないと物体 D を運べない．しかし (1) 時点ではロボット#2 がすでに物体 B を移動しているため，物体 D を動かし始められる可能性がある．全体の作業時間を減らすためには試みるべきである．そのプランニングに失敗する場合には物体 B の移動が完了した時点 (2)，それでも失敗する場合には物体 A の移動完了時点 (3) で試みる．それでも経路が得られない場合は失敗であり，バックトラックが必要である．

ロボットが物体に到達した後には，ロボットが物体を持ち，物体のゴール位置まで移動する．そのときには次の 3 つのタイミングを開始時点として順に試みる．

(M1) ロボットが物体に到達した直後．
(M2) 優先度グラフで先行する物体の移動が終わった直後．
(M3) これまでに経路プランのあるすべてのロボットが移動を終えた直後．

これらの経路生成には，PlanTowardSubgoal ルーチン（2.2.1 節）を利用する．上 2 つの場合（(M1)，(M2)）では，他に物体を移動しているロボットがいる場合に干渉のため経路を生成できないことがある．

3.2.3　タスクへのロボットの割り当て

このマルチエージェント環境ではどの物体をどのロボットが運ぶかを決定しなければならない．しかし本研究のタスクでは，作業時間を最小とするなどの最適なスケジューリングは困難である．なぜならば他のロボットの行動に応じて，ロボットの経路長や行動が変わるためである．そのため全探索以外では最適解を求められないと考えられる．そこで，本研究ではロボットへの最適な作業割り当てではなく，ヒューリスティクス関数による割り当てを行う．

ロボットへの作業の割り当てに際し，複数のロボットが同時に行動するため，ある時点では適当なロボットが他の物体の移動作業をしている場合がある．ロボットに作業を割り当てるには，すでに作業中だが物体までの距離が短いロボットや，アイドル状態にあっても物体までの距離が遠いロボットなどがありうる．その中から適切なロボットを選択しなければならない．

そこで，本研究ではロボットごとに以下のようなコスト値 Cost_i を求め，最小コストとなるロボット i $(i = 1, 2, ..., (\text{ロボット数}))$ を割り当てた．

$$\begin{aligned}\mathrm{Cost}_i = &(\text{現時刻から作業終了までの時間})+\\ &(\text{現在地から物体までの移動に要する時間})\end{aligned} \tag{3.1}$$

第 2 項の「移動に要する時間」の正確な値は，他のロボットも考慮に入れて経路をプランニングしなければ分からない．計算量を省くため，他のロボットが存在しないとしたときの経路をその見積もりとして用いた．物体はそれぞれその時点で得られている最終状態に固定してプランニングする．もしロボットが（障害物などによって）物体に到達することができなければ，候補からはずす．

3.3　複数ロボット環境のための経路プランニングアルゴリズム

プランニングには前章で提案した経路プランニングルーチン PlanTowardSubgoal と PlanWithoutSubgoal，そしてロボットが物体まで到達できるかどうかを検査するために TestReachable を用いている．本章ではそのほかに，一時待避場所を決定する PlanIntermediateState ルーチンにも RRT を用いた．また，拡張優先度グラフの生成は計算量が多いため，不要な計算を省いたアルゴリズムを用いた．以下ではそれらの実装を説明する．

3.3.1　PlanIntermediateState ルーチン

本研究では，物体の一時待避場所（状態）とそこに至る経路を RRT を応用して探索する．このときはロボットが物体を持っている状態なので，衝突検査には荷物を持ったロボットのモデルを用いる．このアルゴリズムでは一時待避場所までの距離を l としてあらかじめ設定しておく．通常の RRT と異なる点は，指定された距離だけ離れた状態までの経路を探索することである．アルゴリズムは以下に示した．

1. 初期状態を根とする探索木 T を生成する．
2. 初期状態から距離 l 以上離れた状態 $\boldsymbol{x}_{\mathrm{rand}}$ をランダムに選ぶ．
3. 探索木 T 中で状態 $\boldsymbol{x}_{\mathrm{rand}}$ に最も近い状態から $\boldsymbol{x}_{\mathrm{rand}}$ に向けて枝を伸ばす．干渉がなく枝を伸ばすことができたならばその状態を $\boldsymbol{x}_{\mathrm{new}}$ とする．
4. $\boldsymbol{x}_{\mathrm{new}}$ が，初期状態から距離 l 以上離れていれば，探索成功．初期状態から $\boldsymbol{x}_{\mathrm{new}}$ までの経路を返して終了．
5. ステップ 2. に戻る．

ここで探索木には時刻パラメータを含めて，他のロボットや物体との干渉もチェックする．

3.3.2　優先度グラフの生成アルゴリズム

優先度グラフの生成（3.1.1 節）では物体数を n とすると n^2 の組み合わせを検査しなければならず，経路プランニングに要する計算時間の影響が非常に大きい．

そこで本研究では不要な組み合わせの検査を省いた以下のようなアルゴリズムを用

いた．

1. ゴール位置まで移動すべき物体の集合を $\boldsymbol{M}$ とする．
2. すべての物体 X （X $\in \boldsymbol{M}$）についてそれぞれステップ (a) ～ (d) を繰り返す．
   (a) すべての物体を集合 $\boldsymbol{O}_{fix}$ に入れる（$\boldsymbol{O}_{fix} \leftarrow \boldsymbol{M}$）．
   (b) 物体 X 以外の物体が存在しないとして，物体 X の初期位置からゴール位置までの移動経路 P を生成する．
   (c) P とすべての物体の位置の重なりをチェックし，P と重なる物体は移動すべき物体の候補となる集合 $\boldsymbol{O}_{move}$ に入れ，$\boldsymbol{O}_{fix}$ から除く．
   (d) 移動すべき物体候補中の物体 Y （$Y \in \boldsymbol{O}_{move}$）すべてについて，ステップ i. ii. を繰り返す（3.1.1 節の処理に相当する）．
      i. $\boldsymbol{O}_{fix}$ 中の物体と，物体 Y をその初期位置に固定した場合について，物体 X の移動経路を生成する．物体 X の経路が生成できない場合には，優先度グラフの対応する辺を追加する．
      ii. $\boldsymbol{O}_{fix}$ 中の物体と，物体 Y をそのゴール位置に固定した場合について，物体 X の移動経路を生成する．物体 X の経路が生成できない場合には，優先度グラフの対応する辺を追加する．

このようにすることで，物体のすべての組み合わせを計算するよりも少ない組み合わせで優先度グラフを生成することができる．ただし最悪時の計算量はすべての組み合わせを試す場合と同じであり，$O(n^2)$ である．

3.3.3　探索の打ち切り条件

RRT を用いた探索は確率的な探索であるため，初期状態とゴール状態をつなぐ経路の存在が確率的にしか判定できない．つまり探索を何度か繰り返して失敗しても，それが経路がないためなのか，それともさらに繰り返せば経路が得られるのかは分からない．

そこで，本研究では探索を一定回数繰り返して得られた最短距離を終了判定の指標として用いる．一定回数のうちにゴール位置までの距離が閾値以下にならなければ，そのような経路は存在しないとみなし，探索を打ち切る．閾値よりも小さくなれば，さらに長時間かけて探索する．

3.4　実験とその結果

以上で説明したアルゴリズムを用いて，マルチエージェント環境での再配置問題を解くことができる．

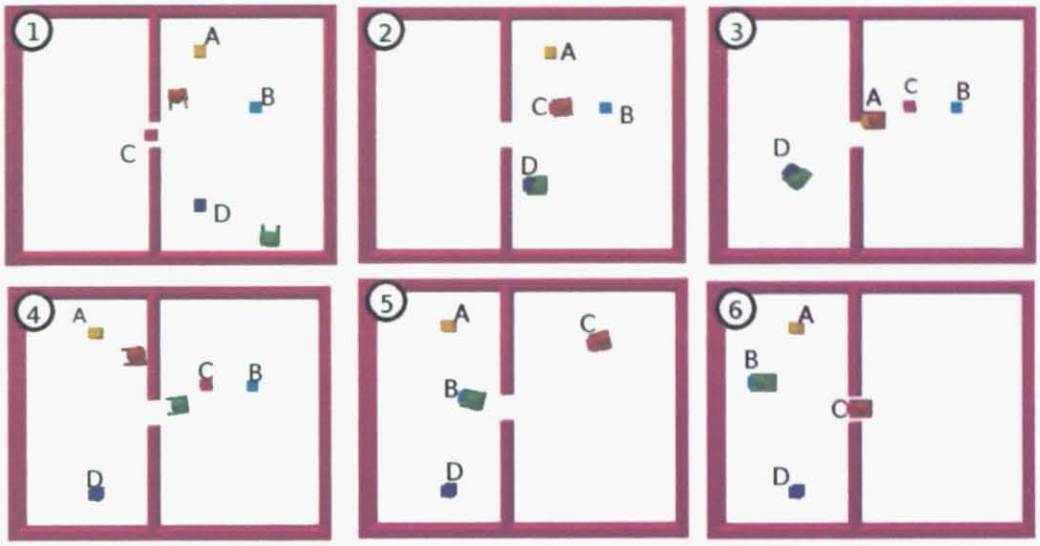


図 3.8　問題 P-4 （ロボット数 2）で得られた行動の例

3.4.1　再配置問題での実験結果

図 3.1, 3.2, 3.3 の問題を用いてシミュレーション実験を行った．ロボット数は 1，2，4 と変化させて実験を行った．ロボット数が 4 より少ない場合には，ロボットのインデックスが小さいほうからその数だけのロボットを用い，それ以外のロボットは無視した．問題 P-2 では物体 C を最後にゴール位置に運ばなければならない．問題 P-4，P-5 では物体の一時待避が必要である．

得られた結果の例として問題 P-4 を解いた結果を図 3.8（ロボット数 2），図 3.9（ロボット数 4）に示した．ロボット数 2 の場合には図 3.8(2) で，まず物体 C が初期位置から離れた場所に一時待避された．それによってできた通路を使ってロボットはほかの荷物の移動作業を行い，最後に物体 C をゴール位置まで移動してタスクを達成した．ロボット数 4 の場合は図 3.9(2)-(4) のようにロボットが物体 C を一時待避したまま最後まで待機する作業が得られた．これはロボットが 4 台いるため，それぞれの物体にロボットを割り当てられるためである．

問題 P-5 を解いた結果を図 3.10（ロボット数 2），図 3.11（ロボット数 4）に示した．ロボット数 2 の場合には，図 3.10(1)-(3) のように，まず物体 A が一時待避された．これにより物体 D のゴール位置が空くので，物体 D をゴール位置まで移動する．その後，ゴール位置の空いた物体から一つずつ移動していき，タスクを達成した．ロボット数 4 の場合には，ロボットがそれぞれ物体を持ち，物体 A を一時待避した後，一つずつ物体をゴール位置に移動しタスクを達成した．

実験は 20 回繰り返し，経路が得られなかった場合や実行時間が 20 分を越える場合には失敗とした．結果を表 3.1 にまとめた．平均値は成功した試行だけで求めた．

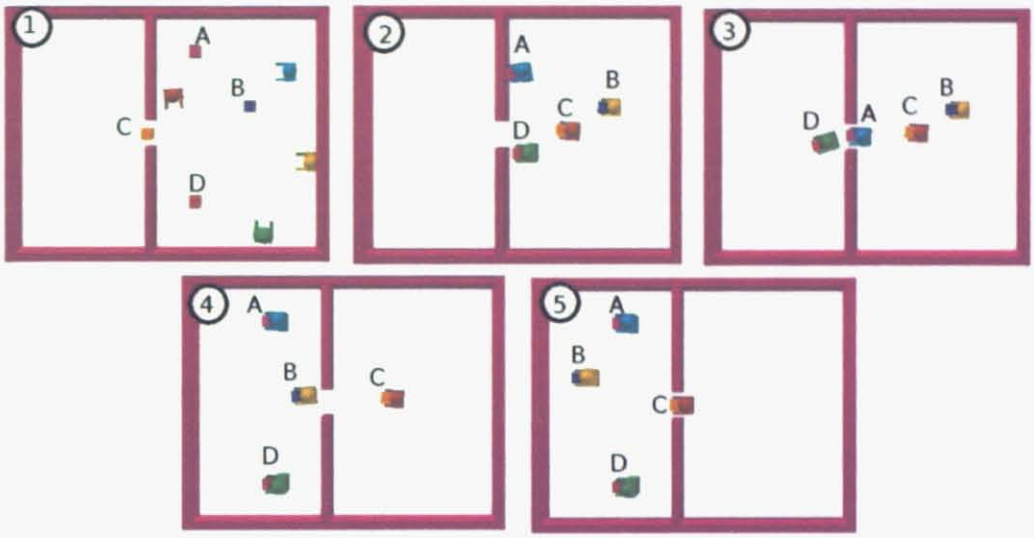


図 3.9　問題 P-4 （ロボット数 4）で得られた行動の例

図 3.10　問題 P-5 （ロボット数 2）で得られた行動の例

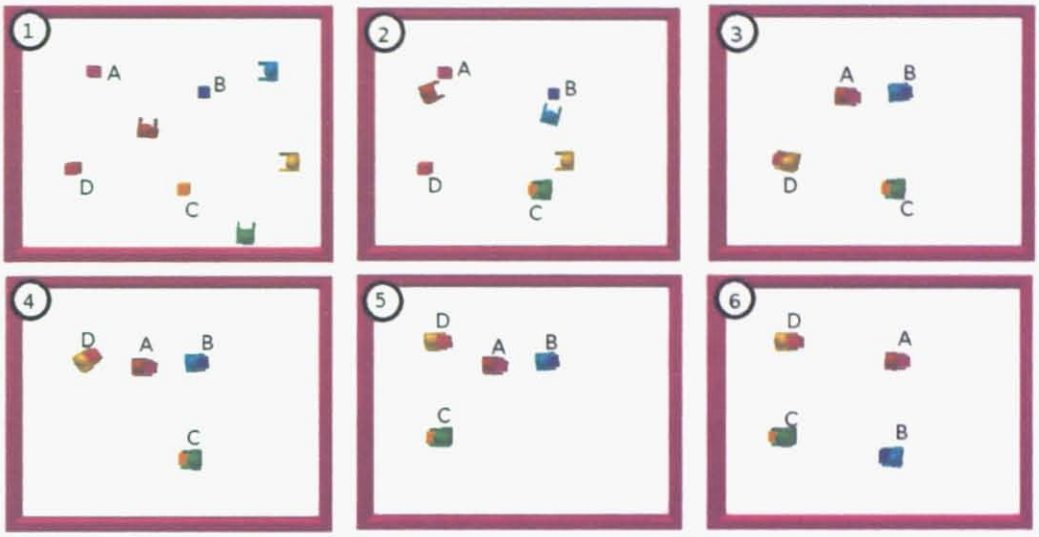


図 3.11　問題 P-5 （ロボット数 4）で得られた行動の例

表 3.1　実験結果（「プラン長の割合」はそれぞれの問題のロボット数 1 でのプラン長を基準とした割合である）

問題番号	P-2			P-4			P-5		
ロボット数	r = 1	r = 2	r = 4	r = 1	r = 2	r = 4	r = 1	r = 2	r = 4
成功率	95%	90%	90%	100%	100%	65%	95%	95%	95%
プラン長の割合	1.00	0.71	0.41	1.00	0.71	0.37	1.00	0.54	0.46
平均作業割合	100.0%	44.6%	38.9%	100.0%	55.2%	45.5%	100.0%	88.8%	43.0%
実行時間 [秒]	152.38	176.31	207.55	238.27	256.23	280.91	90.31	91.28	811.58

表中の「プラン長の割合」は，ロボット数を変化させたときの，タスク完了までの平均行動プラン長の変化を示している．環境中のロボット数を n とすると，作業の分担によって理想的にはプラン長は $\frac{1}{n}$ となりえる．しかしタスクによっては物体の移動に順序関係が存在したり，環境中に移動のボトルネックとなる通路があるため通常はそれより長い．結果によると，プラン長は $\frac{1}{n}$ まで改善はしないものの，ロボットを増やすにつれて減少していることが分かる．このことから適切なタスクの分担がなされていると言える．

表 3.1 中の「平均作業割合」とは全ロボットの行動プランの中でロボットがタスクに関わっている時間の割合を示したものである．すなわち，

$$\frac{\sum_{i=1}^{n}(\text{ロボット } i \text{ がタスクに関わっている時間})}{\sum_{i=1}^{n}(\text{ロボット } i \text{ の行動プラン長})} \tag{3.2}$$

で計算した．この値が大きいほどロボットは作業を行っている期間が長く，無駄時間が少ない．これは全ロボットでのタスク並列度と見なせる．n 台のロボットがいる環境で，ロボット 1 台だけですべての作業を行うという非効率な状況では，この値は $\frac{1}{n}$ となる．うまく作業を分担することで，これよりも大きな値となる．表 3.1 によると，得られた結果はほとんど $\frac{1}{n}$ よりも大きな値であり（ただし問題 P-2 (r=2) は小さい），タスクの並列

(a) 初期状態　　(b) ゴール状態

図 3.12　並べ替え問題

度が高く，作業の適切な分配ができていると言える．問題 P-2（r = 2）では値が小さいが，これは中央の通路がボトルネックとなるため 2 台のロボットでは待ち時間が生じ，並列度があまり上がらないと考えられる．

表中の「実行時間」ではプランの作成に要した時間を示した．ロボット数を増やしても指数関数的な計算量爆発を起こさないことが確認できる．問題 P-5（r = 4）は計算時間が特に長いが，これはタスクの並列化のためのタイミング決定（3.2.2 節）に時間がかかっていることが観察された．

3.4.2　並べ替え問題での実験結果

再配置問題のより複雑な例として，並べ替え問題での実験を行った．この問題では図 3.12(a) のように狭い通路内に並んでいる荷物を逆順に並べ替える（図 3.12(b)）．これはすべての荷物を一時待避する必要があるため，一時待避の場所が適切でないと解くことが難しい．

ロボット数を変化させながら提案手法で得られた結果を図 3.13, 3.14, 3.15 に示した．いずれも荷物の一時待避をうまく行い，タスクを達成できている．ここで，一時待避場所までの距離は前節での問題よりも長く設定した．これは物体 D を狭い通路から取り出せる程度の距離である．

複数ロボットを用いた場合には，通路内に荷物とともにロボットが取り残されてしまう．これはロボットの最終状態を与えていないためである．これを解決するにはロボットの最終状態を与えるか，ロボットが作業できるスペースを確保するように行動させる必要がある．これらの改善は本章の目的からは離れるのでここでは扱わないこととする．

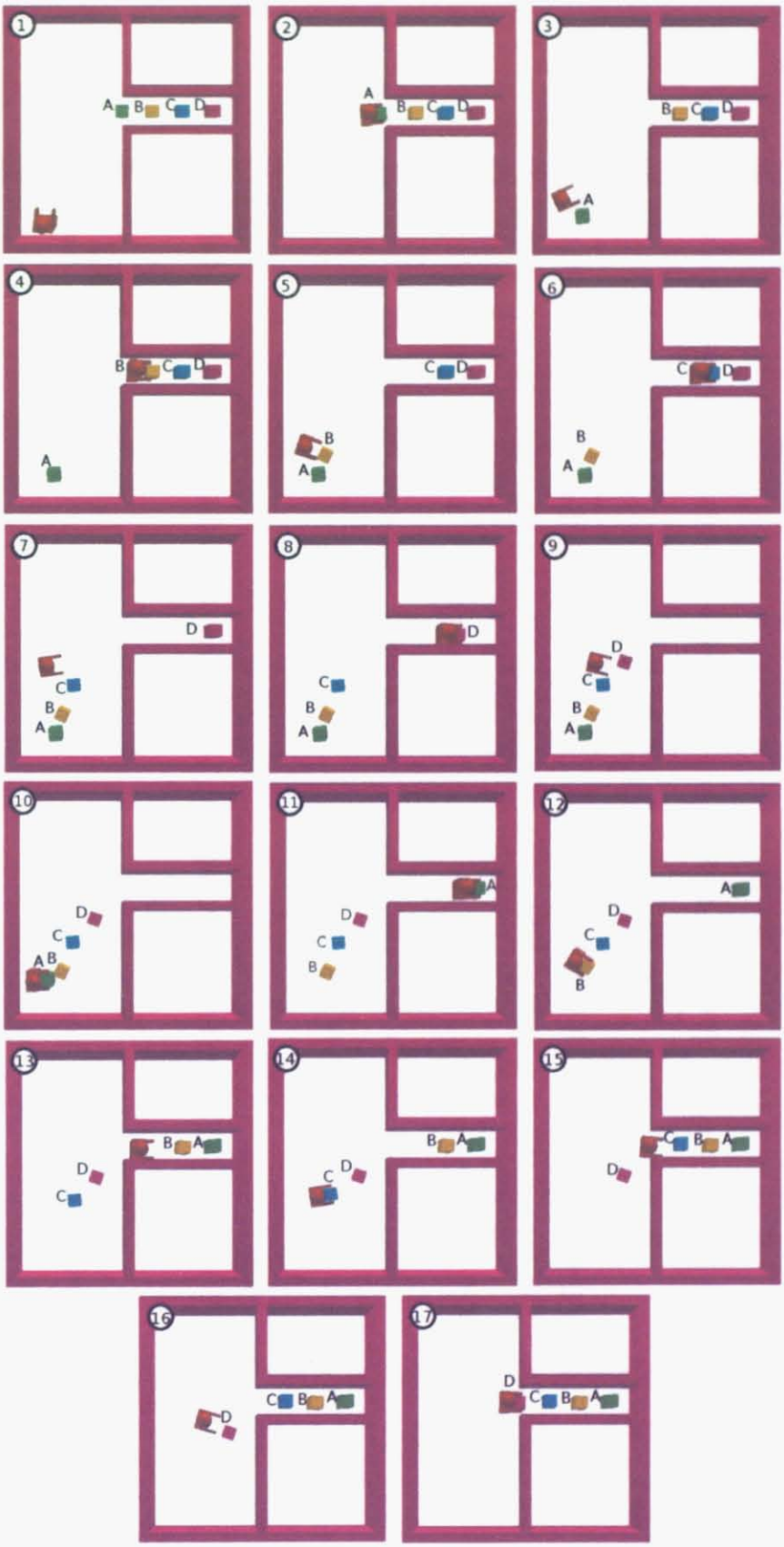


図 3.13　並べ替え問題で得られた経路の例（ロボット数 1 の場合）

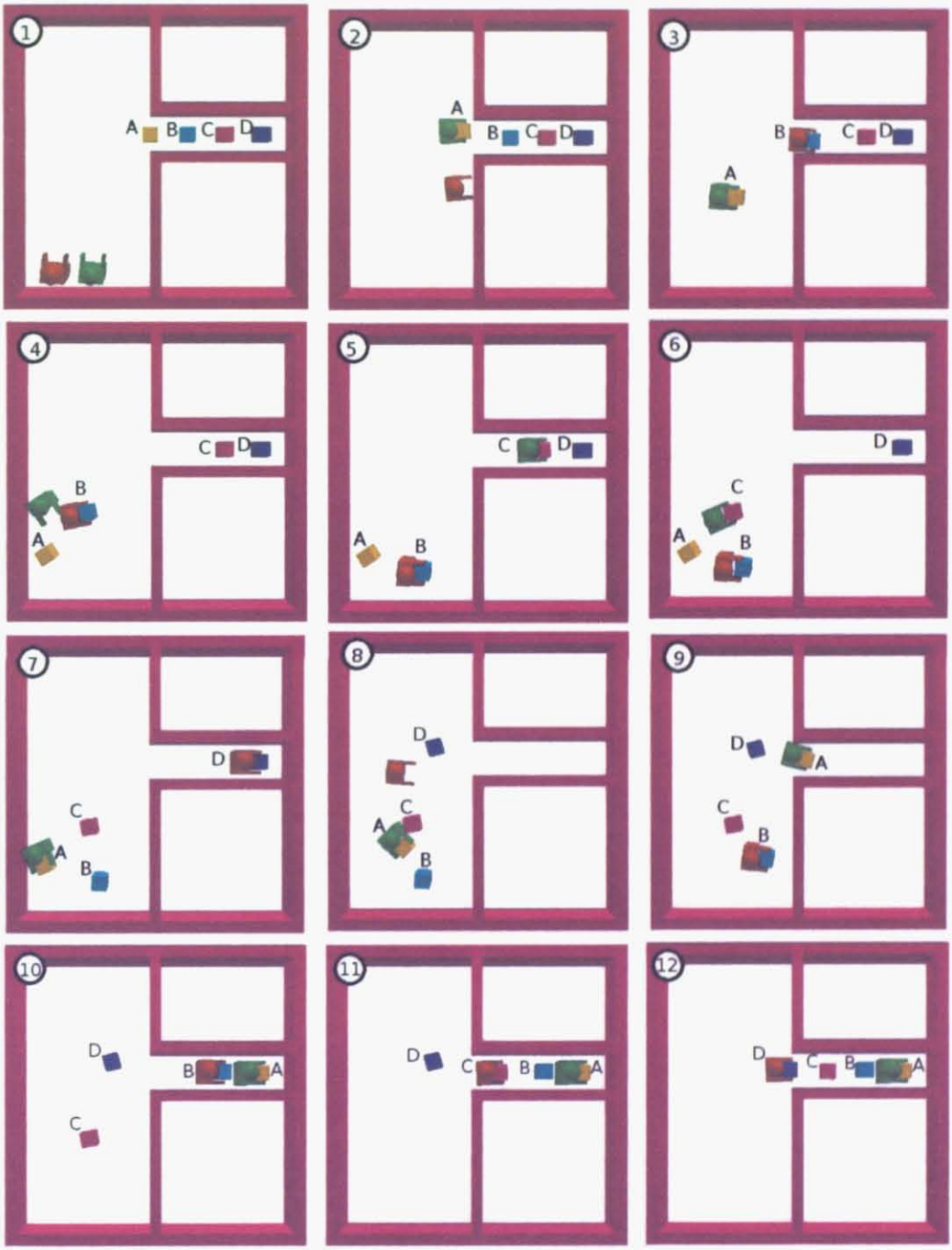


図 3.14　並べ替え問題で得られた経路の例（ロボット数 2 の場合）

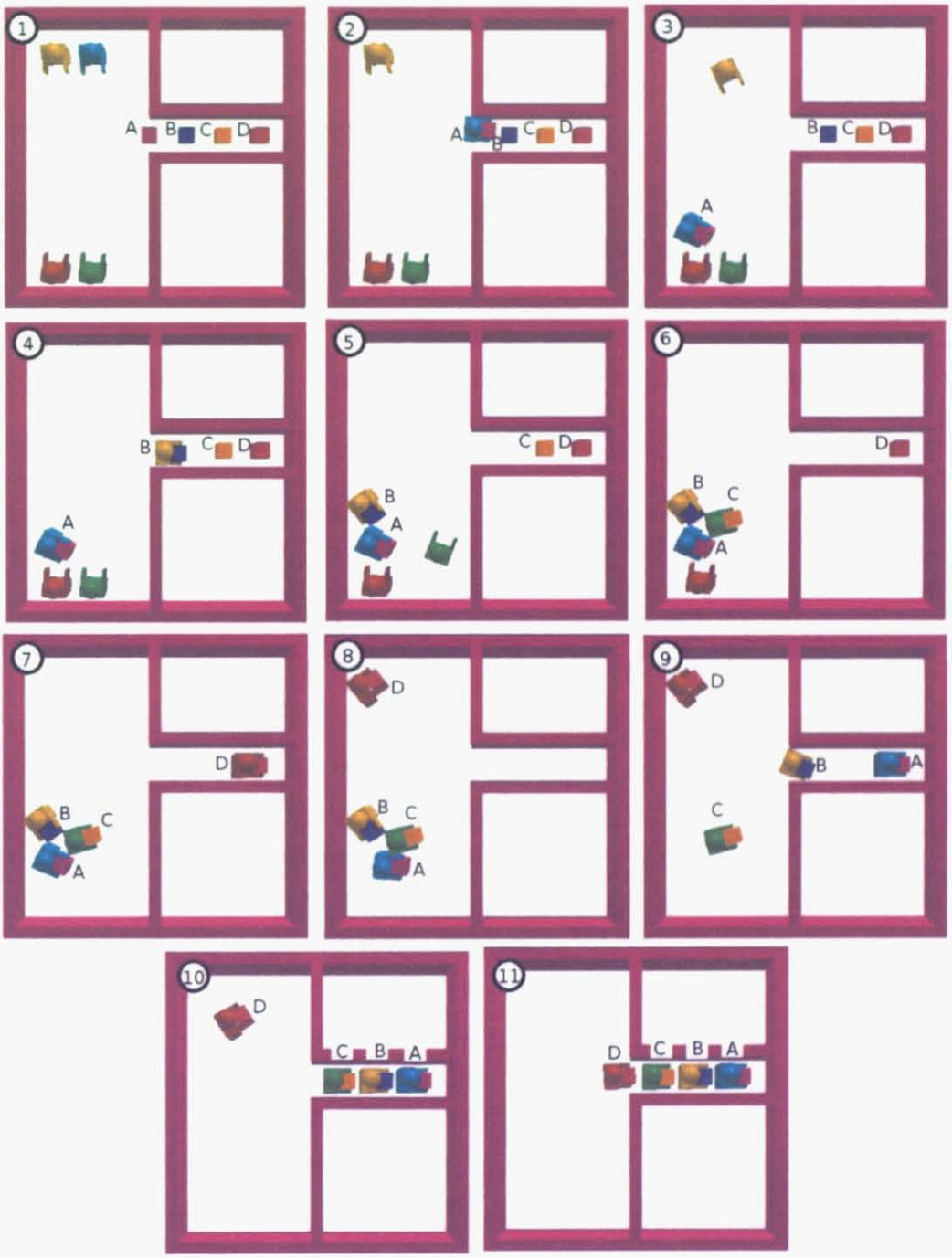


図 3.15　並べ替え問題で得られた経路の例（ロボット数 4 の場合）

3.5　ダイナミックプログラミングによる手法との比較

ここでは従来手法であるダイナミックプログラミング（DP）を用いた手法と提案手法の比較を行う．ここでの DP は離散状態空間を定義し各状態の状態価値を計算するもので，離散状態空間内での経路プランニングとしての厳密解が得られる．ただし複数ロボットを扱う際には厳密解を得ることは容易ではない．

まず状態空間を定義するために環境を離散化する．ここでは離散化のための格子として縦横 5.0cm，回転方向には 4 等分（90 度ずつ）した格子を用いた．提案手法で用いたプランニングの単位長さはこの格子より小さい 1cm であり，回転角度の刻みもより細かいが，1cm の格子で DP を行うには全状態数が膨大となるため 5.0cm 刻みとした．実験に用いた環境は前述の問題 P-2，P-4，P-5 であり，環境のサイズは 600cm×400cm×360 度（回転角度）である．ここで，単純に 4 台のロボットを考慮した状態空間の大きさを求めると，$((600/5.0) \times (400/5.0) \times 4)^4 \approx 2.17 \times 10^{18}$ となる．これは実際上扱いきれない計算量である．

そこで計算量を減らすための工夫を行った．それには以下のような 2 段階でプランニングを行う．

1. 各ロボットについて個別に状態空間で経路プランを生成する．
2. それぞれの経路プランが衝突しないように時間調整をして，全体的なプランを得る．

これにより，個別のプランニングではロボット 1 台分の状態空間を用いれば良く，計算量を抑えることができる．もちろん，ステップ 2. でプランを統合する部分での計算量はロボット数に応じて増える．しかしその計算量は，状態空間のサイズには依存せず，それぞれのロボットのプラン長に依存するため，単純な手法よりも計算量が抑えられる．この手法では必ずしも最適解が得られる保証はなく，原理的に解を得られないケースもありうるが，一般的には実用的な解が得られると考えられる．

状態空間内での経路プランニングには文献 [4] と同様の状態価値によるプランニングを用いる．ただし，経路を作る際には他のロボットが静止していると想定して経路を生成する．すべてのロボットについて経路を得た後，文献 [25] に示されたダイナミックプログラミングを行うことで，それぞれの経路の最適な組み合わせと待ち時間を探索する．ロボット同士のプランが衝突を起こす場合には，適切な待ち時間が挿入されるので衝突のない経路を得られる．

以下ではその計算原理を概説する．

3.5.1　DP に基づくプランニング

ロボットごとの経路プランニング

このプランニングは前処理とその結果を用いたプランニングの 2 段階からなる．前処理では全状態にわたって状態価値を計算する．その結果を用いて，現在位置からのプランニングを行う．

まず，ロボットの離散化された状態 (x, y, z) の価値を $V(x, y, z)$ と表現する．隣接状態の中での最小値を使って，状態価値を以下のように計算する．

$$V(x,\ y,\ z) = \min\left(\ V(x \pm 1,\ y \pm 1,\ z \pm 1)\ \right) + \mathrm{cost} \tag{3.3}$$

ここで，cost は状態の移動に要する時間である．ただし，その状態でロボットと他のロボットや環境との衝突が起こる場合には，$V(x,\ y,\ z) = \infty$ とする．これをゴール状態から隣接状態へと伝搬しながら計算させていけばよい．すべての状態について状態価値を得られたら，前処理は完了である．

プランニングをするには，初期状態の状態価値から始めて，状態価値のもっとも小さい隣接状態への遷移を行えば，ゴール状態までの最小時間の経路が得られる．

このプランニングの計算量は状態空間の状態数に依存する（正確には，状態価値をどの順序で計算するかによって計算量が変わる）．状態変数 x, y, z がそれぞれ n_x, n_y, n_z 個の状態に分割されているとすると，全状態数はそれらの積 $n_x n_y n_z$ となる．

プランの統合

前節のプランニングで得られたロボット $i\,(1 \leq i \leq n)$ のプランを時刻 $t = 0$ から $t = L_i$（タスク終了時刻）までの状態列 $\boldsymbol{x}_i = \{s_0^i,\ s_1^i,\ \ldots,\ s_{L_i}^i\}$ と表現する．するとプランの統合は，それぞれのロボットの状態列の積 $\prod \boldsymbol{x}_i$ によって張られる状態空間中での最小時間経路の探索になる．すなわち各要素はそれぞれのロボットについての状態をパラメータとする状態価値 $V_{\mathrm{multi}}(s_{j_1}^1,\ s_{j_2}^2,\ \ldots,\ s_{j_n}^n),\ (1 \leq j_i \leq L_i)$ として表現される．状態空間は図 3.16 のようになる．

まずゴール状態（すべてのロボットがゴールしている状態）の状態価値を 0 とする．すなわち，

$$V_{\mathrm{multi}}(s_{L_1}^1,\ s_{L_2}^2,\ \ldots,\ s_{L_n}^n) = 0, \tag{3.4}$$

とし，そこから初期状態（すなわち $V_{\mathrm{multi}}(s_0^1,\ s_0^2,\ \ldots,\ s_0^n)$）にむけて隣接状態の状態価値を順次計算する．時刻は一方向にしか進まないので，隣接状態としては時刻が減少する方向（つまり，s_t^i で t が減少する方向）に向けてだけ計算すれば良い．状態価値は隣接状態を用いて以下のように計算する．

$$V_{\mathrm{multi}}(s_{j_1}^1,\ s_{j_2}^2,\ \ldots,\ s_{j_n}^n) = \min\left(\text{隣接状態の状態価値}\right) + \mathrm{cost} \tag{3.5}$$

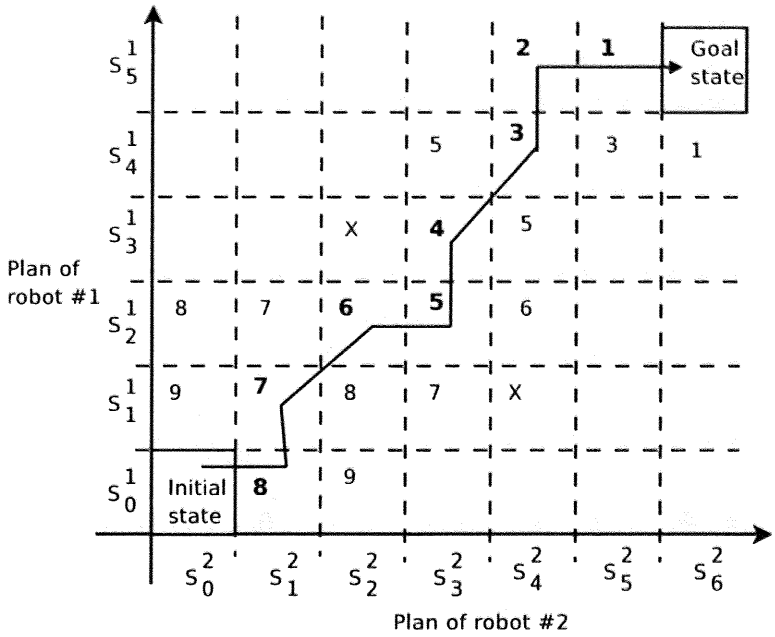


図 3.16　状態空間で得られたプランを統合する例（ロボットが 2 台の場合）．各セル中の数値は状態価値を表す．

ここで，cost は状態を移動するのに要する時間である．ただし，ロボット同士や環境との衝突が起こる場合には状態価値を ∞ とする．これを初期状態に到達するまで求めることで前処理は終了である．それぞれのロボットのプラン長を L_i とした場合，状態空間のサイズは $\prod L_i$ となる．計算量もこれにおよそ比例する（ただし計算順序により変動する）．

この後，初期状態から状態価値の減少する方向へと状態を探索することで，統合されたプランが得られる．これは適切な待ち時間を含んだプランである．探索の例を図 3.16 に示した．

3.5.2　比較実験

DP による手法で計算した結果を表 3.2 にまとめた．実験は各実行時間を最大 60 分とし，20 回繰り返した．アルゴリズム中での優先度グラフの構築と一時待避場所の決定には，条件を同じとするため提案手法と同じ RRT を用いた（一時待避場所までの経路の生成には状態空間を用いている）．表中の「プラン長の割合」「実行時間の割合」はそれぞれの提案手法での値を基準とした割合を示した．

問題 P-2，P-4 では，ロボット数が 4 のときに計算が 60 分以内に終了しなかった．そのためプラン長などのデータが得られていない．これはロボット数が増大にするにつれて，前記のプラン統合での状態空間が指数関数的に増大し，計算時間が増大したためである．この実験での環境の量子化サイズは縦横 5cm，回転方向は 4 等分（90 度ずつ）である．この実験とは別に，環境の量子化サイズを縦横 2cm，回転方向 4 等分とした場合の実行も試みたが，計算量が増大するために，どの問題も 60 分以内には計算が終了しなかった．

表 3.2　状態空間を用いた実験結果．「プラン長の割合」「実行時間の割合」はそれぞれの提案手法での値を基準とした割合である．

問題番号	P-2			P-4		
ロボット数	r = 1	r = 2	r = 4	r = 1	r = 2	r = 4
成功率	100.0%	100.0%	0.0%	95.0%	90.0%	0.0%
プラン長の割合	99.46%	95.67%	N/A	91.32%	85.19%	N/A
実行時間の割合	303.8%	396.4%	> 1734.5%	252.5%	326.5%	> 1281.5%

問題番号	P-5		
ロボット数	r = 1	r = 2	r = 4
成功率	100.0%	100.0%	95.0%
プラン長の割合	113.5%	120.2%	112.4%
実行時間の割合	460.2%	674.1%	109.9%

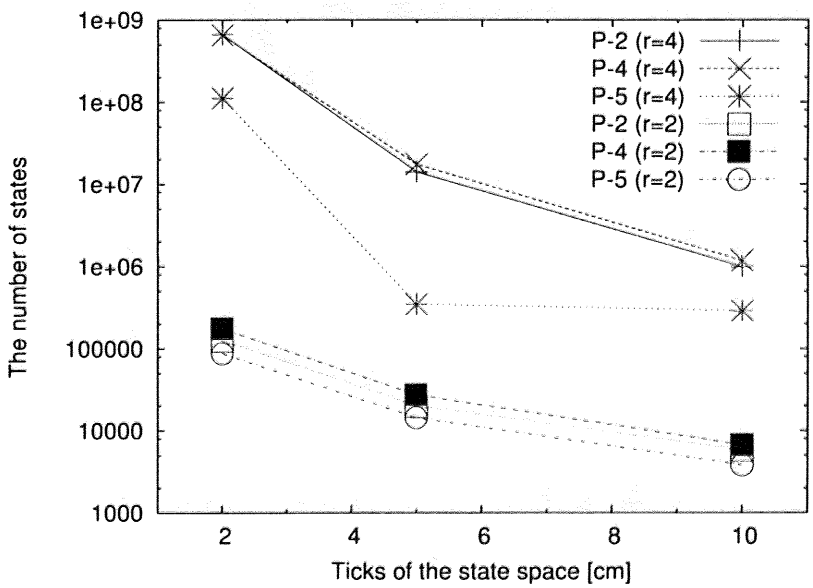


図 3.17　プランの統合に要する状態数．縦軸はログスケールである．

図 3.17 にはプランの統合に要する状態空間サイズ（状態数）を示した．横軸の環境の量子化サイズは 2cm，5cm，10cm と変化させて計測した．縦軸はログスケールである．図から分かるとおり，環境の量子化サイズが細かくなるにつれ，状態数が指数関数的に増加している．これは実用的な時間内に計算することができない．例えば，環境が 5cm 刻みでロボット数が 4 の場合には，問題 P-2，P-4 ではこのプラン統合処理だけを行うにも 60 分以内に終了しなかった．このように DP 手法は量子化サイズやロボット数によって計算量が大きく変動してしまう．提案手法はこのような問題がなく，より実用的な手法といえる．

プラン長について提案手法と比較すると（表 3.2），DP 手法によるものは最短で

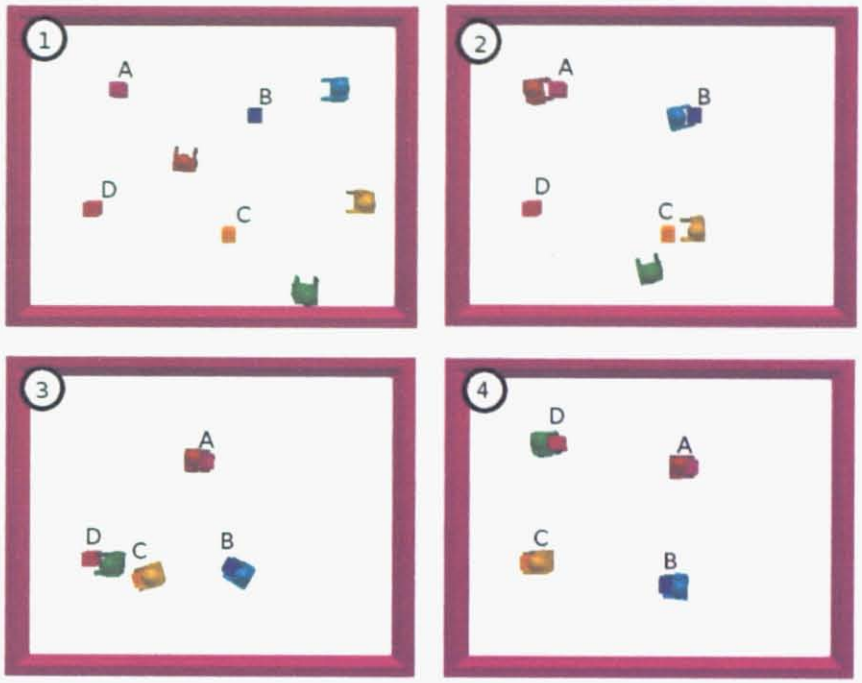


図 3.18　問題 P-5（ロボット数 4）で得られた最短行動

85.19%（問題 P-4，ロボット 2 台の場合）だが，そのほかは 90% 台であり，問題 P-5 では提案手法よりも長くなった．提案手法よりも長くなる理由としては，それぞれのプランが最適なものであっても統合した結果が最適とは限らないためと考えられる．さらに DP 手法では，それぞれの移動タスクの完了時刻がそれぞれのプランを統合するまで確定しないという問題点がある．そのため，タスクへのロボット割り当て（3.2.3 節）が適切に実行できない可能性がある．問題 P-5 でロボット 4 台の場合に全く作業を行わないロボットのいることが観察されており，これを裏付けている．全く作業を行わないロボットが存在するとプランの統合に要する計算量は減少するが，全体のプランはその分長くなる．

提案手法は DP 手法より短い時間で DP 手法と同程度の長さのプランを得られる．一方，DP 手法は多くの場合に比較的短いプランを得ることができる．しかし時間をかけたからといって必ずしも短いプランを得られない．しかもこの結果は従来手法のほうが提案手法よりもプランニングの精度は粗い．これらのことから，提案手法のほうがより短い時間で，満足できるプラン長の結果が安定的に得られる点で有利な手法と言える．

3.6　考察

3.6.1　タスク並列度と計算時間の関係

本研究では，タスクの並列度を高め経路プランをできるだけ短くするようプランニングを行った．それにより興味深い結果が得られた．

問題 P-5 は優先度グラフに基づくと一時待避が必要となる．しかしロボット数 4 の場

合には，一時待避は本来必要ない．すべてのロボットが 1 つずつ荷物を持ち同時に移動すれば，どのロボットも一時待避する必要がなく，待ち時間が生じないからである．提案アルゴリズムでも同様な結果（一時待避を行った直後に移動を続けるので実質的な待ち時間がない）を得られた（図 3.18）．この場合には平均プラン長はロボット数 1 のときの約 1/4 にまで短縮された（最小値は 24.4% であった [*2]）．ただし，提案手法は確率的な探索を利用しているため，最適な分担を常に見つけるわけではなく，ロボット数 4 の時の平均のプラン長は 46% にとどまった．

表 3.1 では問題 P-5（ロボット数 4）は特に計算時間が長いが，これはタスクの並行化のためのタイミング決定（3.2.2 節）に時間がかかっていることも明らかにした．計算時間とタスク並列化とはトレードオフの関係にある．タスク並列化を目的として探索を試みることで，前記のように待ち時間のない効率的な行動が得られる一方，タスク並列化に失敗した場合には，それを試みた計算時間は無駄となってしまう．それが結果として計算時間の増加となって現れている．

3.6.2　関連研究

Ben-Shahar ら [3] や Ota [32] はロボット 1 台で複数物体の再配置問題を解くアルゴリズムを提案している．Fukazawa らは Ota の手法を用いて，実ロボットでの実験を行った [13]．しかしこれらの行動プランニングはマルチエージェント環境では計算量が膨大となってしまう．それだけでなく，彼らが示したアルゴリズムではマルチエージェントによる再配置問題には対応できない．なぜならばロボットが複数存在するために，ロボットの行動プランをどのように割り当てるか（3.2.2 節）や，物体の搬送作業をどのロボットに割り当てるか（3.2.3 節）という点を解決する必要が生じるからである．本研究では，それらを解決する手法を示した．

Cherif らは複数のロボットで複数物体を搬送する行動プランニングを示した [9]．彼らのアルゴリズムでは環境のマップを前処理し，そこで得られた静的な経路上をロボットが移動する．そのため本研究で扱ったタスクのように時刻に応じて複雑に変化する環境に適用するのは困難である．また，彼らの手法ではロボット同士の経路が衝突した場合の扱いが特殊であり，提案手法のように衝突を回避する行動を実現できない．

Okada らは環境中の物体を移動しながらタスクを実行する行動プランナを提案している [27]．しかし彼らのアルゴリズムでは，あらかじめ決められたゴール位置までロボットが移動する際に障害となる物体を一時待避するだけであり，再配置問題とは異なる．また，彼らの行動プランナではシングルエージェント環境しか想定されていない．そのためマルチエージェント再配置問題に適用することはできない．

[*2] ロボット数 4 のときに得られた最短の平均プラン長は 24.4% であり，理想的なタスク分担の値（1/4 = 25%）よりも小さくなった．その理由は，提案手法が常に最短経路を見つけるわけでないためロボット数 1 の経路プランに無駄時間を含んでおり，そのプラン長を基準としているために理想値以上に改善できたと考えられる．

Gravot らはシンボルによるプランニングと経路プランニングを組み合わせて再配置問題を解決した [16]．彼らの経路プランニングは PRM（Probabilistic Roadmap）を用いた確率的手法 [36] である．しかし複数ロボットの並行作業は考慮しておらず，同時に一台のロボットしか作業できていなかった．本提案手法では複数のロボットに作業タスクを割り当て，作業プランを短縮できることを示した．このような並行作業は実環境への応用を考える際に重要な機能と考えられる．Gravot らが用いた PRM は同じ環境で何度も経路プランニングを行う際には有効である [8]．しかしマルチエージェント環境では他のロボットの行動によって環境が変化するため，PRM によるプランニングは適さない．また，ロボット数が増えるにしたがって探索空間も指数関数的に増大するため，あらかじめ十分なサンプルを得ることが難しい．提案手法ではマルチエージェント環境で環境が複雑に変化しても問題なくプランニングを行えた．この点からも提案手法のほうが有効と考えられる．

3.7　本章のまとめ

本章では提案する RRT 経路プランニングアルゴリズムを物体再配置問題に適用した．本研究では，全体のタスク実行時間を減らすようにロボットの並行作業を実現した．シミュレーション実験を行い，ロボットを増やすことで再配置問題がより効率的に解けることを示した．

比較実験として，ダイナミックプログラミングによる手法と提案手法との比較を行った．その結果，提案手法のほうが短い実行時間で同程度の結果が安定的に得られていることが確認された．

第 4 章

協調荷物搬送問題への適用

これまで，複数の移動ロボットの協調によって荷物を搬送する研究が行なわれている（例えば [30]）．それらの手法ではロボットのゴール位置があらかじめ与えられていた．しかし複数ロボットが協調するためのゴールやサブゴールをすべて指定するのは操作者にとって煩雑である．

本章では，第 2 章で提案した経路プランニングアルゴリズムを応用して協調荷物搬送問題を解決する．提案手法を用いることで，最終的なゴールを達成するために必要なサブゴールを少ない情報から自動的に生成することができる．また，提案手法は従来手法よりも効率的であることをシミュレーションによって示す．

4.1 協調荷物搬送問題

本章で対象とする問題は，三体のヒューマノイドロボットが協力して，荷物をスタート位置からゴール位置まで運搬するタスクである．環境の状態は既知であるとする．このタ

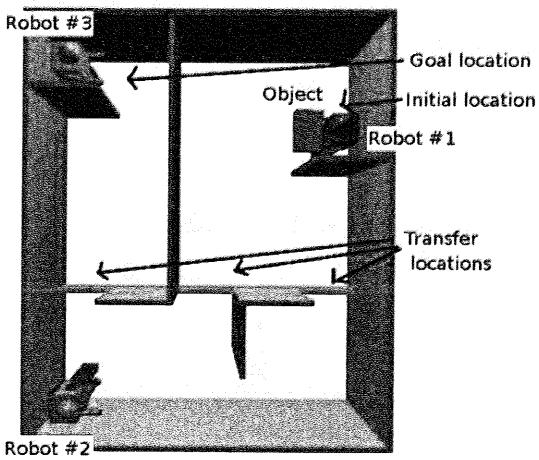


図 4.1 環境 A

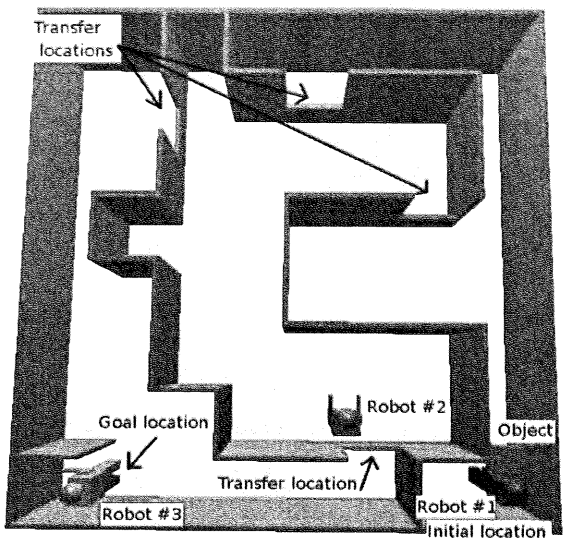


図 4.2　環境 B

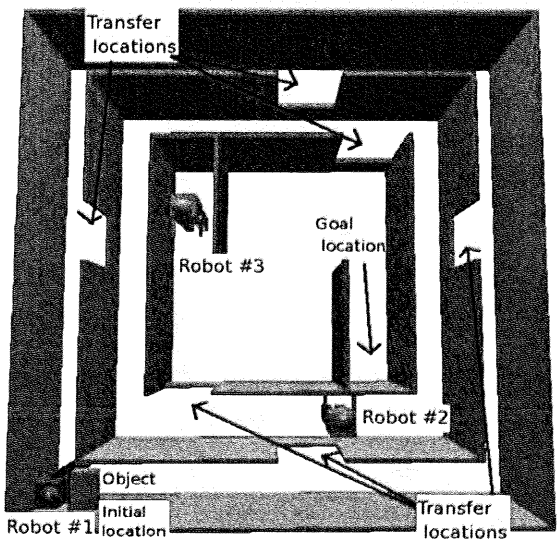


図 4.3　環境 C

スクでは荷物を受け渡しする必要があるため，ヒューマノイドロボットを対象とする．人間とヒューマノイドロボットの協調作業に関しては，経済産業省による「人間協調・共存型ロボットシステム研究開発（Humanoid Robotics Project, HRP）」[54] プロジェクトにおいて研究が行なわれた [42]．しかしヒューマノイドロボット同士の協調作業 [57] に関しての研究はほとんど行なわれていない．

使用するロボットは三体とした．ロボットのサイズは高さ 48.3cm，腕の長さは 10cm とした．これは筆者らの従来研究 [20, 57] で用いている，富士通の HOAP-1 ヒューマノイドロボット [47] のサイズを参考にした．各ロボットの状態は三次元（環境中の x 座標，y 座標，z 軸回りの回転角）の情報で表現される．

表 4.1　協調する手順

サブタスク番号	サブタスクの内容
1	ロボット#1 から ロボット#2 へ荷物を渡す
2	ロボット#2 から ロボット#3 へ荷物を渡す
3	ロボット#3 から ゴール位置へ荷物を置く

本研究では同時に二体のロボットが協調して荷物を受渡しするものとする．最初に初期位置で荷物を持っているロボットが，別のロボットに荷物を受け渡す．荷物を受け取ったロボットはまた別のロボットに荷物を受け渡す．最後のロボットはゴール地点に荷物を運ぶ．荷物の受渡しのためには，二体のロボットが協調して，互いに荷物の受渡しが可能な位置まで近付く必要がある．

本研究では以下のものが既知であるとした．

- 環境とロボットの３次元モデル
- 荷物のゴール位置
- 協調の手順

このタスクでは，ロボットはそれぞれ壁で区切られた別々の部屋にいる．これはロボットの作業範囲が限定されている工場などでのロボット群の協調行動を想定している．ロボットは壁を越えられないが，壁が低くなっているところではロボット同士が荷物を受渡し可能である（図 4.1-4.3 中の「受け渡し場所（Transfer locations)」）．ただし，ロボットはこの場所の存在を知らされるわけではなく，RRT を用いた経路プランニングにおいてロボットと環境との衝突検査が行われる中で，自動的に発見する．

提案するアルゴリズムは環境の地図によらず適用できるが，後述の実験を行なうために，3 つの環境 A（図 4.1，120 cm × 100 cm），環境 B（図 4.2，160 cm × 160 cm）と環境 C（図 4.3, 160 cm × 160 cm）を作成した．これらは，環境の対称性や行動の障害となる壁の数を変えてランダムに生成したものである．

荷物の最終的なゴール位置は与えられるが，各ロボットのゴール位置は与えられないものとした．与えられる協調手順とは，ロボットが協調するためのサブタスクを並べたものである．ここでサブタスクとは，荷物を持ったロボットが別のロボットに荷物を受け渡すまでを指す．使用した協調手順を表 4.1 に示した．

荷物の受渡しを行なう条件を，「二体のロボットが距離 25cm 以内，向き合った角度と進行方向のずれが，0.2 ラジアン（約 10 度）以内であること」とした．これがそれぞれのサブタスクのゴール条件である．以下ではこれを「サブゴール条件」と呼ぶ．

ヒューマノイドロボットを対象とするので，環境中を自由に動くことができるという仮定から，環境はホロノミック環境として扱った．プランニングの刻み幅は 1 秒とし，ロボットが 1 秒に進む距離（cm）と角度（ラジアン）はユークリッドノルムが 1 となるよ

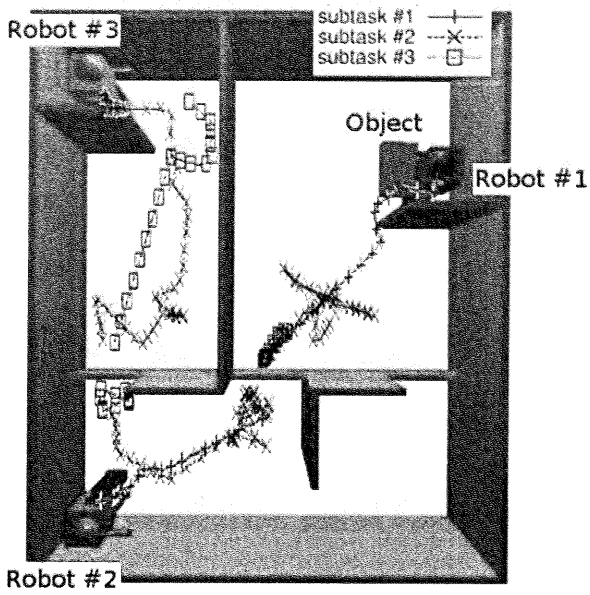


図 4.4　環境 A に従来の RRT を適用して得た経路

う制限した．これは各次元の重み付けを変えることでロボットの特性に合わせて調整可能である．

4.1.1　従来手法の問題点

従来の RRT 手法でもこのタスクのための経路を得られるが，そのためにはサブタスクごとにすべてのロボットに対してゴール位置を指定する必要がある．これは煩雑である上に，適切なゴール位置を決定すること自体も難しい問題であることが多い．さらに，サブタスクのゴールを与えても，1.5.3 節で述べたように無駄な動きが生じるため，得られる結果は望ましいものではない．従来手法で得られた経路の例を図 4.4 に示した．この図から分かるように，受け渡し位置やゴール位置に到達するまでに細かい動きが多く冗長な経路である．さらに，各サブタスクで受け渡しに関係するロボットだけでなく，すべてのロボットが動いている．これらは無駄な動きである．以下で述べる提案手法ではこのような無駄な動きを生じにくい．

4.2　提案手法

本提案手法を用いることによる特徴は以下の 3 点である:

- ロボットごとに順にプランニングを行うため，タスクに関係しない無駄な動きが少ない．
- 各ロボットの最終時刻，ゴール状態を定める必要がない．

- サブゴールまでのサブプランのプランニングにおいて，最終時刻を定める必要のない双方向探索を実現した．これによって適用範囲が広がり，探索効率が向上した．

ここで，ロボットが現状態からサブゴールに達するまでの経路プランを「サブプラン」と呼ぶことにする．

これら特徴は第 2 章の提案アルゴリズムを適用することで得られる．1 つ目の特徴は 2.1.2 節で述べた方針による特徴である．2.2.3 節のアルゴリズムでサブタスクに必要なサブゴールを自動的に生成することで，2 つ目の特徴が実現される．3 つ目の特徴は 2.2.1 節で提案した双方向探索を利用することで実現される．

以下では，協調荷物搬送問題を解決するアルゴリズムの流れを説明する．

4.2.1　サブタスクを扱うアルゴリズム

協調荷物搬送問題のプランニングは以下のようなサブタスクごとの再帰的な手続き (plan_ith_subtask ルーチンと呼ぶ) で実現される:

1. 協調手順に基づき i 番目 $(1 \leq i \leq N)$ のサブタスクで協調する二体のロボットを選ぶ．二体のそれぞれにサブゴール（状態）を生成する（GenerateSubgoal ルーチン）．
2. 荷物を持ったロボットについて，現在状態からサブゴール状態までのサブプランを生成する（plan_subplan ルーチン）．
3. 荷物を持ったロボットと協調する（荷物を受け取る）ロボットについて，現在状態からサブゴール状態までのサブプランを生成する（plan_subplan ルーチン）．
4. 二つのサブプランの最終時刻をそろえる．
5. $i+1$ 番目のサブタスクに関して，再帰的にステップ 1. から実行する．これ以上サブタスクがなければ終了．

サブゴールとサブプランの生成には，RRT に基づいたアルゴリズム（4.2.2 節, 4.2.3 節）を用いる．RRT は確率的なアルゴリズムであるため，サブゴールまたはサブプランの生成に（可能であったとしても）失敗することがある．そこで，失敗した場合にはバックトラックを行ないサブゴールの生成をやり直す．i 番目のサブゴールもしくはサブプランの生成に一定回数（MAX_RETRY パラメータ）失敗した場合には，バックトラックして $i-1$ 番目のサブゴールの生成からやり直す．

提案するアルゴリズムでは各ロボットで個別にプランニングを行なう．その際，プランニングを行う優先順序を決め，デッドロックを防いでいる．上記のようにサブタスクごとに，荷物を持ったロボットを優先的にプランニングし（ステップ 2.），それと協調するロボットをその次にプランニングしている（ステップ 3.）．このとき，先にプランニングされたロボットと衝突しないようにプランを生成する．このサブタスクに関係しないロボットは，これ以降のサブタスクで必要に応じてプランが作られるので，ここではプランを作

る必要はない．すべてのサブタスクが完了したときにプランの最終時刻をそろえる操作を行う（4.2.4 節参照）．

この提案手法は，環境が変化する場合にも，各時刻の環境の状態が既知であれば適用できる．それには，プランニング中に実行されるロボットと環境との衝突検査において，その時刻に対応した環境モデルを用いれば良い．ただし，次節で述べるサブゴールの生成では，探索効率向上のために時刻を考慮していないため，適切なサブゴールが生成できない可能性がある．そのため，環境が変化する場合には必ずしもプランを生成できるわけではない．

4.2.2　サブゴールの生成

サブゴールの生成には，上記の問題設定のうち，(1) 協調手順で指定された二体のロボット（r_1, r_2 とする）が荷物を受け渡しすること，(2) 荷物の受け渡しのためロボット同士の距離と向き合う角度に関して一定の条件（サブゴール条件，4.1 節参照）を満たすこと，を利用する．このサブゴールの生成には 2.2.3 節で示したサブゴール生成アルゴリズム GenerateSubgoal が利用できる．

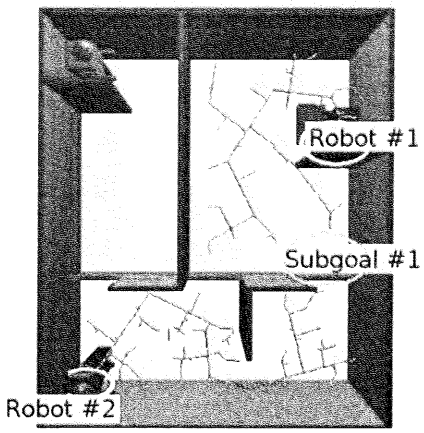
サブゴール生成アルゴリズムでは，2 つの RRT が協調する二体のロボットの現在の状態からそれぞれ成長する．そして，互いに「サブゴール条件を満たす標準状態 $\boldsymbol{x}_{\mathrm{coop}}$」に向けて探索木を伸ばす．ここでの標準状態は，ロボット同士の距離 20cm，向き合った角度のずれがゼロとした．これは最も安定して荷物受け渡しを行える状態である．

タスクに応じたサブゴールを生成するために，2.2.3 節に示したアルゴリズム中の r_1 は荷物を持ったロボットとし，r_2 は荷物を持たないロボットとする．そのため，探索木 T_1 を構成するには，荷物を持ったロボットモデルで環境との衝突判定を行い，探索木 T_2 を構成するには，荷物を持たないロボットモデルを用いる．

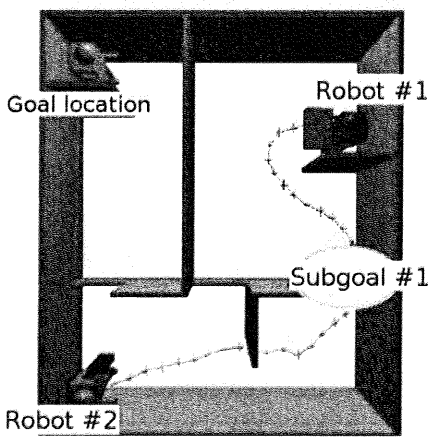
例として図 4.5(a) に，環境 A でのサブタスク#1 のためにサブゴールを生成した例を示した．ロボット#1 とロボット#2 の現在状態から探索木を成長させることで，サブゴール条件を満たす状態ペアを見つけた．図中の “subgoal #1’ を挟む上下両側がサブゴール条件を満たす状態ペアである．

このサブゴール生成では状態に時刻を含めずにプランニングしているため，他のロボットが存在する場合には必ずしも実行可能な経路ではない．そこで提案手法では，サブゴールを生成した後に，時刻を含めたサブプランを改めて生成する（次節参照）ことで，プランの正しさを保証している．

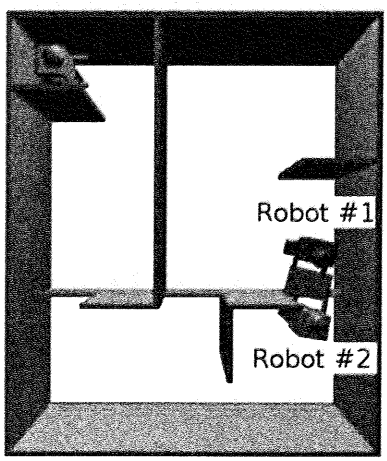
バックトラックが発生する可能性があるので，探索に用いた探索木を保存しておき再利用することで，サブゴールの探索を高速化する．



(a) サブタスク 1 のためのサブゴールを生成した探索木の一例



(b) サブゴールに向けて生成されたサブプラン



(c) 生成されたサブプランでサブタスク#1 を完了した状態

図 4.5　環境 A でのサブタスク#1 のためのプランニング例

4.2.3　サブプランの生成

サブゴールが生成された後，plan_subplan ルーチンでロボットの現状態からサブゴール状態に至る経路（サブプラン）を生成する．これはロボットごとに個別に行う．このときは，時刻を考慮に入れ，他のロボットが移動していても衝突しないようプランニングする．

plan_ith_subtask（第 4.2.1 節）で示しているように，協調する二体のロボットのうち，まず荷物を持っているロボット（r_1 とする）をサブゴールまでプランニングする．それと協調するロボット（r_2 とする）はロボット r_1 と衝突しないようにサブゴールまでプランニングする．ロボット r_2 のプランニングでは，ロボット r_1 がサブゴール状態に達した時刻を最終時刻（t_{end}）として，少なくとも時刻が t_{end} まではプランニングを行なう．時刻が t_{end} 以降では，ロボット r_1 はサブゴール状態に静止しているとしてプランニングする．

すなわち，サブプラン生成アルゴリズム（plan_subplan）は以下のようになる：

1. ロボットが荷物を持っている場合，
   (a) サブゴール状態にいて，かつ衝突のない状態であるとき，終了．
   (b) それ以外なら，サブゴール状態に向けたプランニングを行なう（2.2.1 節の PlanTowardSubgoal ルーチン）．
2. ロボットが荷物を持っていない場合，
   (a) サブゴール状態，かつ衝突のない状態であり，なおかつ最終時刻（t_{end}）まで達していないなら，その状態に静止したまま時刻を進める．
   (b) サブゴール状態，かつ衝突のない状態であり，最終時刻（t_{end}）まで達したなら終了．
   (c) それ以外なら，サブゴール状態に向けたプランニングを行なう（2.2.1 節の PlanTowardSubgoal ルーチン）．

これをループで実行することで，サブゴールまでのサブプランを生成できる．

図 4.5(a) で生成されたサブゴールに向けてサブプランを生成すると，例えば図 4.5(b) のプランが生成される．このサブプランを実行することで，荷物の受け渡しが図 4.5(c) のように実現される．

4.2.4　プランの最終時刻の調整

すべてのサブタスクに関してプランが生成された後，すべてのロボットのプランの最終時刻を最長のプランにそろえる．このためにそれぞれのロボットのプランを衝突のない状態で埋める．ロボットにはゴール状態が指定されないので，ゴール状態なしで探索を行わなければならない．これは 2.2.2 節で述べた PlanWithoutSubgoal ルーチンを適用することで実現できる．

表 4.2　提案手法と従来手法（RRT-ConCon）との実行時間の比較結果．括弧内のパーセント値は従来手法を基準とした値である．

	実行時間（秒）	
	提案手法	従来手法
環境 A	13.08 (254.2%)	5.15 (100.0%)
環境 B	41.52 (13.1%)	315.92 (100.0%)
環境 C	95.29 (17.9%)	533.24 (100.0%)

4.3　実験

4.3.1　従来手法との実行時間の比較

提案手法のオーバーヘッドを見積もるため，従来の RRT 手法を用いた場合との実行時間の比較実験を行った．実験には図 4.1，4.2，4.3 に示した 3 種類の環境と表 4.1 に示した協調手順を用いた．実験に用いた計算機は Linux オペレーティングシステム，Athlon XP 3200+（2.2GHz）である．各サブプランの再試行回数（4.2.1 節での MAX_RETRY パラメータ）は 5 回までとした．

比較結果を表 4.2 に示した．測定は 30 回行ない，表には平均値を示した．提案手法によって生成されたプランニング結果の一例を図 4.6, 4.7, 4.8 にそれぞれ示した．図 4.6 は，同じ環境に従来の RRT を適用した図 4.4 に比べて無駄な動きが少ないことが確認できる．

従来の RRT 手法（RRT-ConCon, 1.5.2 節）を適用するには，サブタスクごとにサブゴールを明示的に与える必要がある．そこで，実行のたびにサブタスクごとのサブゴール位置を「受け渡し場所」の中からランダムに選択して設定した．サブタスクに関係しないロボットについては，同じ場所にとどまり続けるようにした．それぞれのサブタスクのプランニングは 10 分で打ち切った．

実験結果によると，環境 A では提案手法の方が従来手法よりも 2 倍程度の実行時間を必要とした．しかし，環境 B,C においては提案手法の実行時間は従来手法よりも短くなった．環境 B では従来手法のほぼ 13% の時間でプランニングが終了し，環境 C でも約 18% の時間で終了した．

4.3.2　衝突回避実験

前述のタスクではロボットが部屋ごとにわかれて存在するため，ロボット同士で衝突を起こす可能性があるのは荷物を受け渡しする時だけである．しかしより一般的な環境であっても，提案手法は衝突を起こさない経路プランを生成することができる．

このことを示すため，ロボット同士の衝突が起こりやすい環境下での実験を行った．こ

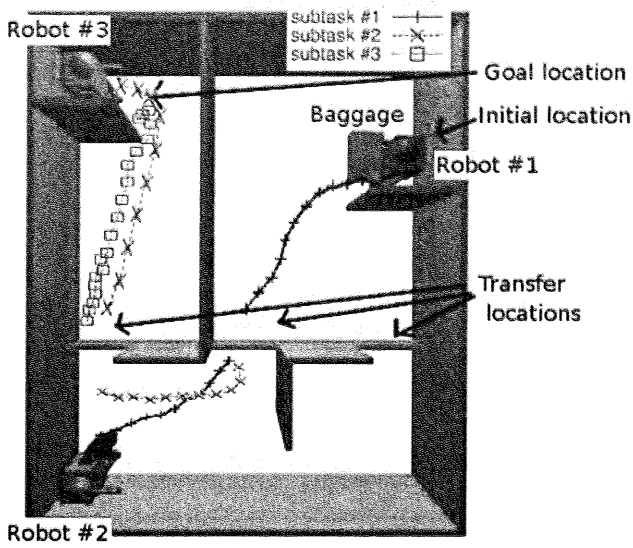


図 4.6　環境 A で生成されたプランの一例

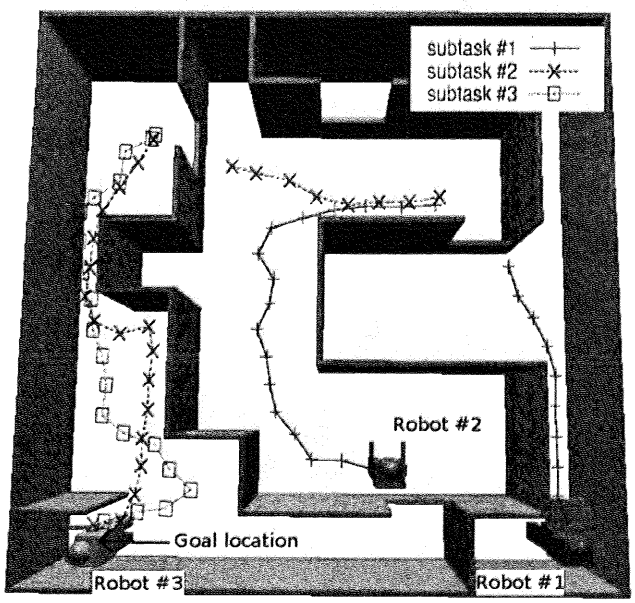


図 4.7　環境 B で生成されたプランの一例

の実験では三体のロボットが同じ部屋で行動する．問題をより難しくするために，ロボットの移動を妨げる 4 本の棒を配置した（図 4.9 参照）．初期状態ではロボット#1 が荷物を持っている．この環境での協調手順はロボット#1 がロボット#2 に荷物を受け渡すというサブタスク 1 つだけとし，荷物のゴール位置は定めなかった．ロボット#3 はロボット#1 とロボット#2 に挟まれた位置を初期位置とした．

得られた経路プランを図 4.9 に示した．提案手法により，サブゴールがロボット#3 の

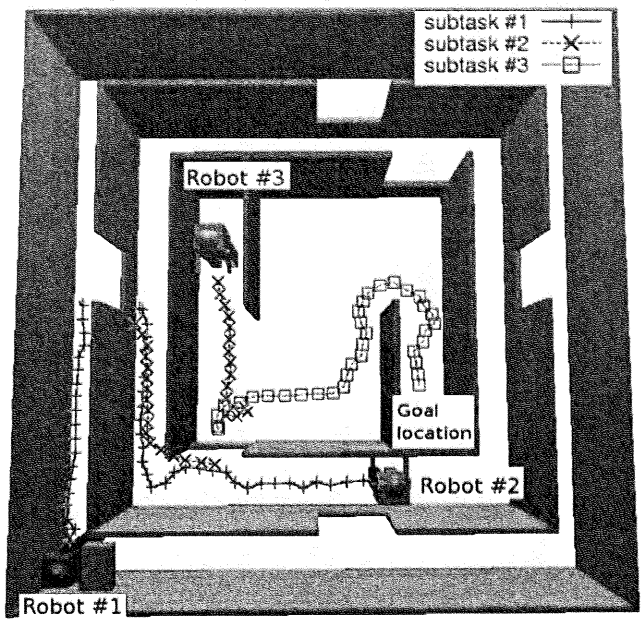


図 4.8　環境 C で生成されたプランの一例

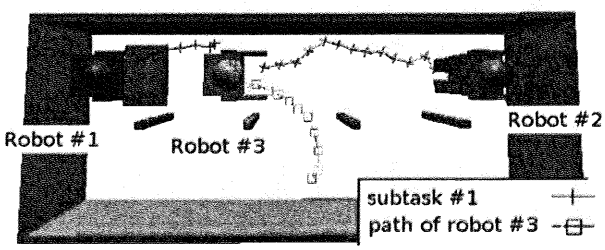


図 4.9　同一の部屋に三体のロボットがいる環境でのプランの一例

初期位置周辺に生成された．そして，ロボット#1 とロボット#2 はそのサブゴールに向けて移動する経路プランを得た．その際，ロボット#3 はタスクの邪魔となるので，衝突を起こさない位置まで移動する経路プランが自動的に生成された．このことから，提案手法はロボット同士の衝突が起こらない経路プランニングを実現することが確認された．

4.4　考察

4.4.1　提案手法の特性

4.3.1 節では，従来の RRT（RRT-ConCon）との実行時間を比較した．環境 A での実行時間は従来手法の 2 倍程度だった．しかし環境 B, C では提案手法の方が実行時間が短くなり，従来手法に比べ 80% 以上短縮された．

環境 B，C で両手法の差がこれほど広がった原因としては，2 つの要因が考えられる．まず，環境 A に比べて環境 B，C のほうが探索空間が広く，込み入っているため，プラ

ンニング問題として難しいことが考えられる．特に RRT では障害物に向けて成長しやすいという特徴があるため [2]，壁が多く込み入った環境での探索は，そうでない環境に比べて難しい．次に，従来の RRT ではすべてのロボットに対してサブゴールを与えているため，タスクに関係しないロボットの経路探索までもが行なわれている．このため余計な実行時間が消費されている．これらの結果から考えて，提案手法の実行時間は従来手法よりも長いことがあるが，難しい問題であるほどその実行時間は従来手法より短くなると結論できる．

また，いずれの環境も荷物の受け渡し場所が複数存在するので，提案手法を実行した際にどの場所がサブゴールに選ばれるかは，アルゴリズムの実行ごとに変わることがあった．ただし，協調場所の選択されやすさは RRT 探索の進みやすさに依存している．そのため，ロボットの現在位置から近い場所が選ばれることが多かった．これは移動距離を短縮することになり，プランニングアルゴリズムとして望ましい性質である．

4.3.2 節では，複数のロボットが同一の部屋にいる場合にも，提案手法を用いることでロボット同士が衝突せずにタスクを実行する経路プランを生成できることを示した．

4.4.2　関連研究

太田らは移動ロボット群の動作計画を扱った [50]．彼らは複数のロボットが群としてまとまるような経路計画を行うためにヒューリスティクス関数を用いた．提案手法ではそのようなヒューリスティクス関数を用いることなく，複数ロボットの経路プランニングを実現した．

深澤らは中間ゴールを自律的に生成する手法を提案した [58]．しかし彼らの方法では中間ゴールを得るために，初期状態からゴール状態までの行動列データを必要とする．そのため彼らは試行錯誤的に得た行動系列から中間ゴールを生成した．本研究のタスクでは複数のロボットが関与するため，中間ゴールを生成するために彼らの手法を適用することは難しい．

Curtiss [11] は，優先度をつけた RRT を用いて複数エージェントのプランニングを実現した．しかし彼らは協調タスクを目的としておらず，すべてのロボットにゴール状態をあらかじめ与えることを想定している．さらに彼らが探索に用いたのは通常の RRT 手法であり，本章でのようなマルチエージェント環境で必要となる，時刻パラメータを含めた探索には向かない．

4.5　本章のまとめ

本章では，第 2 章で提案した経路プランニングアルゴリズムを協調荷物搬送問題に適用する手法を提案した．提案手法ではサブゴールを自動的に生成するため，従来手法に比べて必要とする情報が少なく扱いやすい．シミュレーション実験を行ない，提案手法は従来手法よりも速く探索が行えることを示した．また，短い経路が選ばれやすいというプラン

ニングアルゴリズムとして望ましい性質を持つことも確認した.