多目標状況下における経路探索手法に関する研究

著者	松田　健護
学位授与機関	Tohoku University
URL	http://hdl.handle.net/10097/49869

修士学位論文

多目標状況下における経路探索手法に関する研究

東北大学 大学院情報科学研究科

システム情報科学専攻 篠原研究室

博士課程前期二年の課程

松田　健護

2011年2月15日

目次

第1章

序論

1.1 背景

現実世界において，多くの場合探索者は類似した経路探索問題を繰り返し解かなければならない．例えば，火星の探索ロボットを考える．この場合，ロボットは周囲の地形の情報を基に現在地点から目標物までの経路を求めるが，大抵の場合は1回だけ目標物に到達して終わるのではなく，同じ地形に対して別の目標物までの経路を求める必要がある．以降，探索者のことをエージェント，目標物のことをターゲットと呼ぶ．Koenig と Likhachev によって 2005 年に提案された Adaptive A* [7] は漸進的ヒューリスティック探索の1つであり，過去の探索で得られた情報を用いてヒューリスティック関数 (h) を更新することで，そのような経路探索問題を A*より高速に解くことができる．彼らは，Adaptive A*の漸進的手続きがヒューリスティック関数の無矛盾性を保つことを理論的に示している [8]．Adaptive A*は他の競合アルゴリズム [6, 10, 14] より簡単であり拡張性も高いため，様々な問題に特化した応用手法が提案されている [9, 11, 15, 16].

本論文では，多目標状況下における経路探索に Adaptive A*を適用させた手法を提案する．そこで， (1) 目標のうちどれか1つに到達すればよい， (2) 全ての目標に到達しなければならない，という2つの問題設定を考える．以降，前者を *OR* **設定**，後者を *AND* **設定**と呼ぶ．実アプリケーションの例として，ビデオゲームのパックマンを考える [17]．図 1.1 にパックマンの初期状態を示す．点数を上げるための良い戦略の1つが，“パワーエサ” を食べることでパックマンを一定時間だけ強化し，その状態

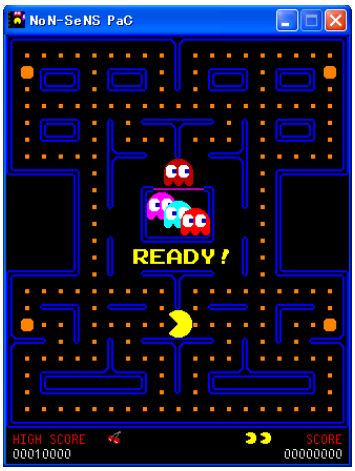

図 1.1: パックマンの初期状態

で “ゴースト (またはモンスター)” を倒して点数を稼ぐというものである．パワーエサはゲーム中に 4 つ登場するが，そのうち 1 つを食べればパックマンを強化することができる．また，ゴーストは 4 匹登場するが，1 回の強化時間中に倒した数が多いほど得点が上昇するため，全てのゴーストに到達しなければならない．そのため，最も近いパワーエサまでの経路探索問題が 4 つのターゲットのうち 1 つに到達する OR 設定に，全てのゴーストを食べるまでの経路探索問題が全てに到達する AND 設定にそれぞれ対応する．

多目標状況下において，粒子群最適化 [3] や進化的アルゴリズム [18]，ニューラルネットワーク [1] など最適性を保証しないヒューリスティック探索の研究は盛んに行われているが，その一方で最適な探索に関する研究は少ない [2, 4]．本研究は，Adaptive A*を多目標状況下での最適な経路探索に適用した最初の手法である．

第2章

準備

2.1 記号

本論文で用いる記号を以下にまとめる．集合 S の大きさを $|S|$ と表す．同様に，配列（タプル）$p \in S^*$ の長さを $|p|$ と表す．配列 p の i 番目の要素を $p[i]$ と表す．自然数の集合を $\mathbf{N}$ とする．任意の整数 $i, j \in \mathbf{N}(i \leq j)$ に対し，$[i, j] := \{\, i, i+1, \cdots, j \,\}$ と定義する．$[1, n]$ の全ての順列の集合を Π_n とする．実数の集合を $\mathbf{R}$，正の実数の集合を $\mathbf{R}^+$ とする．

重み付きグラフを $G := (V, E, C)$ と表す．ここで，V はノード集合，$E \subseteq V^2$ はエッジ集合，そして $C : E \to \mathbf{R}^+$ は正の重みを各エッジに付与する加重関数をそれぞれ表している．$e = (u, v) \in E$ が与えられたとき，$C(e)$ を $C(u, v)$ とも表記する．エッジ配列 $(e_1, \cdots, e_\ell) \in E^*$ において，$e_i = (v_i, v_i')$，$v_1 = \sigma$，$v_\ell' = \gamma$ とする．この時任意の $i \in [1, \ell - 1]$ に対して $v_i' = v_{i+1}$ であるならば，この配列を $\sigma \in V$ から $\gamma \in V$ までの経路と呼ぶ．σ から γ までの全ての経路の集合を，$P(\sigma, \gamma) \subseteq E^*$ と表記する．$p \in P(\sigma, \gamma)$ である経路 p に対し，p に含まれるエッジの総コスト，つまり経路 p の長さを次のように定義する．

$$C(p) := \sum_{i \in [1, |p|]} C(p[i])$$

$P(v_1, \cdots, v_n) := P(v_1, v_2) \times \cdots \times P(v_{n-1}, v_n)$ と定義する．

長さ n のバイナリの集合を $B_n := \{0, 1\}^n$ と表す．$b \in B_n$ である任意のバイナリ b に対し，$r_i(b) := b[1] \cdots b[i-1] 1 b[i+1] \cdots b[n]$ と定義する．

2.2 経路探索問題

以下のように形式化する．

定義 1 (経路探索問題)．経路探索問題は次の 3 項組で与えられる．

$$\mathcal{P} := (G, \sigma, \Gamma).$$

ここで，

- $G := (V, E, C)$ は重み付きグラフである．ノード $v \in V$ は，エージェント及びターゲットが留まることのできる位置を意味する．エッジ $e = (u, v) \in E$ は，エージェント及びターゲットがノード u からノード v まで移動可能であることを意味する．そしてエッジ e の重み $C(e)$ は，その移動に必要なコストを意味する．
- $\sigma \in V$ はエージェントが探索を開始する初期ノードである．
- $\Gamma \subseteq V$ はターゲットが位置する目標ノードの集合である．目標は区別可能なため，i 番目の目標に対し γ_i でアクセスできる．特に断りがなければ，目標の数として n を用いる．すなわち，$n := |\Gamma|$ である．

経路探索関連の論文では，経路探索問題におけるグラフは**状態空間**とも呼ばれる．簡略化のため，経路探索問題 $\mathcal{P}$ が与えられた場合，$\mathcal{P} = (G, \sigma, \Gamma)$，$G = (V, E, C)$ を仮定した上で，G，σ，Γ，V，E，そして C の各記号を，説明せずに用いる．もし $\mathcal{P}$ に追加表記がある場合，$\mathcal{P}$ の全ての要素に派生して表記される．例えば，$\mathcal{P}' = (G', \sigma', \Gamma')$ となる．

通常の経路探索問題の場合，表記上は多目標を用いているにも関わらず，ほとんどの研究者は単一目標の経路探索にのみ焦点を絞っている．単一目標の経路探索では，エージェントがターゲットに最小コストで到達することを目的としている．本論文では目標が 1 つである場合の解を**単一解**と定義し，下記のように形式化する．

定義 2 (単一解)．$|\Gamma| = 1$ である経路探索問題 $\mathcal{P}$ が与えられたとき，$\mathcal{P}$ 上の $\gamma \in \Gamma$ に対する単一解 p^*_γ を，初期ノード σ から目標ノード γ までの最短経路と定義する．すなわち，

$$p^*_\gamma := argmin_{p \in P(\sigma,\gamma)} C(p)$$

と定義する．

本論文では，エージェントは全てのターゲットを同時に考慮しなければならないという場合を考える．そこで，OR と AND という 2 つの設定を考える．それぞれの設定については，第 4 章及び第 5 章で定義を与える．

2.3 ヒューリスティック

経路探索問題 $\mathcal{P}$ が与えられたとき，**最適コスト関数** $H^* : V^2 \to \mathbf{R}^+ \cup \{0\}$ を定義する．ここで，$H^*(u, v)$ はノード $u \in V$ からノード $v \in V$ までの最短経路の長さである．すなわち，

$$H^*(u, v) := \min_{p \in P(u,v)} C(p).$$

エージェントは任意の 2 つのノード $u, v \in V$ に対し，$H^*(u, v)$ の見積もりである**ヒューリスティック関数** $H(u, v)$ を持つ．ヒューリスティック関数としてよく用いられるものには，マンハッタン距離や直線距離などがある．

ここで，ヒューリスティック関数について以下の 2 つの性質を定義する．**許容性**は，ヒューリスティック関数が決して実際のコストである最適コスト関数を超えないことを示している．

定義 3 (許容性)．$\mathcal{P}$ を経路探索問題，H をヒューリスティック関数とする．$\mathcal{P}$ 上のノード $v \in V$ が与えられたとき，任意のノード $u \in V$ に対して次の条件を満たすならば，H は $\mathcal{P}$ において v に対して許容的であるという．

$$H(u, v) \leq H^*(u, v).$$

また，H が $\mathcal{P}$ において任意のノード $v \in V$ に対して許容的であるならば，H は $\mathcal{P}$ において許容的であるという．

無矛盾性は，ヒューリスティック関数が三角不等式を満たすことを示している．

定義 4 (無矛盾性)．$\mathcal{P}$ を経路探索問題，H をヒューリスティック関数とする．$\mathcal{P}$ 上のノード $v, u, w \in V$ が与えられたとき H が次の条件を満たすならば，H は $\mathcal{P}$ において

v に対して無矛盾であるという.

$$H(v, v) = 0$$

かつ

$$H(u, v) \leq C(u, w) + H(w, v).$$

また，H が $\mathcal{P}$ において任意のノード $v \in V$ に対して無矛盾であるならば，H は $\mathcal{P}$ において無矛盾であるという.

経路探索問題 $\mathcal{P}$ においてヒューリスティック関数 H が無矛盾であるならば，H は $\mathcal{P}$ において許容的であることが知られている [12].

第3章

単一目標に対する経路探索問題の拡張

3.1 経路コストが変化する場合

Koenig と Likhachev は，漸進的ヒューリスティック探索の 1 つとして *Adaptive A** を提案した [7]．今後，t 回目の探索を行い，エージェントが移動し，$t+1$ 回目の探索が始まるまでを t **回目の探索エピソード**と呼ぶ．エージェントの移動は，経路コストの変化，目標の移動といった要因により中断され，次の探索が開始される．Adaptive A*は探索エピソード毎にヒューリスティック関数を更新しながら，ステップ毎に $\mathcal{P}$ が変化する場合の探索問題を効率的に解くことができる．今，t 回目の探索エピソードにおける経路探索問題を $\mathcal{P}^t$ とする．Adaptive A*は，任意のエッジ $e \in E^t$ におけるコスト $C^t(e)$ の増加，すなわち $C^t(e) \leq C^{t+1}(e)$ のようなコスト C^t の変化にも対応できる．$\mathcal{P}^t$ の他の要素は，$\mathcal{P}^{t-1}$ の対応する要素とそれぞれ等しい．

ここで，慣例に従い t 回目の探索エピソードにおけるノード v から目標 γ までのヒューリスティック関数として $h^t_\gamma(v)$ という表記を用いる．

$(t+1)$ 回目の探索エピソードにおけるヒューリスティック関数 $h^{t+1}_\gamma(v)$ は，次の更新式によって計算される．

$$h^{t+1}_\gamma(v) = g^t(\gamma) - g^t(v) \tag{3.1}$$

ここで，$g^t(v)$ は**経路コスト関数**と呼ばれ，初期ノード σ から t 回目の探索エピソードに展開されたノード v までの実際の最小コストを表す．1 回の探索が終了する度に，展開された全てのノードに対して式 (3.1) を適用することでヒューリスティック関数を更新する．Koenig と Likhachev は，もし h^0_γ が $\mathcal{P}^0$ において γ に対して無矛盾ならば，

任意の $t \in \mathbf{N}$ について式 (3.1) によって更新された h_γ^t は $\mathcal{P}^t$ において γ に対して無矛盾であると証明した [8]．さらに，彼らは h_γ^t は t に対して単調非減少であり，従って時間とともに優位になることを示した．つまり，任意の $t \in \mathbf{N}$ に対して $h_\gamma^t(v) \leq h_\gamma^{t+1}(v)$ が成り立つ．

その後，Sun らによって任意のエッジ $e \in E$ においてコスト $C^t(e)$ が減少した場合にも対応できるように拡張した *Generalized Adaptive A**が提案された [15]．コストが減少した場合，Generalized Adaptive A*は即座にコストが減少したノードから幅優先探索を行い，無矛盾性が崩れている箇所について h_γ^t を更新することで h_γ^{t+1} の無矛盾性を保つ．

3.2 目標が移動する場合

Γ^t が移動する場合，つまりターゲットが動き得る経路探索問題については，いくつかの既存研究が存在する [5, 13, 11, 16]．Koenig らは，Adaptive A*をターゲットが動く場合に対応させた拡張手法，*Moving-Target Adaptive A**を提案した [11]．t 回目の探索エピソードにおいてターゲットが γ^t から γ^{t+1} に移動した場合，Moving-Target Adaptive A*は次の更新式によって $h_{\gamma^{t+1}}^{t+1}$ を計算する．

$$h_{\gamma^{t+1}}^{t+1}(v) = \max\left\{ H(v, \gamma^{t+1}),\, h_{\gamma^t}^t(v) - h_{\gamma^t}^t(\gamma^{t+1}) \right\} \tag{3.2}$$

式 (3.2) はターゲットが移動した場合に適用され，探索空間中の全てのノードにおいてヒューリスティックの更新が行われる．. 彼らは，H が $\mathcal{P}^0$ において無矛盾であり，かつ $h_{\gamma^0}(v) := H(v, \gamma^0)$ であるならば，任意の $t \in \mathbf{N}$ について式 (3.2) によって更新された $h_{\gamma^t}^t$ は $\mathcal{P}^t$ において γ^t に対して無矛盾であると示した．

次章からは，目標が複数ある場合についてのアルゴリズムを提案する．

第4章

OR 設定

4.1 定義

与えられた経路探索問題 $\mathcal{P}$ において目標が複数あり，その中の最も近い目標の 1 つまでの最短経路を求める問題を OR **設定**と呼び，求める解を以下のように定義する．

定義 5 (OR 設定の解)**.** 経路探索問題 $\mathcal{P}$ が与えられたとき，$\mathcal{P}$ における OR 設定の解 p^*_{or} を初期ノード σ から目標ノード集合 Γ 内の最も近いノードまでの最短経路と定義する．すなわち，

$$p^*_{or} := argmin_{p \in P_{or}(\sigma,\Gamma)} C(p)$$

と定義する．ここで，$P_{or}(\sigma,\Gamma) := \bigcup_{\gamma\in\Gamma} P(\sigma,\gamma)$ である．

4.2 素朴な手法

素朴な手法としては，Γ に含まれる全ての目標ノードに対してそれぞれ最短経路を計算すればよい．この場合，明らかに $|\Gamma|$ 回の A*の計算が必要となる．A*の最悪時間計算量を $ASTAR$ とすると，無矛盾なヒューリスティック関数を用いた場合には $ASTAR = O(l \log m)$ であるため，素朴な手法の最悪時間計算量は，$O(nl \log m)$ となる．ここで，$n = |\Gamma|$，$l = |E|$，$m = |V|$ である．この章では，まず単一解に対するヒューリスティック関数の与え方を変えることで，$\mathcal{P}$ における OR 設定の解を 1 回の A*の計算で求める方法を提案し，次にその正当性について述べる．

経路探索問題$\mathcal{P}$が与えられたとき，$\mathcal{P}$におけるOR設定の解の最適コスト関数$h^*_{or}(v)$をOR設定の解の最小コストと定義する．つまり，

$$h^*_{or}(v) := \min_{p \in P_{or}(v,\Gamma)} C(p)$$

である．ここで，$\mathcal{P}$におけるOR設定の解のヒューリスティック関数としてh_{or}を考える．h_{or}はγに対する単一解のヒューリスティック関数であるh_γと同じ形式であるため，OR設定の解を求める際にそのままA*を用いることができる．

次に，h_{or}における許容性と無矛盾性を，単一解の場合と同様に定義する．

定義 6 (h_{or}における許容性)**.** 任意のノード$v \in V$に対して

$$h_{or}(v) \leq h^*_{or}(v)$$

が成り立つならば，h_{or}は$\mathcal{P}$においてOR設定の解に対して許容的であると言う．

定義 7 (h_{or}における無矛盾性)**.** 任意の目標$\gamma \in \Gamma$に対して

$$h_{or}(\gamma) = 0$$

であり，かつ任意のエッジ$(u, v) \in E$に対して

$$h_{or}(u) \leq C(u, v) + h_{or}(v)$$

が成り立つならば，h_{or}は$\mathcal{P}$においてOR設定の解に対して無矛盾であると言う．

4.3 固定型目標に対する探索

OR設定において，目標がある位置に固定され移動しない場合を考える．この問題を解く際に，以下の式で定義されたヒューリスティック関数h_{or}を用いる．

$$h_{or}(v) := \min_{\gamma \in \Gamma} h_\gamma(v) \tag{4.1}$$

式(4.1)における各目標γへのヒューリスティック関数h_γには，マンハッタン距離のような普段用いているヒューリスティック関数を用いることができる．次の定理では，h_{or}が無矛盾であることを示す．

定理 8. 任意の経路探索問題 $\mathcal{P}$ において，任意の目標 $\gamma \in \Gamma$ に対して h_γ が無矛盾であるならば，式 *(4.1)* によって定義されるヒューリスティック関数 h_{or} もまた *OR* 設定の解に対して無矛盾である．

Proof. 任意の目標 $\gamma \in \Gamma$ に対し，

$$h_{or}(\gamma) = \min_{\gamma \in \Gamma} h_\gamma(\gamma) = 0$$

が成り立つ．また，任意のエッジ $(u, v) \in E$ に対し次式が成り立つ．

$$h_{or}(u) \leq \min_{\gamma \in \Gamma} \{\, C(u, v) + h_\gamma(v) \,\} = C(u, v) + h_{or}(v)$$

□

上の定理より，固定型目標の場合の多目標 OR 経路探索問題に対して，簡単に Adaptive A*を適用できることがわかる．さらに，Adaptive A*の性質より以下の系が示せる．

系 9. t 回目の探索エピソードにおける経路探索問題を $\mathcal{P}^t$ とし，$\mathcal{P}^t$ における *OR* 設定のヒューリスティック関数を h^t_{or} とする．もし h_γ が $\mathcal{P}^0$ において任意の目標 $\gamma \in \Gamma^0$ に対して無矛盾であり，かつ $\gamma \in \Gamma^0$ が式 *(4.1)* で h_{or} によって初期化されたならば，式 *(3.1)* によって更新された h^t_{or} も，任意の $t \in \mathbf{N}$ で $\mathcal{P}^t$ において *OR* 設定の解に対して無矛盾である．また，そのような h^t_{or} は t とともに単調非減少である．従って時間とともに優位になっていく．すなわち，任意の $t \in \mathbf{N}$ に対して $h^t_{or}(v) \leq h^{t+1}_{or}(v)$ が成り立つ．

4.4 移動型目標に対する探索

OR 設定において，目標が時間によって移動する場合を考える．目標が移動する場合には，直接 Moving-Target Adaptive A*を用いることはできない．そこで，式 (3.2) の代わりに以下の更新式を定義する．

$$h^{t+1}_{or}(v) = \max \left\{ \min_{\gamma \in \Gamma^{t+1}} H(v, \gamma),\ h^t_{or}(v) - \max_{\gamma \in \Gamma^{t+1}} h^t_{or}(\gamma) \right\} \tag{4.2}$$

次の定理では，この正当性を示す．

定理 10. t 回目の探索エピソードにおける経路探索問題を $\mathcal{P}^t$，ヒューリスティック関数を H とし，さらに $\mathcal{P}^t$ における OR 設定の解に対するヒューリスティック関数を h^t_{or} とする．もし H が $\mathcal{P}^0$ において無矛盾であり，かつ $h_{\gamma^0}(v) := H(v, \gamma^0)$ が成り立つならば，式 *(4.2)* によって更新された h^t_{or} も，任意の $t \in \mathbf{N}$ で $\mathcal{P}^t$ において OR 設定の解に対して無矛盾である．

Proof. $t = 0$ の場合，明らかに h^0_{or} は $\mathcal{P}^0$ において OR 設定の解に対して無矛盾である．

$t > 0$ の場合，h^t_{or} が $\mathcal{P}^t$ において OR 設定の解に対して無矛盾だと仮定する．この仮定より，任意の目標 $\gamma' \in \Gamma^{t+1}$ において，

$$h^{t+1}_{or}(\gamma') \;=\; \max\left\{0, h^t_{or}(\gamma') - \max_{\gamma \in \Gamma^{t+1}} h^t_{or}(\gamma)\right\} = 0,$$

が成り立ち，かつ任意のエッジ $(v, u) \in E$ において

$$\begin{aligned} h^{t+1}_{or}(v) &\leq C^t(v,u) + \max\left\{\begin{array}{c} \min\limits_{\gamma \in \Gamma^{t+1}} H(u,\gamma), \\ h^t_{or}(u) - \max\limits_{\gamma \in \Gamma^{t+1}} h^t_{or}(\gamma) \end{array}\right\} \\ &= C^{t+1}(v,u) + h^{t+1}_{or}(u). \end{aligned}$$

が成り立つ．以上より，h^{t+1}_{or} は $\mathcal{P}^{t+1}$ において OR 設定の解に対して無矛盾である．

数学的帰納法により，任意の整数 $t \geq 0$ に対して h^t_{or} は $\mathcal{P}^t$ において無矛盾である． □

従って，OR 設定の場合には，初期ヒューリスティックの与え方と更新式を工夫することで，単一目標の場合と同様の計算で最適解を求めることができる．この時の最悪時間計算量は $O(l \log m)$ となる．

第5章

AND 設定

5.1 定義

与えられた経路探索問題 $\mathcal{P}$ において目標が複数あり，その目標全てに到達する最短経路を求める問題を AND **設定**と呼び，求める解を以下のように定義する．

定義 11 (AND 設定の解)**.** 経路探索問題 $\mathcal{P}$ が与えられたとき，$\mathcal{P}$ における AND 設定の解 p^*_{and} を，初期ノード σ から始めて目標ノード集合 Γ 内の全てのノードを通過する最短経路と定義する．すなわち，

$$p^*_{and} := argmin_{p \in P_{and}(\sigma,\Gamma)} C(p)$$

と定義する．ここで，$P_{and}(\sigma,\Gamma) := \bigcup_{\pi \in \Pi_n} P(\sigma, \gamma_{\pi[1]}, \cdots, \gamma_{\pi[n]})$ である．

ノード $v \in V$ はエージェントがどのノードに到達したかという情報を持っていないため，OR 設定での手法を AND 設定に適用することはできない．この章では，Adaptive A*を用いるための 3 つの手法を提案する．

5.2 直接的手法

最初に直接的手法を考える．この手法では，まず与えられた経路探索問題 $\mathcal{P}$ の任意の 2 目標 $\gamma_1, \gamma_2 \in \Gamma$ 間の最適コスト $H^*(\gamma_1, \gamma_2)$ を求め，次に最適コストを重みとしたグラフ G' における最短ハミルトン路を求める．このアルゴリズムを StrPlan と呼ぶ．また G' を**抽象グラフ**と呼び，以下のように形式化する．

入力: 経路探索問題 $\mathcal{P}$ とヒューリスティック関数 H

出力: $\mathcal{P}$ における AND 設定の解

foreach $u, v \in \Gamma \cup \{\sigma\}$ **do**
    $Q(u,v) \leftarrow$ A*を用いて求められた u から v までの最短経路;
    $H^*(u,v) \leftarrow C(Q(u,v))$;
end
$p^*_{shp} \leftarrow \sigma$ を初期状態とした抽象グラフ G' 上の最短ハミルトン路;
$p^*_{and} \leftarrow (Q(p^*_{shp}[1]), \cdots, Q(p^*_{shp}[n]))$;
return p^*_{and};

Algorithm 1: StrPlan

定義 12 (抽象グラフ)**.** 経路探索問題 $\mathcal{P}$ と $\mathcal{P}$ におけるヒューリスティック関数 H に対し，$\mathcal{P}$ における H に関しての抽象グラフ G' を次のように定義する．

$$G' := (V', E', C')$$

ここで，$V' := \Gamma \cup \{\sigma\}$，$E' := \{(u,v) \mid u, v \in V'\}$，そして任意のノード $u, v \in V'$ において $C'(u,v) := H(u,v)$ である．

経路探索問題 $\mathcal{P}$ と $\mathcal{P}$ におけるヒューリスティック関数 H が与えられたとき，抽象グラフ G' を作成し，Algorithm 1 で示す StrPlan アルゴリズムを実行することで，$\mathcal{P}$ における AND 設定の解を求める．このアルゴリズムは一見正しい AND 設定の解を返すように見えるが，議論しなければならない点が残っている．σ を初期状態とする G' 上の最短ハミルトン路は，$\gamma \in \Gamma$ に含まれる各目標状態を 1 度だけ通る．それに対し，$\mathcal{P}$ 上の AND 設定の解は目標状態を 2 度以上通過する可能性がある．そのため，$p^*_{shp}[i] = (u_i, v_i) \in V'$ である G' 上の最短ハミルトン路 p^*_{shp} が求められたとき，その最後の目標ノード v_n を，途中のいずれかの u_i から v_i までの最短経路 $Q(p^*_{shp}[i])$ 上で既に通過しているという可能性が考えられる．このことを考慮すると，StrPlan アルゴリズムは間違った解を返すかもしれないと考えるかもしれない．そこで次の定理では，このアルゴリズムが正しい解を返すことを示す．

定理 13. 任意の経路探索問題 $\mathcal{P}$ において，AND 設定の解 p^*_{and} の総コストは，$\mathcal{P}$ における最適コスト関数 H^* を用いた抽象グラフ $G' = (V', E', C')$ 上の最短ハミルトン路 p^*_{shp} のコストに等しい．すなわち以下の等式が成り立つ．

$$C(p^*_{and}) = C'(p^*_{shp})$$

Proof.

$$\begin{aligned}
C(p^*_{and}) &= \min_{p \in P_{and}(\sigma,\Gamma)} C(p) \\
&= \min_{\pi \in \Pi_n} \min_{p \in P(\sigma, \gamma_{\pi[1]}, \cdots, \gamma_{\pi[n]})} C(p) \\
&= \min_{\pi \in \Pi_n} \left\{ \min_{p_1 \in P(\sigma, \gamma_{\pi[1]})} C(p_1) + \cdots \right. \\
&\qquad \left. \cdots + \min_{p_n \in P(\gamma_{\pi[n-1]}, \gamma_{\pi[n]})} C(p_n) \right\} \\
&= \min_{\pi \in \Pi_n} \left\{ H^*(\sigma, \gamma_{\pi[1]}) + \cdots + H^*(\gamma_{\pi[n-1]}, \gamma_{\pi[n]}) \right\} \\
&= \min_{\pi \in \Pi_n} \left\{ C'(\sigma, \gamma_{\pi[1]}) + \cdots + C'(\gamma_{\pi[n-1]}, \gamma_{\pi[n]}) \right\} \\
&= C'(p^*_{shp}).
\end{aligned}$$

□

StrPlan アルゴリズムは，最短経路の計算をちょうど $n(n+1)/2$ 回だけ呼び出す．ここで，$n = |\Gamma|$ である．さらに最短ハミルトン路の計算を 1 回呼び出す．最短ハミルトン路の計算に要する最悪時間計算量を SHP とすると，$SHP = O(n^2 2^n)$ である．従って，StrPlan アルゴリズムの最悪時間計算量は，$O(n^2 ASTAR + SHP) = O(n^2(l \log m + 2^n))$ となる．

5.3 漸進的手法

次に漸進的手法を考える．この手法は，Algorithm 2 で示すようにヒューリスティック関数 H を更新しながら，H を用いて作成した抽象グラフ G' における最短ハミルトン路を繰り返し計算する，というものである．この手法を IncPlan アルゴリズムと呼ぶ．

入力: 経路探索問題 $\mathcal{P}$ とヒューリスティック関数 H

出力: $\mathcal{P}$ における AND 設定の解

1 $F \leftarrow (\Gamma \cup \{\sigma\}) \times \Gamma$;
2 **while** *true* **do**
3 $p_{shp} \leftarrow H$ から得られた抽象グラフ G' 上における最短ハミルトン路;
4 **if** $F = \emptyset$ **または** $\forall i \in [1, n], p_{shp}[i] \notin F$ **then** break;
5 **foreach** (u, v) *in* p_{shp} **do**
6 **if** $(u, v) \in F$ **then**
7 $Q(u, v) \leftarrow$ A*によって計算された u から v までの最短経路;
8 $H(u, v) \leftarrow C(Q(u, v))$;
9 $F \leftarrow F - \{(u, v)\}$;
10 **end**
11 **end**
12 **end**
13 $p^*_{and} \leftarrow (Q(p_{shp}[1]), \cdots, Q(p_{shp}[n]))$;
14 **return** p^*_{and};

Algorithm 2: IncPlan

IncPlan アルゴリズムの 4 行目は終了条件を表している．このアルゴリズムは明らかに，高々$|F|$ 回の繰り返しで終了する．$F = \emptyset$ の条件が満たされる場合は，任意の 2 目標状態間の最小コストが既に計算されているため，定理 13 によって IncPlan アルゴリズムの正当性が導かれる．任意の $i \in [1, n]$ に対して $p_{shp}[i] \notin F$ の条件が満たされる場合は，次の定理より IncPlan アルゴリズムは常に正しい解を返すことが言える．

定理 14. $G = (V, E, C)$ と $G' = (V, E, C')$ を，同じノード集合とエッジ集合を持った重み付きグラフとする．任意のノード $v \in V$ において p を G における v からの最短ハミルトン路と仮定すると，もし任意の $i \in [1, |p|]$ において $C(p[i]) = C'(p[i])$ であり，かつ任意のエッジ $e \in E$ に対して $C(e) \leq C'(e)$ であるならば，p は G' における v からの最短ハミルトン路である．すなわち，以下の式が成り立つ．

$$C(p) = \min_{p' \in P_{inc}(V)} C'(p')$$

ここで，$P_{inc}(V) := \bigcup_{\pi \in \Pi_{|p|}} P(v_{\pi[1]}, \cdots, v_{\pi[|p|]})$ であり，v_i は V における任意の順番での i 番目のノードを表している．

Proof. 任意の $i \in [1, |p|]$ において $C(p[i]) = C'(p[i])$ であるため，

$$C(p) = C'(p) \geq \min_{p' \in P_{inc}(V)} C'(p')$$

である．一方，任意のエッジ $e \in E$ において $C(e) \leq C'(e)$ なので，

$$C(p) = \min_{p' \in P_{inc}(V)} C(p') \leq \min_{p' \in P_{inc}(V)} C'(p')$$

が成り立つ． □

IncPlan アルゴリズムは，$\mathcal{P}$ における最短経路の計算を高々$n(n+1)/2$ 回，G' における最短ハミルトン路の計算を高々$(n^2 - n + 2)/2$ 回呼び出す．従って，IncPlan アルゴリズムの最悪時間計算量は，$O(n^2 ASTAR + n^2 SHP) = O(n^2(l \log m + n^2 2^n))$ となる．

5.4 変換手法

最後に変換手法を考える．この手法は，与えられた経路探索問題 $\mathcal{P}$ を別の問題 $\mathcal{P}'$ に変換することで，A*を 1 回呼び出すだけで解けるようにしたものである．変換の方法としては，各ノード $v \in V$ にバイナリ $b \in B_n$ を付与する．ここで，$b[i]$ はエージェントが既に目標ノード $\gamma_i \in \Gamma$ に到達したかどうかを表している．変換後の目標状態の集合 Γ' は，1^n のバイナリを持つ全ての目標ノードから成る．エージェントが $\mathcal{P}$ における AND 設定の解を求めるためには，$\mathcal{P}'$ における OR 設定の解を求めればよい．この変換を次のように形式化する．

定義 15 (OR 変換)**.** 経路探索問題 $\mathcal{P}$ が与えられた時、$\mathcal{P}$ の OR 変換を次のように定義する．

$$\mathcal{P}' := (G', \sigma', \Gamma')$$

また，G' は $G' := (V', E', C')$ の形で表される．各要素の定義は次のように与える．

- $V' := \{\, (v,b) \,|\, v \in V, b \in B_n, \phi(v,b) = 1 \,\}$．ここで

$$\phi(v,b) := \begin{cases} 1 & (v = \gamma_i \in \Gamma \quad \Rightarrow \quad b[i] = 1) \\ 0 & (otherwise) \end{cases}$$

- $E' := \{\, (u',v') \,|\, u', v' \in V', \psi(u',v') = 1 \,\}$．ここで $u' = (u, b_u)$，$v' = (v, b_v)$ はバイナリを付与したノードであり，$\psi(u',v')$ は次式で表される．

$$\psi(u',v') := \begin{cases} 1 & \left(\begin{array}{c} (u,v) \in E \\ v \notin \Gamma \quad \Rightarrow \quad b_v = b_u \\ v = \gamma_i \in \Gamma \quad \Rightarrow \quad b_v = r_i(b_u) \end{array} \right) \\ 0 & (otherwise) \end{cases}$$

- 任意の $e' \in E'$ に対して $C'(e') := C(e)$.

- $\sigma' := (\sigma, 0^n)$.

- $\Gamma' := \{\, (\gamma, b) \in V' \mid \gamma \in \Gamma, b = 1^n \,\}$.

OR 変換の正当性を示すために，まず次の補題を証明する．

補題 16. 任意の $\mathcal{P}$ について，$\mathcal{P}$ における AND 設定の経路集合 $P_{and}(\sigma, \Gamma)$ は，次の集合と等しい．

$$\hat{P}_{or}(\sigma', \Gamma') := \left\{ \hat{p} \;\middle|\; \begin{array}{c} \hat{p}[i] := (u_i, v_i), \forall p' \in P_{or}(\sigma', \Gamma') \\ s.t.\ p'[i] := ((u_i, b_{u,i}), (v_i, b_{v,i})) \end{array} \right\}$$

ここで $P_{or}(\sigma', \Gamma')$ は，$\mathcal{P}$ を OR 変換した経路探索問題 $\mathcal{P}' = (G', \sigma', \Gamma')$ における OR 設定の経路集合である．すなわち，次の式が成り立つ．

$$P_{and}(\sigma, \Gamma) = \hat{P}_{or}(\sigma', \Gamma')$$

Proof. ($\Rightarrow$) $p \in P_{and}(\sigma, \Gamma)$ を考え，$p[i] := (u_i, v_i)$ とする．$p'[i] := ((u_i, b_{u,i}), (v_i, b_{v,i}))$ であるような経路 p' を構成することができる．ここで，

$$b_{u,i} := \begin{cases} 0^n & (i = 1) \\ b_{v,i-1} & (i > 1) \end{cases}$$

であり，

$$b_{v,i} := \begin{cases} b_{u,i} & (v_i \notin \Gamma) \\ r_j(b_{u,i}) & (v_i = \gamma_j \in \Gamma) \end{cases}$$

である．OR 変換の定義より，$p' \in P_{or}(\sigma', \Gamma')$ が得られる．従って，$p \in \hat{P}_{or}(\sigma', \Gamma')$ が成り立つ．

($\Leftarrow$) $\hat{p} \in \hat{P}_{or}(\sigma', \Gamma')$ を考え，$\hat{p}[i] := (\hat{u}_i, \hat{v}_i)$ とする．$\hat{P}_{or}(\sigma', \Gamma')$ の定義より，$\hat{u}_i = u'_i$ かつ $\hat{v}_i = v'_i$ であるような $p' \in P_{or}(\sigma', \Gamma')$ が存在する．ここで，$p'[i] := ((u'_i, b'_{u,i}), (v'_i, b'_{v,i}))$ である．OR 変換の定義より，$b'_{u,1} = 0^n$ であり，$b'_{v,n} = 1^n$ である．すなわち，$\hat{u}_i = \sigma$ かつ $\hat{v}_n = \gamma \in \Gamma$ である．OR 変換における ψ の定義より，任意の目標状態 $\gamma \in \Gamma$ に対し，$\hat{v}_i = \gamma$ であるような i が存在する．従って，$\hat{p} \in P_{and}(\sigma, \Gamma)$ が成り立つ． □

次の定理では，$\mathcal{P}$ における AND 設定を解くことと，$\mathcal{P}$ を OR 変換した $\mathcal{P}'$ における OR 設定を解くことは確かに等しいことを示す．

定理 17. 任意の経路探索問題 $\mathcal{P}$ において，$\mathcal{P}$ における *AND* 設定の解 p^*_{and} のコストは，$\mathcal{P}$ の *OR* 変換における *OR* 設定の解 p^*_{or} のコストに等しい．すなわち，次式が成り立つ．

$$C(p^*_{and}) = C'(p^*_{or})$$

Proof. 補題 16 より，次の式を導くことができる．

$$\begin{aligned} C(p^*_{and}) &= \min_{p \in P_{and}(\sigma,\Gamma)} C(p) \\ &= \min_{\hat{p} \in \hat{P}_{or}(\sigma',\Gamma')} C(\hat{p}) \\ &= \min_{p' \in P_{or}(\sigma',\Gamma')} C'(p') \\ &= C'(p^*_{or}) \end{aligned}$$

□

$\mathcal{P}'$ における OR 設定のヒューリスティック関数を設定する際に，$\mathcal{P}$ で用いたマンハッタン距離のような通常の方法を用いることはできない．そこで次の定理では，$\mathcal{P}$ における単一解のヒューリスティックを用いて $\mathcal{P}'$ での無矛盾なヒューリスティック関数を設定する方法を示す．

定理 18. max_0 と min_0 を，$S \neq \emptyset$ ならばそれぞれ $\max S$ と $\min S$ を返し，そうでなければ *0* を返す関数とする．任意の経路探索問題 $\mathcal{P}$ において，$\mathcal{P}$ を *OR* 変換したものを $\mathcal{P}'$ とする．もしヒューリスティック関数 H が $\mathcal{P}$ において無矛盾ならば，次式で定義する $\mathcal{P}'$ におけるヒューリスティック関数 h_{cnv} もまた $\mathcal{P}'$ において無矛盾である．

$$h_{cnv}(v') := \max \left\{ \begin{array}{c} max_{0\gamma \in R(b_v)} H(v,\gamma) \\ min_{0\gamma \in R(b_v)} H(v,\gamma) + k(\,|\,R(b_v)\,|\,-1) \end{array} \right\}$$

ここで，$v' := (v, b_v)$，$R(b) := \{\, \gamma_i \in \Gamma \mid b[i] = 0 \,\}$，そして $k := \min_{e \in E} C(e)$ である．

Proof. $v' \in \Gamma'$ の場合，明らかに $R(b) = \emptyset$ である．従って，次式が成り立つ．

$$h_{cnv}(v') = \max\{\, 0, -k \,\} = 0$$

次に，h_{cnv} が三角不等式を満たすことを示す．$(v', u') \in E'$ を，$v' := (v, b_v)$，$u' := (u, b_u)$ であるようなエッジとする．H が $\mathcal{P}$ において無矛盾であり，また $(v, u) \in E$ なので，次の式が導ける．

$$h_{cnv}(v') \leq C(v,u) + \max \left\{ \begin{array}{c} max_{0\gamma \in R(b_v)} H(u,\gamma) \\ min_{0\gamma \in R(b_v)} H(u,\gamma) + k(\,|\,R(b_v)\,|\,-1) \end{array} \right\}$$

$v' \in \Gamma'$ の場合，明らかに $h_{cnv}(v') = 0 \leq C'(u', v') + h_{cnv}(v')$.

$v' \notin \Gamma'$ かつ $u \notin \Gamma$ の場合，OR 変換の ψ の定義より $b_v = b_u$ が得られる．つまり，$R(b_v) = R(b_u)$ である．従って $h_{cnv}(v') \leq C(v,u) + h_{cnv}(u') = C'(v', u') + h_{cnv}(u')$ が得られる．

$v' \notin \Gamma'$ かつ $u \in \Gamma$ の場合，H の無矛盾性より $H(u,u) = 0$ が導かれる．また，ψ の定義より $R(b_v) = R(b_u) \cup \{u\}$ が得られる．max 操作の最初の項は，

$$max_{0\gamma \in R(b_v)} H(u,\gamma) = max_{0\gamma \in R(b_u) \cup \{u\}} H(u,\gamma) = max_{0\gamma \in R(b_u)} H(u,\gamma)$$

であり，2 番目の項は $u \neq \gamma$ において $H(u,\gamma) \geq k$ かつ $k|\emptyset| = 0$ なので，

$$\begin{aligned} & min_{0\gamma \in R(b_v)} H(u,\gamma) + k(|R(b_v)| - 1) \\ = \;& min_{0\gamma \in R(b_u) \cup \{u\}} H(u,\gamma) + k(|R(b_u) \cup \{u\}| - 1) \\ = \;& k|R(b_u)| \\ \leq \;& min_{0\gamma \in R(b_u)} H(u,\gamma) + k(|R(b_u)| - 1) \end{aligned}$$

である．従って，$h_{cnv}(v') \leq C'(v', u') + h_{cnv}(u')$ が得られる． □

ここで留意すべきは，変換手法では A*の呼び出しが 1 度になっているが，探索空間は大きくなっていることである．変換手法の最悪時間計算量は，$O(|E'| \log |V'|) = O(2^n |E| \log(2^n |V|)) = O(2^n l \log(2^n m))$ となる．

第6章

実験

6.1 実験条件

提案手法の適用例として，図 6.1 に示すような 100×100 の迷路上での経路探索問題 $\mathcal{P}$ を考える．G はグリッド状の 2 次元格子グラフとして定義される．任意のエッジ $(u, v) \in E$ に対して，u と v が共に塞がれていなければ $C(u, v) = 1$，それ以外ならば $C(u, v) = \infty$ でコストを設定する．$C(u, v)$ は一定の確率 $\alpha \in \mathbf{R} : 0 \leq \alpha \leq 1$ で変化する．具体的には，ノード $v \in V$ が塞がれているならば塞がれていない状態に，塞がれていないならば塞がれている状態に変化する．ターゲットもまた一定の確率 $\beta \in \mathbf{R} : 0 \leq \beta \leq 1$ で隣接するノードへと移動する．こちらは，対応する目標状態 $\gamma \in \Gamma$ が隣接するノードに移動する．エージェントが目標に到達するまでを 1 試行とし，各試行が終了する度にエージェントの初期ノードと目標ノードの集合 Γ をランダムに再配置する．また，エージェントがあらかじめ地形を知らない場合についても考える．この場合，エージェントは観測によって自分のノードと 4 近傍のノードについての情報（塞がれているかそうでないかなど）を得ることができる．一度観測したノードについては，その試行の間は情報を保持しているものとする．全ての実験において，Adaptive A*の最新の拡張である Generalized Adaptive A* [15] を用いる．ターゲットの数は n と表記する．

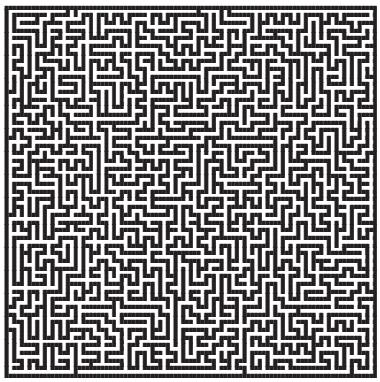

図 6.1: 迷路

6.2 OR 設定

ここでは，NaivePlan と MinPlan の比較を行う．NaivePlan は全ての目標ノードに対して最短経路を求める素朴な手法を，MinPlan は式 (4.1) のヒューリスティック関数を用いた提案手法を表している．500 回の試行を行い，(a)1 試行あたりの平均探索数，(b)1 試行あたりの平均移動数，(c)1 試行あたりの平均展開数，そして (d)1 試行あたりの平均実行時間 [ms] を求めた．ターゲットが固定の場合と動く場合，それぞれについて実験を行った．さらに両方の実験において，エージェントが地形情報を既に知っている場合と知らない場合とで条件を変え，ターゲットの数を 5，10，15 として探索手法の比較を行った．どちらの場合も，壁が変化する確率は $\alpha = 0.1$ で固定とした．

表 6.1，表 6.2 はターゲットを固定とした場合のデータをまとめたものである．それぞれ，地形情報を知っている場合と知らない場合のデータを表している．また，表 6.3，表 6.4 はターゲットが動く場合についてのデータをまとめたものである．こちらもそれぞれ，地形情報を知っている場合と知らない場合のデータを表している．ここで，ターゲットが移動する確率を $\beta = 0.1$ とした．つまり，エージェントが 1 ステップ移動するごとに，全てのターゲットがそれぞれ 10 %の確率でランダムに移動する．

NaivePlan と MinPlan を比較したとき，全ての結果において明らかに MinPlan の方が NaivePlan より良い結果が出ている．エージェントの平均移動回数は最短経路の長さを意味しているため，(b) の結果は 2 つの探索手法でほぼ等しくなっている．違いは，コストの変化やターゲットの移動などの確率的なものに起因すると考えられる．

表 6.1: OR 設定 (固定型目標，地形情報を知っている場合)

	(a)	(b)	(c)	(d)
ターゲット数	平均探索数	平均移動数	平均展開数	平均実行時間 [ms]
	NaivePlan			
5	54.81	98.99	87369.62	20.29
10	75.60	66.46	137012.91	27.98
15	90.78	51.66	173474.40	34.32
	MinPlan			
5	10.96	98.99	2373.26	1.08
10	7.56	66.46	1040.69	0.61
15	6.05	51.66	573.52	0.43

表 6.2: OR 設定 (固定型目標，地形情報を知らない場合)

	(a)	(b)	(c)	(d)
ターゲット数	平均探索数	平均移動数	平均展開数	平均実行時間 [ms]
	NaivePlan			
5	768.84	432.97	1612872.98	591.43
10	1012.56	276.79	2797733.59	1120.27
15	1022.52	181.77	3157937.16	1264.39
	MinPlan			
5	149.63	422.64	7502.98	31.30
10	105.74	291.22	4121.48	21.57
15	71.81	194.98	2119.70	14.08

表 6.3: OR 設定 (移動型目標，地形情報を知っている場合)

	(a)	(b)	(c)	(d)
ターゲット数	平均探索数	平均移動数	平均展開数	平均実行時間 [ms]
	NaivePlan			
5	232.19	97.77	397006.24	97.40
10	446.76	64.90	836500.39	236.71
15	594.84	48.58	1116453.08	349.50
	MinPlan			
5	46.44	97.77	9499.80	3.86
10	44.43	64.35	5005.14	2.90
15	42.72	52.24	3425.08	2.62

表 6.4: OR 設定 (移動型目標，地形情報を知らない場合)

	(a)	(b)	(c)	(d)
ターゲット数	平均探索数	平均移動数	平均展開数	平均実行時間 [ms]
	NaivePlan			
5	1345.01	434.88	2880903.98	1195.86
10	2129.80	273.57	6092105.11	2652.74
15	2672.34	203.92	8377218.38	3589.50
	MinPlan			
5	279.51	453.41	18097.55	23.49
10	200.26	257.62	8428.36	13.14
15	163.97	188.48	5520.92	9.98

NaivePlan は Adaptive A*を n 回呼び出すため，n が増加するに従って NaivePlan の (a) や (c)，(d) の結果も増加している．直感に反して，n が増加するに従って，MinPlan の全ての値と NaivePlan の (b) の値が減少している．これは n の増加に伴い，最も近いターゲットまでの距離が小さくなったためだと考えられる．

6.3 AND 設定

ここでは，StrPlan，IncPlan，CnvPlan の 3 つの手法を比較する．それぞれ，節 5.2，節 5.3，節 5.4 で提案された手法を表している．各手法の最悪時間計算量は，StrPlan が $O(n^2(l\log m+2^n))$，IncPlan が $O(n^2(l\log m+n^2 2^n))$，CnvPlan が $O(2^n l\log(2^n m))$ となる．ここで，n はターゲットの数，l はエッジの数，そして m はノードの数を表している．この実験では OR 設定の実験に追加し，(e)1 試行あたりの平均 A*呼び出し回数，(f)1 試行あたりの平均 SHP 呼び出し回数を求めた．ターゲットの数を変化させ，それぞれ 100 回の試行を行いその平均を求めた．

まず，ターゲット及びコストは固定として実験を行った．つまり，$\alpha = 0$，$\beta = 0$ として，ターゲットの数を 2 から 12 まで 2 刻みで変化させた．その結果をまとめたものが，表 6.5 である．$\alpha = 0$ かつ $\beta = 0$ なので，各手法とも得られる最短経路の長さが等しい．これは (b) の値からもわかることである．CnvPlan の (d) の結果に注目すると，他の 2 つの手法よりも実行時間が極めて大きい．これは，最悪時間計算量の $m \gg n$ の場合と同じ傾向である．一方で，最悪時間計算量で見ると StrPlan の方が IncPlan より良いにも関わらず，$n \leq 8$ の場合に IncPlan の方が実行時間が小さくなっている．この結果は，n が比較的小さい場合には IncPlan の方が StrPlan よりも実用的であることを示している．これは (e) の結果からもわかるように，Adaptive A*を実行する回数が IncPlan の方が平均的に少ないためと考えられる．

次に，ターゲット及びコストが変化する条件で実験を行った．変化する確率は，OR 設定の場合と同じく $\alpha = 0.1$，$\beta = 0.1$ とした．先ほどの実験より CnvPlan の実行時間が極めて大きいことがわかるため，StrPlan と IncPlan のみで比較を行った．今回は，ターゲットの数は 2 から 10 まで 2 刻みで変化させた．その結果をまとめたものが，表 6.6 である．(d) の結果に注目すると，ターゲットが移動する場合には IncPlan

の方が高速に解を求められている．これは，ターゲットの数が小さい場合には経路探索よりも最短ハミルトン路を求める方が計算が軽いためだと考えられる．実際に，今回の実験では n が最大でも 10 と，比較的小さい値を用いている．パックマンのような実際の状況においては，n はそれほど大きくないことが多い．現に，以前示したようにパックマンにはゴーストが 4 匹しか存在しない．以上の結果より，n が比較的小さい場合には StrPlan よりも IncPlan の方が実用的であると結論づける．

表 6.5: AND 設定 (固定型目標，地形情報を知っている，壁固定の場合)

	(a)	(b)	(c)	(d)	(e)	(f)
ターゲット数	平均探索数	平均移動数	平均展開数	平均実行時間 [ms]	平均 A* 回数	平均 SHP 回数
			StrPlan			
2	1	390.65	5134.89	1.02	3	1
4	1	581.54	15936.72	3.24	10	1
6	1	751.50	32133.77	6.94	21	1
8	1	927.84	54660.20	11.24	36	1
10	1	1091.01	89479.78	19.00	55	1
12	1	1231.73	128265.40	28.91	78	1
			IncPlan			
2	1	390.65	4204.60	1.02	2.67	2.67
4	1	581.54	10455.06	2.48	8.26	4.11
6	1	751.50	19340.37	4.77	17.24	6.47
8	1	927.84	28963.47	8.73	27.70	8.74
10	1	1091.01	40889.10	23.25	39.42	11.08
12	1	1231.73	52139.51	102.05	52.64	14.62
			CnvPlan			
2	1	390.65	5254.49	1.46	1	0
4	1	581.54	27227.24	9.19	1	0
6	1	751.50	111743.10	74.05	1	0
8	1	927.84	501397.10	435.48	1	0
10	1	1091.01	1707189.00	2091.85	1	0
12	1	1231.73	6403776.00	9936.36	1	0

表 6.6: AND 設定 (移動型目標，地形情報を知らない，壁変化の場合)

	(a)	(b)	(c)	(d)	(e)	(f)
ターゲット数	平均探索数	平均移動数	平均展開数	平均実行時間 [ms]	平均 A* 回数	平均 SHP 回数
	StrPlan					
2	563.16	1313.21	127988.08	141.12	1130.86	563.16
4	940.76	1985.61	632899.49	910.66	5087.52	940.76
6	1288.23	2476.56	1670939.79	2461.18	13771.51	1288.23
8	1508.82	2723.85	2725152.46	3052.70	25296.78	1508.82
10	1803.10	3085.55	4479755.65	5440.42	43322.37	1803.10
	IncPlan					
2	617.93	1447.83	122864.37	142.40	999.10	1265.41
4	973.16	2046.34	242898.16	569.17	2822.84	2022.78
6	1274.12	2470.05	439344.59	849.37	5428.25	2703.30
8	1542.41	2731.55	503090.62	1096.02	8906.53	3280.14
10	1809.14	3055.18	709256.66	2693.74	12887.53	3912.87

第7章

まとめと今後の課題

7.1 まとめ

本論文では，多目標状況下における経路探索にAdaptive A*を適用させた手法を提案した．まず，OR設定とANDを形式化した．それぞれの目的は，(1) エージェントは目標のうちどれか1つに到達すればよい，(2) エージェントは全ての目標に到達しなければならない，である．OR設定に関しては，Adaptive A*を用いるために，無矛盾なヒューリスティックの構築法を提案した．これは，ターゲットが固定の場合，移動する場合のどちらに対しても無矛盾性を保つ．AND設定に関しては，各ターゲットに対して直接Adaptive A*を用いることのできる3つの手法を提案した．また，全ての手法で常にAND設定における最適経路を得られることを示した．そして実験結果は，迷路のようなアプリケーションにおいて，OR設定，AND設定のどちらであっても提案手法が正確に機能することを示した．

7.2 今後の課題

AND設定に関して3つの手法を提案したが，これらの手法はいずれも改良の余地があると思われる．そのため，まずはより効率的な手法を提案する必要がある．また，本論文ではターゲットが複数の場合の経路探索問題を取り扱ったが，例えばエージェントが複数の場合を考え，エージェント同士がお互いの持っている情報をやり取りできるなど，様々な問題設定における経路探索問題についても考えたい．

参考文献

[1] Jeff Bueckert, Simon X. Yang, Xiaobu Yuan, and Max Q.-H. Meng. Neural Dynamics Based Multiple Target Path Planning for a Mobile Robot. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO 2007)*, pp. 1047–1052. IEEE, 2007.

[2] Dmitry Davidov and Shaul Markovitch. Multiple-Goal Heuristic Search. *Journal of Artificial Intelligence Research*, Vol. 26, pp. 417–451, 2006.

[3] Kurt Derr and Milos Manic. Multi-Robot Multi-Target Particle Swarm Optimization Search in Noisy Wireless Environments. In *Proceedings of the 2nd IEEE Conference on Human System Interaction (HSI 2009)*, pp. 81–86. IEEE, 2009.

[4] Dominik Henrich, Christian Wurll, and Heinz Wörn. Multi-Directional Search with Goal Switching for Robot Path Planning. In *Proceedings of the 11th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-98)*, Vol. 1416 of *Lecture Notes in Computer Science*, pp. 75–84. Springer-Verlag, 1998.

[5] Toru Ishida and Richard E. Korf. Moving-Target Search: A Real-Time Search for Changing Goals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 6, pp. 609–619, 1995.

[6] Sven Koenig and Maxim Likhachev. D* Lite. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pp. 476–483. American Association for Artificial Intelligence, 2002.

[7] Sven Koenig and Maxim Likhachev. Adaptive A*. In *Proceedings of 4th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2005)*, pp. 1311–1312. ACM, 2005.

[8] Sven Koenig and Maxim Likhachev. A new principle for incremental heuristic search: Theoretical results. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS 2006)*, pp. 402–405. AAAI, 2006.

[9] Sven Koenig and Maxim Likhachev. Real-time Adaptive A*. In *Proceedings of 5th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2006)*, pp. 281–288. ACM, 2006.

[10] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong Planning A*. *Artificial Intelligence*, pp. 93–146, 2004.

[11] Sven Koenig, Maxim Likhachev, and Xiaoxun Sun. Speeding up Moving-target Search. In *Proceedings of 6th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pp. 1136–1143. IFAAMAS, 2007.

[12] J. Pearl. *Heuristics*, Vol. 78. Addison-Wesley Reading, MA, 1984.

[13] Masashi Shimbo and Toru Ishida. Towards Real-Time Search with Inadmissible Heuristics. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, pp. 609–613. IOS Press, 2000.

[14] Xiaoxun Sun and Sven Koenig. The Fringe-Saving A* Search Algorithm - A Feasibility Study. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pp. 2391–2397, 2007.

[15] Xiaoxun Sun, Sven Koenig, and William Yeo. Generalized Adaptive A*. In *Proceedings of 7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, pp. 469–476. IFAAMAS, 2008.

[16] Xiaoxun Sun, William Yao, and Sven Koenig. Efficient Incremental Search for Moving Target Search. In *Proceedings of the 22th International Joint Conference on Artificial Intelligence (IJCAI 2009)*, pp. 615–620, 2009.

[17] István Szita and András Lörincz. Learning to play using low-complexity rule-based policies: Illustrations through ms. pac-man. *Journal of Artificial Intelligence Research*, Vol. 30, pp. 659–684, 2007.

[18] D. Zu, J.D. Han, and Mark Campbell. Artificial Potential Guided Evolutionary Path Plan for Multi-Vehicle Multi-Target Pursuit. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO 2004)*, pp. 855–861. IEEE, 2004.

研究業績

査読付き国際会議 1 件

- Kengo Matsuta, Hayato Kobayashi, and Ayumi Shinohara.

  "Multi-Target Adaptive A*". In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, pp. 1065-1072.

謝辞

平成 20 年 4 月から平成 23 年 3 月までの 3 年間にわたり，このような研究の場を与えていただくとともに，本研究の直接的な指導教員として多くのご教示，ご鞭撻を賜りました東北大学 大学院情報科学研究科 篠原 歩 教授，成澤 和志 助教に，心より深く感謝いたします．また，篠原教授からは，研究活動のみならず，教育に対する真摯なご姿勢や，人柄の温かさの大切さ，また人としてあるべき姿を学ばせて頂きましたことに，重ねて厚く御礼申し上げます．

また，本論文審査の副審査員を務めて頂きました，東北大学 大学院情報科学研究科 周 暁 教授，ならびに東北大学 大学院情報科学研究科 木下 哲男 教授には，御専門の立場から的確なご助言や貴重なご意見を賜りましたことを，心から御礼申し上げます．

また，本研究を進めていくうえで常日頃より時間をともにして下さり，多くのご助言，ご協力を頂きました，篠原研究室の先輩であり現株式会社東芝の小林 隼人氏に，心より感謝申し上げます．そして，日頃の研究室での活動全般にわたりご支援頂いた東北大学 大学院情報科学研究科 篠原研究室の皆様に心から感謝申し上げます．

最後に，東北大学 大学院情報科学研究科 博士前期課程までという長い間，進学の機会を与えてくださり，最後まで信じて見守ってくださった両親，私生活においていつも温かく支えてくれた友人一同に，深く感謝いたします．