Multi-Target Adaptive A*

Kengo Matsuta Hayato Kobayashi Ayumi Shinohara
Graduate School of Information Sciences, Tohoku University, Japan
{matsuta, kobayashi}@shino.ecei.tohoku.ac.jp ayumi@ecei.tohoku.ac.jp

ABSTRACT

Agents often have to solve series of similar path planning problems. Adaptive A* is a recent incremental heuristic search algorithm that solves such problems faster than A*, updating a heuristic function (also known as h-values) using information from previous searches. In this paper, we address path planning with multiple targets on Adaptive A* framework. Although we can solve such problems by calculating the optimal path to each target, it would be inefficient, especially when the number of targets is large. We consider two cases whose objectives are (1) an agent reaches one of the targets, and (2) an agent has to reach all of the targets. We propose several methods to solve such problems keeping consistency of a heuristic function. Our experiments show that the proposed methods properly work on an application, i.e., maze problems.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving

General Terms

Algorithms

Keywords

A*; Adaptive A*; Heuristic Search; Incremental Search; Shortest Paths; Multi-Target Search

1. INTRODUCTION

Agents often have to solve series of similar path planning problems. Adaptive A*, which is proposed by Koenig and Likhachev [7], is a recent incremental heuristic search algorithm that solves such problems faster than A*, updating a heuristic function (also known as h-values) using information from previous searches. They theoretically showed that the incremental procedure of Adaptive A* keeps consistency of a heuristic function [8]. Although the procedure is simpler than other competitive algorithms [6, 10, 13], its extensibility yields various versions of Adaptive A* [9, 11, 14, 15].

Cite as: Multi-Target Adaptive A*, Kengo Matsuta, Hayato Kobayashi, and Ayumi Shinohara, *Proc. of 9th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2010)*, van der Hoek, Kaminka, Lespérance, Luck and Sen (eds.), May, 10–14, 2010, Toronto, Canada, pp. 1065-1072

In this paper, we address path planning with multiple targets on Adaptive A* framework. We consider two cases whose objectives are (1) an agent has to reach one of the targets, and (2) an agent has to reach all of the targets. We call the former *OR setting* and the latter *AND setting*. For example, let us consider the Pac-Man video game as an application as in Szita and Lörincz [16]. If we want to improve the score, a good strategy is eating "ghosts" (or enemy) to earn bonus points by utilizing "power pills", which allow Pac-Man to eat ghosts for a few seconds. In this case, a path planning problem for eating the nearest power pill is OR setting, and the problem for eating all of the ghosts is AND setting.

There are a few studies about optimal path planning with multiple targets [4, 2], while there are many studies on heuristic algorithms without the guarantee of optimality, such as Particle Swarm Optimization [3], Evolutionary Algorithm [17], and Neural Network [1]. Our study is important in the sense that it supports the research domain of optimal path planning. At least, our study is the first approach to path planning with multiple targets in recently developed Adaptive A* framework.

2. PRELIMINARIES

We denote the size of a set S by $|S|$ and similarly the length of a sequence (or tuple) $p \in S^*$ by $|p|$. We denote by $p[i]$ the i-th element of a sequence p. Let $\mathbb{N}$ be the set of natural numbers. We define $[i,j] := \{ i, i+1, \cdots, j \}$ for any integers $i, j \in \mathbb{N}$ with $i \leq j$. Let Π_n be the set of all permutations of $[1,n]$. Let $\mathbb{R}$ and $\mathbb{R}_+$ be the sets of real numbers and positive real numbers, respectively.

Let $G := (V, E, C)$ be a weighted graph, where V is a set of nodes, $E \subseteq V^2$ is a set of edges, and $C : E \to \mathbb{R}_+$ is a weight function that attaches a positive weight to each edge. Given $e = (u, v) \in E$, we also write $C(u, v)$ instead of $C(e)$. A sequence $(e_1, \cdots, e_\ell) \in E^*$ of edges is called a *path from* $\sigma \in V$ *to* $\gamma \in V$, if letting $e_i = (v_i, v'_i)$, $v_1 = \sigma$, $v'_\ell = \gamma$, and $v'_i = v_{i+1}$ for any $i \in [1, \ell-1]$. We denote the set of all paths from σ to γ by $P(\sigma, \gamma) \subseteq E^*$. For any path $p \in P(\sigma, \gamma)$, we define the total cost of edges in p by

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

We define $P(v_1, \cdots, v_n) := P(v_1, v_2) \times \cdots \times P(v_{n-1}, v_n)$.

We denote the set of binaries of length n by $B_n := \{0,1\}^n$. For any binary $b \in B_n$, we define $r_i(b) := b[1] \cdots b[i-1] 1 b[i+1] \cdots b[n]$.

2.1 Path Planning Problem

We formalize our problem as follows.

Definition 1 (Path Planning Problem). *A path planning problem is given by a tuple*

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

where

- $G := (V, E, C)$ *is a weighted graph. A node* $v \in V$ *means a position where an agent and targets can stay. An edge* $e = (u, v) \in E$ *means that an agent and targets can move from a node* u *to a node* v*, and a weight* $C(e)$ *of the edge* e *means a cost required for its movement.*
- $\sigma \in V$ *is an initial node, where an agent starts to search.*
- $\Gamma \subseteq V$ *is a set of goal nodes, where targets stay. When the goals must be distinguished, we can access the* i*-th goal by* γ_i*. We use* n *as the number of goals, that is* $n := |\Gamma|$*, if not otherwise specified.*

The graph in a path planning problem is also called *a state space* in the literature on path planning. For the sake of simplicity, given a path planning problem $\mathcal{P}$, we use G, σ, Γ, V, E, and C without explanation, assuming that $\mathcal{P} = (G, \sigma, \Gamma)$ such that $G = (V, E, C)$. If $\mathcal{P}$ has some additional notation, all elements in $\mathcal{P}$ derive it, e.g., $\mathcal{P}' = (G', \sigma', \Gamma')$.

In the case of an ordinary path planning problem, most researchers focus only on path planning to a single target, even though some of them use the notation with multiple targets. The objective of path planning to a single target is that an agent reaches the target with the minimum total cost. We formalize it as a *single solution* defined below.

Definition 2 (Single Solution). *Given a path planning problem* $\mathcal{P}$*, we define that a* single solution on $\mathcal{P}$ with respect to γ *is a shortest path from the initial node* σ *to a goal node* $\gamma \in \Gamma$*, i.e.,*

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

This paper focuses on the case where an agent must consider all targets simultaneously. We consider two settings of solution, OR and AND defined in Section 4 and Section 5, respectively.

2.2 Heuristics

Given a path planning problem $\mathcal{P}$, we define the *optimal cost function* $H^* : V^2 \to \mathbb{R}_+ \cup \{0\}$, where $H^*(u, v)$ is the length of a shortest path from a node $u \in V$ to a node $v \in V$. That is,

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

We assume that an agent has a *heuristic function* $H(u, v)$ that estimates $H^*(u, v)$ for any two nodes $u, v \in V$.

We define the following two properties for a heuristic function. *Admissibility* indicates that a heuristic function never overestimates the optimal cost function.

Definition 3 (Admissibility). *Let* $\mathcal{P}$ *be a path planning problem and* H *be a heuristic function. Given a node* $v \in V$ *on* $\mathcal{P}$*, we say that* H *is* admissible with respect to v in $\mathcal{P}$*, if for any node* $u \in V$*,*

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

We also say that H *is* admissible in $\mathcal{P}$*, if for any node* $v \in V$*,* H *is admissible with respect to* v *in* $\mathcal{P}$*.*

Consistency indicates that the triangle inequality holds over a heuristic function.

Definition 4 (Consistency). *Let* $\mathcal{P}$ *be a path planning problem and* H *be a heuristic function. Given a node* $v \in V$ *on* $\mathcal{P}$*, we say that* H *is* consistent with respect to v in $\mathcal{P}$*, if*

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

and for any edge $(u, w) \in E$*,*

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

We also say that H *is* consistent in $\mathcal{P}$*, if for any node* $v \in V$*,* H *is consistent with respect to* v *in* $\mathcal{P}$*.*

It is well known that if a heuristic function H is consistent in a given path planning problem $\mathcal{P}$, then H is also admissible in $\mathcal{P}$.

3. ADAPTIVE A*

Koenig and Likhachev [7] proposed *Adaptive A**, an incremental version of A*. Adaptive A* solves series of similar search problems, updating its heuristic function between search episodes. Let $\mathcal{P}_t$ be a path planning problem in the t-th search episode. Adaptive A* can deal with change of C_t, such that a cost $C_t(e)$ may increase for any edge $e \in E_t$, i.e., $C_t(e) \leq C_{t+1}(e)$. The other elements of $\mathcal{P}_t$ are equivalent to the corresponding elements of $\mathcal{P}_{t-1}$.

Let us define H^t as a heuristic function in the t-th search episode. We use the notation of $h^t_\gamma(v) := H^t(v, \gamma)$ as a heuristic function from a node v to a goal γ in t-th search episode according to the custom.

A heuristic $h^{t+1}_\gamma(v)$ in $(t+1)$-th episode is calculated by the following update equation

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

where $g^t(v)$, which is called a *path-cost function*, is the actual minimum cost from the start node σ to a node v that expanded in t-th search episode. Koenig and Likhachev [8] proved that if h^0_γ is consistent with respect to γ in $\mathcal{P}_0$, then h^t_γ updated by Eq. (1) is also consistent with respect to γ in $\mathcal{P}_t$ for any $t \in \mathbb{N}$. Furthermore, they also proved that h^t_γ is monotonically nondecreasing with respect to t and thus indeed becomes more informed over time, that is, $h^t_\gamma(v) \leq h^{t+1}_\gamma(v)$ for any $t \in \mathbb{N}$.

Sun et al. [14] proposed *Generalized Adaptive A**, which can deal with the case that a cost $C_t(e)$ may decrease for any edge $e \in E$. In such a case, Generalized Adaptive A* uses a consistency procedure that eagerly updates h^t_γ with a version of Dijkstra's algorithm, so that h^{t+1}_γ keeps consistency.

There are several studies [5, 12, 11, 15] about a path planning problem with moving targets, where Γ_t can change. Koenig et al. [11] proposed *MT Adaptive A**, a version of Adaptive A*. When a target moves from γ^t to γ^{t+1} in the t-th search episode, MT Adaptive A* calculates $h^{t+1}_{\gamma^{t+1}}$ by the following update equation

$$h^{t+1}_{\gamma^{t+1}}(v) = \max\left\{ H^t(v, \gamma^{t+1}),\, h^t_{\gamma^t}(v) - h^t_{\gamma^t}(\gamma^{t+1}) \right\}, \tag{2}$$

where $H^{t+1}(v,\gamma^{t+1}) = h^{t+1}_{\gamma^{t+1}}(v)$ and $H^{t+1}(v,u) = H^t(v,u)$ for any $u \in V$ satisfying $u \neq \gamma^{t+1}$. They proved that if H^0 is consistent in $\mathcal{P}_0$ and $h_{\gamma^0}(v) := H^0(v,\gamma^0)$, then $h^t_{\gamma^t}$ updated by Eq. (2) is also consistent with respect to γ^t in $\mathcal{P}_t$ for any $t \in \mathbb{N}$.

4. OR SETTING

We consider *OR setting* in a given path planning problem $\mathcal{P}$. In this case, an agent must find a shortest path reaching the nearest target. We call it an *OR solution* and define as follows.

DEFINITION 5 (OR SOLUTION). *Given a path planning problem $\mathcal{P}$, we define that an* OR solution on $\mathcal{P}$ *is a shortest path from the initial node σ to the nearest node in the set Γ of goal nodes, i.e.,*

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

where $P_{or}(\sigma,\Gamma) := \bigcup_{\gamma\in\Gamma} P(\sigma,\gamma)$.

One naive approach is calculating the shortest paths to all goal nodes in Γ. Obviously, we need $|\Gamma|$ calculations of A*. In this section, we change the configuration of a heuristic function for single solution, so that only one calculation achieves an OR solution on $\mathcal{P}$, and discuss validity of it.

Given a path planning problem $\mathcal{P}$, we define the optimal cost function for OR solutions on $\mathcal{P}$ as the minimum cost of the OR solution, i.e.,

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

Let us consider a heuristic function h_{or} for OR solution on $\mathcal{P}$. Since h_{or} is the same form as h_γ for single solution with respect to γ, we can directly use A* for finding an OR solution.

We define admissibility and consistency of h_{or} in a similar way to single solution. We say that if for any node $v \in V$,

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

then h_{or} is *admissible with respect to OR solution in $\mathcal{P}$* and if for any $\gamma \in \Gamma$,

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

and for any edge $(u,v) \in E$,

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

then h_{or} is *consistent with respect to OR solution in $\mathcal{P}$*.

4.1 Stationary Target Search

We use the following heuristic h_{or} for OR solution, defined as

$$h_{or}(v) := \min_{\gamma\in\Gamma} h_\gamma(v). \tag{3}$$

Eq. (3) allows us to use an ordinary heuristic function h_γ for each goal γ, e.g., Manhattan distance. The next theorem proves its consistency.

THEOREM 6. *For any path planning problem $\mathcal{P}$, if h_γ is consistent with respect to γ in $\mathcal{P}$ for any $\gamma \in \Gamma$, then a heuristic function h_{or} defined in Eq. (3) is also consistent with respect to OR solution in $\mathcal{P}$.*

PROOF. For any $\gamma \in \Gamma$,

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

For any edge $(u,v) \in E$,

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

□

From the above theorem, we can easily apply Adaptive A* to a path planning problem with multiple targets of OR setting. The property of Adaptive A* immediately proves the following corollary.

COROLLARY 7. *Let $\mathcal{P}_t$ be an path planning problem in the t-th search episode and h^t_{or} be a heuristic function for OR solution in $\mathcal{P}_t$. If h_γ is consistent with respect to γ in $\mathcal{P}_0$ for any $\gamma \in \Gamma_0$, and h^0_{or} is initialized by h_{or} in Eq. (3), then h^t_{or} updated by Eq. (1) is also consistent with respect to OR solution in $\mathcal{P}_t$ for any $t \in \mathbb{N}$, and is monotonically nondecreasing with respect to t and thus indeed becomes more informed over time, that is, $h^t_{or}(v) \leq h^{t+1}_{or}(v)$ for any $t \in \mathbb{N}$.*

4.2 Moving Target Search

In the case where targets can move, we cannot directly use MT Adaptive A*. Instead of Eq. (2), we define the following update formula

$$h^{t+1}_{or}(v) = \max\left\{ \min_{\gamma\in\Gamma^{t+1}} H^t(v,\gamma),\ h^t_{or}(v) - \max_{\gamma\in\Gamma^{t+1}} h^t_{or}(\gamma) \right\}. \tag{4}$$

Note that $H^{t+1}(v,\gamma)$ is calculated by a similar fashion to the case of Eq. (2). The next theorem shows its validity.

THEOREM 8. *Let $\mathcal{P}_t$ be an path planning problem in the t-th search episode, H^t be a heuristic function in $\mathcal{P}_t$, and h^t_{or} be a heuristic function for OR solution in $\mathcal{P}_t$. If H^0 is consistent in $\mathcal{P}_0$ and $h_{\gamma^0}(v) := H^0(v,\gamma^0)$, then h^t_{or} updated by Eq. (4) is also consistent with respect to OR solution in $\mathcal{P}_t$ for any $t \in \mathbb{N}$.*

PROOF. When $t = 0$, h^0_{or} is obviously consistent with respect to OR solution in $\mathcal{P}^0$.

When $t > 0$, suppose that h^t_{or} is consistent with respect to OR solution in $\mathcal{P}^t$. From the assumption, H^t is also consistent in $\mathcal{P}^t$. Thus, for any $\gamma' \in \Gamma^{t+1}$,

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

and for any edge $(v,u) \in E$,

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

Therefore, h^{t+1}_{or} is consistent with respect to OR solution in $\mathcal{P}^{t+1}$.

By induction, for any integer $t \geq 1$, h^t_{or} is consistent in $\mathcal{P}^t$. □

5. AND SETTING

We consider *AND setting* where an agent must find a shortest path reaching all of the targets. We call it an *AND solution* and define as follows.

Definition 9 (AND Solution). *Given a path planning problem $\mathcal{P}$, we define that an* AND solution on $\mathcal{P}$ *is a shortest path starting from the initial node σ, traversing all of the nodes in the set Γ of goal nodes, i.e.,*

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

where $P_{and}(\sigma,\Gamma) := \bigcup_{\pi\in\Pi_n} P(\sigma,\gamma_{\pi[1]},\cdots,\gamma_{\pi[n]})$.

We cannot take a similar approach to OR solution for AND solution, since a node $v \in V$ cannot reserve which nodes an agent has already reached. In this section, we discuss three approaches for directly using Adaptive A*.

5.1 Straightforward Method

First we consider a straightforward method that calculates the optimal cost $H^*(\gamma_1,\gamma_2)$ between any two goals $\gamma_1,\gamma_2 \in \Gamma$ in a given path planning problem $\mathcal{P}$ and a shortest Hamiltonian path on a corresponding graph G' with weight of the calculated costs. We call G' an *abstract graph* and formalize it as follows.

Definition 10 (Abstract Graph). *Given a path planning problem $\mathcal{P}$ and a heuristic function H on $\mathcal{P}$, we define the* abstract graph of $\mathcal{P}$ with respect to H *by*

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

where $V' := \Gamma \cup \{\sigma\}$, $E' := \{(u,v) \mid u,v \in V'\}$, *and* $C'(u,v) := H(u,v)$ *for any* $u,v \in V'$.

input : Path planning problem $\mathcal{P}$ and heuristic function H
output: AND solution on $\mathcal{P}$

1 **foreach** $u,v \in \Gamma \cup \{\sigma\}$ **do**
2 $Q(u,v) \leftarrow$ a shortest path from u to v calculated by A* based on H;
3 $H^*(u,v) \leftarrow C(Q(u,v))$;
4 **end**
5 $p^*_{shp} \leftarrow$ a shortest Hamiltonian path from σ on the abstract graph G' of $\mathcal{P}$ with respect to H^*;
6 $p^*_{and} \leftarrow (Q(p^*_{shp}[1]),\cdots,Q(p^*_{shp}[n]))$;
7 **return** p^*_{and};

Algorithm 1: StrPlan

Given a path planning problem $\mathcal{P}$ and a heuristic function H on $\mathcal{P}$, we calculate an AND solution on $\mathcal{P}$ by creating the abstract graph G' of $\mathcal{P}$ and executing StrPlan algorithm shown in Algorithm 1. One may think that the algorithm correctly returns an AND solution on $\mathcal{P}$. However, a shortest Hamiltonian path from σ on G' visits each goal node $\gamma \in \Gamma$ exactly once, while an AND solution on $\mathcal{P}$ may visit each goal node $\gamma \in \Gamma$ more than once. We should concern about that for a calculated shortest Hamiltonian path p^*_{shp} of G' such that $p^*_{shp}[i] = (u_i, v_i) \in V'$, there is a possibility that the last goal node v_n is on a shortest path $Q(p^*_{shp}[i])$ from u_i to v_i on G for some $i \in [1, n-1]$. At a first look, you may suspect that StrPlan algorithm sometimes returns a wrong answer, since a path $p'_{and} := (Q(p^*_{shp}[1]),\cdots,Q(p^*_{shp}[n-1]))$ allows an agent to reach all goals. However, the next theorem guarantees that our algorithm is correct indeed.

Theorem 11. *For any path planning problem $\mathcal{P}$, the cost of an AND solution p^*_{and} on $\mathcal{P}$ is equivalent to that of a shortest Hamiltonian path p^*_{shp} from σ on the abstract graph $G' = (V', E', C')$ of $\mathcal{P}$ with respect to the optimal cost function H^*, that is,*

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

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 algorithm calls calculation of a shortest path on $\mathcal{P}$ exactly $n(n+1)/2$ times, where $n = |\Gamma|$, and calculation of a shortest Hamiltonian path on G' once. Let $ASTAR$ and SHP be the worst case time complexity of the former and the later, respectively. That is, $ASTAR = O(l \log m)$ with a consistent heuristic function, where $l = |E|$ and $m = |V|$, and $SHP = O(n^2 2^n)$. Therefore, the worst case time complexity of StrPlan algorithm is $O(n^2 ASTAR + SHP) = O(n^2(l \log m + 2^n))$.

5.2 Incremental Method

Next we consider an incremental method that repeats calculation of a shortest Hamiltonian path on the abstract graph G' of a given path planning problem $\mathcal{P}$ with respect to a heuristic function H and update of H as shown in Algorithm 2. We call the algorithm IncPlan.

Line 4 in IncPlan algorithm means the termination condition. The algorithm clearly terminates after at most $|F|$ iterations. In the case that $F = \emptyset$ holds, Theorem 11 also leads validity of IncPlan algorithm, since all of the minimum costs between any two goals are calculated. In the case that $p_{shp}[i] \notin F$ for any $i \in [1, n]$ holds, IncPlan algorithm always returns a correct answer from the following theorem.

Theorem 12. *Let $G = (V, E, C)$ and $G' = (V, E, C')$ be weighted graphs with the same sets of nodes and edges. For any node $v \in V$, letting p be a shortest Hamiltonian path from v on G, if $C(p[i]) = C'(p[i])$ for any $i \in [1, |p|]$ and $C(e) \le C'(e)$ for any $e \in E$, then p is also a shortest Hamiltonian path from v on G', that is,*

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

where $P_{inc}(V) := \bigcup_{\pi\in\Pi_{|p|}} P(v_{\pi[1]},\cdots,v_{\pi[|p|]})$, *and v_i represents the i-th node in V after arbitrarily ordering.*

Proof. From $C(p[i]) = C'(p[i])$ for any $i \in [1, |p|]$,

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

input : Path planning problem $\mathcal{P}$ and heuristic function H
output: AND solution on $\mathcal{P}$

$F \leftarrow (\Gamma \cup \{\sigma\}) \times \Gamma$;
while *true* **do**
  $p_{shp} \leftarrow$ a shortest Hamiltonian path from σ on the abstract graph G' of $\mathcal{P}$ with respect to H;
  if $F = \emptyset$ *or* $\forall i \in [1,n], p_{shp}[i] \notin F$ **then** break;
  foreach (u,v) *in* p_{shp} **do**
    if $(u,v) \in F$ **then**
      $Q(u,v) \leftarrow$ a shortest path from u to v calculated by A* based on H;
      $H(u,v) \leftarrow C(Q(u,v))$;
      $F \leftarrow F - \{(u,v)\}$;
    end
  end
end
$p^*_{and} \leftarrow (Q(p_{shp}[1]), \cdots, Q(p_{shp}[n]))$;
return p^*_{and};

Algorithm 2: IncPlan

From $C(e) \le C'(e)$ for any $e \in E$,

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

□

IncPlan algorithm calls calculation of a shortest path on $\mathcal{P}$ at most $n(n+1)/2$ times and calculation of a shortest Hamiltonian path on G' at most $(n^2-n+2)/2$ times. Therefore, the worst case time complexity of IncPlan algorithm is $O(n^2 ASTAR + n^2 SHP) = O(n^2(l \log m + n^2 2^n))$.

5.3 Conversion Method

Finally we consider an conversion method that converts a given path planning problem $\mathcal{P}$ to another problem $\mathcal{P}'$ so that we can solve it using A* only once. The idea of the conversion is to attach a binary $b \in B_n$ to each node $v \in V$, where $b[i]$ represents whether an agent has already visited a goal node $\gamma_i \in \Gamma$. A set Γ' of converted goal nodes consists of all goal nodes in Γ with 1^n. An agent has only to find an OR solution on $\mathcal{P}'$ for finding an AND solution on $\mathcal{P}$. We formalize the conversion as follows.

Definition 13 (OR Conversion). *Given a path planning problem $\mathcal{P}$, we define the* OR conversion of $\mathcal{P}$ *by*

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

such that $G' := (V', E', C')$, where

- $V' := \{ (v,b) \mid v \in V, b \in B_n, \phi(v,b) = 1 \}$, *where*

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

- $E' := \{ (u',v') \mid u', v' \in V', \psi(u',v') = 1 \}$, *where*

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

  such that $u' = (u, b_u)$ and $v' = (v, b_v)$.

- $C'(e') := C(e)$ *for any $e' \in E'$.*
- $\sigma' := (\sigma, 0^n)$.
- $\Gamma' := \{ (\gamma, b) \in V' \mid \gamma \in \Gamma, b = 1^n \}$.

We prove the following lemma for Theorem 15 that shows validity of the OR conversion.

Lemma 14. *For any path planning problem $\mathcal{P}$, the set $P_{and}(\sigma, \Gamma)$ for AND solution on $\mathcal{P}$ is equivalent to the set*

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

with respect to the set $P_{or}(\sigma', \Gamma')$ for OR solution on the OR conversion $\mathcal{P}' = (G', \sigma', \Gamma')$ of $\mathcal{P}$. That is,

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

Proof. ($\Rightarrow$) Suppose that $p \in P_{and}(\sigma, \Gamma)$, letting $p[i] := (u_i, v_i)$. We can construct a path p' such that $p'[i] := ((u_i, b_{u,i}), (v_i, b_{v,i}))$, where

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

and

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

By the definition of the OR conversion, we have $p' \in P_{or}(\sigma', \Gamma')$. Therefore, $p \in \hat{P}_{or}(\sigma', \Gamma')$.

($\Leftarrow$) Suppose that $\hat{p} \in \hat{P}_{or}(\sigma', \Gamma')$, letting $\hat{p}[i] := (\hat{u}_i, \hat{v}_i)$. By the definition of $\hat{P}_{or}(\sigma', \Gamma')$, there exists $p' \in P_{or}(\sigma', \Gamma')$, such that $\hat{u}_i = u'_i$ and $\hat{v}_i = v'_i$, where $p'[i] := ((u'_i, b'_{u,i}), (v'_i, b'_{v,i}))$. From the definition of OR conversion, we have $b'_{u,1} = 0^n$ and $b'_{v,n} = 1^n$. That is, $\hat{u}_i = \sigma$ and $\hat{v}_n = \gamma \in \Gamma$. From the definition of ψ in OR conversion, for any goal node $\gamma \in \Gamma$, there exists i such that $\hat{v}_i = \gamma$. Therefore, $\hat{p} \in P_{and}(\sigma, \Gamma)$. □

The next theorem shows that finding an AND solution on $\mathcal{P}$ is surely equivalent to finding an OR solution of the OR conversion $\mathcal{P}'$ of $\mathcal{P}$.

Theorem 15. *For any path planning problem $\mathcal{P}$, the cost of an AND solution p^*_{and} on $\mathcal{P}$ is equivalent to that of an OR solution p^*_{or} on the OR conversion of $\mathcal{P}$, that is,*

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

Proof. From Lemma 14,

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

□

When we set up a heuristic function for OR solution on $\mathcal{P}'$, we cannot take a trivial approach such as Manhattan distance on $\mathcal{P}$. The next theorem shows how to construct a consistent heuristic function for OR solution on $\mathcal{P}'$ by using one for single solution on $\mathcal{P}$.

THEOREM 16. *Let* $\max_0$ *and* $\min_0$ *be functions returning* $\max S$ *and* $\min S$ *if* $S \neq \emptyset$*, respectively, zero otherwise. For any path planning problem* $\mathcal{P}$*, letting* $\mathcal{P}'$ *be the OR conversion of* $\mathcal{P}$*, if a heuristic function* H *is consistent in* $\mathcal{P}$*, then a heuristic function* h_{cnv} *for OR solution on* $\mathcal{P}'$ *defined by*

$$h_{cnv}(v') := \max \left\{ \begin{array}{c} \max_0 \limits_{\gamma \in R(b_v)} H(v, \gamma), \\ \min_0 \limits_{\gamma \in R(b_v)} H(v, \gamma) + k(\mid R(b_v) \mid -1) \end{array} \right\}$$

is also consistent with respect to OR solution on $\mathcal{P}'$*, where* $v' := (v, b_v)$*,* $R(b) := \{ \gamma_i \in \Gamma \mid b[i] = 0 \}$*, and* $k := \min_{e \in E} C(e)$.

PROOF. When $v' \in \Gamma'$, clearly $R(b) = \emptyset$. Therefore,

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

Next we show that the triangle inequality holds for h_{cnv}. Let $(v', u') \in E'$ be an edge such that $v' := (v, b_v)$ and $u' := (u, b_u)$. Since H is consistent in $\mathcal{P}$ and $(v, u) \in E$, we have

$$h_{cnv}(v') \leq C(v,u) + \max \left\{ \begin{array}{c} \max_0 \limits_{\gamma \in R(b_v)} H(u, \gamma), \\ \min_0 \limits_{\gamma \in R(b_v)} H(u, \gamma) + k(\mid R(b_v) \mid -1) \end{array} \right\}.$$

When $v' \in \Gamma'$, obviously $h_{cnv}(v') = 0 \leq C'(u', v') + h_{cnv}(v')$.

When $v' \notin \Gamma'$ and $u \notin \Gamma$, we have $b_v = b_u$ from the definition of ψ in OR conversion. That is, $R(b_v) = R(b_u)$. Therefore, $h_{cnv}(v') \leq C(v,u) + h_{cnv}(u') = C'(v', u') + h_{cnv}(u')$.

When $v' \notin \Gamma'$ and $u \in \Gamma$, consistency of H leads $H(u, u) = 0$. By the definition of ψ, we have $R(b_v) = R(b_u) \cup \{u\}$. The first term of max operation is

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

The second term of max operation is

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

since $H(u, \gamma) \geq k$ with $u \neq \gamma$ and $k|\emptyset| = 0$. Therefore, $h_{cnv}(v') \leq C'(v', u') + h_{cnv}(u')$. □

Note that the search space of $\mathcal{P}'$ becomes larger than that of $\mathcal{P}$, although A* is called only once in the conversion method. The worst case time complexity of the conversion method is $O(|E'| \log |V'|) = O(2^n |E| \log(2^n |V|)) = O(2^n l \log(2^n m))$.

6. EXPERIMENTS

We consider path planning problem $\mathcal{P}$ on a 100×100 maze shown in Fig. 1, as an application of the proposed methods. G is defined as a two dimensional, undirected, grid graph. For any edge $(u, v) \in E$, we set $C(u, v) = 1$ if both u and v are unblocked, $C(u, v) = \infty$ otherwise. $C(u, v)$ changes with a fixed probability $\alpha \in \mathbb{R} : 0 \leq \alpha \leq 1$, where a node $v \in V$ changes either from blocked to unblocked or from unblocked to blocked. Each target also moves to an adjacent node with a fixed probability $\beta \in \mathbb{R} : 0 \leq \beta \leq 1$, where the corresponding goal node $\gamma \in \Gamma$ becomes the adjacent node.

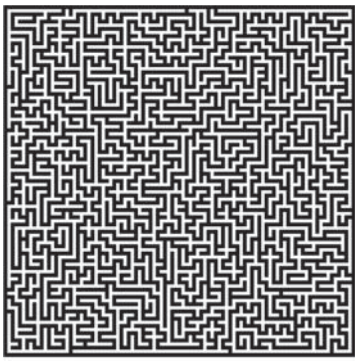

Figure 1: Maze Problem

After each trial is over, which means that an agent achieves a solution, both an initial node σ of the agent and a set Γ of goal nodes are randomly reset. We also consider the case where the agent does not know the map of a given maze beforehand, where the agent can only observe information (i.e., either blocked or unblocked) in its own node and the adjacent four nodes and remember them in each trial. We utilize Generalized Adaptive A* [14], the latest version of Adaptive A* in all experiments. We denote the number of targets by n.

6.1 OR Setting

Table 1 shows (a) the total number of calculations of optimal path in each trial, (b) the total movements of the agent in each trial, (c) the total number of node expansions in A* algorithm in each trial, (d) the total runtime in each trial, for both known and unknown maze settings in experiments on a maze problem with stationary targets of OR setting. NaivePlan represents the naive method calculating the shortest paths to all goal nodes. MinPlan represents the proposed method utilizing a heuristic function in Eq. (3). We set $\alpha = 0.1$. Each value is the average in 500 trials. Table 2 shows experimental results in a problem with moving targets in a similar way in Table 1. We set $\beta = 0.1$.

Comparing NaivePlan and MinPlan, all results show that MinPlan is clearly better than NaivePlan. Looking at (b), since the total movements of the agent means the length of a shortest path, the results of MinPlan and NaivePlan are almost the same, although they do not exactly identical due to probabilistic change of costs and movement of targets. Since NaivePlan calls Adaptive A* n times, the results in (a), (c) and (d) of NaivePlan increase, as n increases. Contrary to our intuition, all results of MinPlan and the results in (b) of NaivePlan decrease, as n increases. That is because the distance to the nearest target becomes smaller, as n increases.

6.2 AND Setting

Table 3 shows experimental results on a maze problem with stationary targets of AND setting, in a similar way in Table 1. We set $\alpha = 0$ and $\beta = 0$. We add (e) the total calls of A* in each trial and (f) the total number of calculations of a shortest Hamiltonian path in each trial. In this experiment, each value is the average in 100 trials. StrPlan, IncPlan, and CnvPlan represent the proposed methods in Section 5.1, Section 5.2, and Section 5.3, respectively. The worst case time complexities of StrPlan, IncPlan,

Table 1: OR Setting with Stationary Targets

	Known Maze				Unknown Maze			
	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
number of targets	searches per trial	moves per trial	expansions per trial	runtime [ms] per trial	searches per trial	moves per trial	expansions per trial	runtime [ms] per trial
	NaivePlan							
5	54.81	98.99	87369.62	20.29	768.84	432.97	1612872.98	591.43
10	75.60	66.46	137012.91	27.98	1012.56	276.79	2797733.59	1120.27
15	90.78	51.66	173474.40	34.32	1022.52	181.77	3157937.16	1264.39
	MinPlan							
5	10.96	98.99	2373.26	1.08	149.63	422.64	7502.98	31.30
10	7.56	66.46	1040.69	0.61	105.74	291.22	4121.48	21.57
15	6.05	51.66	573.52	0.43	71.81	194.98	2119.70	14.08

Table 2: OR Setting with Moving Targets

	Known Maze				Unknown Maze			
	(a)	(b)	(c)	(d)	(a)	(b)	(c)	(d)
number of targets	searches per trial	moves per trial	expansions per trial	runtime [ms] per trial	searches per trial	moves per trial	expansions per trial	runtime [ms] per trial
	NaivePlan							
5	232.19	97.77	397006.24	97.40	1345.01	434.88	2880903.98	1195.86
10	446.76	64.90	836500.39	236.71	2129.80	273.57	6092105.11	2652.74
15	594.84	48.58	1116453.08	349.50	2672.34	203.92	8377218.38	3589.50
	MinPlan							
5	46.44	97.77	9499.80	3.86	279.51	453.41	18097.55	23.49
10	44.43	64.35	5005.14	2.90	200.26	257.62	8428.36	13.14
15	42.72	52.24	3425.08	2.62	163.97	188.48	5520.92	9.98

and CnvPlan are $O(n^2(l \log m + 2^n))$, $O(n^2(l \log m + n^2 2^n))$, and $O(2^n l \log(2^n m))$, respectively, where n is the number of targets, l is the number of edges, and m is the number of nodes.

The results in (b) show that all of the values, each of which means the length of a shortest path, are exactly the same, since $\alpha = 0$ and $\beta = 0$. Focusing on the results in (d) of CnvPlan, each running time is quite bigger than the other methods in the same tendency of its worst case time complexity when $m \gg n$. On the other hand, the results in (d) of IncPlan is better than that of StrPlan when $n \leq 8$, despite that the worst case time complexity of IncPlan is worse than that of StrPlan. The result indicates that IncPlan is practically efficient than StrPlan when n is relatively small. That is because that IncPlan executes Adaptive A* fewer times than StrPlan in average case, as shown in the results in (e). In actual situation such as Pac-Man, n is often not so large. Actually, there exist only four ghosts in the Pac-Man game as described previously. We conclude that IncPlan is practically better than StrPlan, when n is relatively small.

7. CONCLUSIONS

We addressed path planning with multiple targets on recently developed Adaptive A* framework. We formalized OR and AND settings, whose objectives are (1) an agent has to reach one of the targets, and (2) an agent has to reach all of the targets, respectively. For OR setting, we proposed a construction method of a consistent heuristic function to utilize Adaptive A*, keeping its consistency for both stationary and moving targets. For AND setting, we proposed three methods to directly utilize Adaptive A* for each target. We also proved that all of the methods always achieve an optimal path of AND setting. Our experimental results showed that the proposed methods properly work on an application, i.e., maze problems, both in OR and in AND settings.

8. REFERENCES

[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)*, pages 1047–1052. IEEE, 2007.

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

Table 3: AND Setting

number of targets	(a) searches per trial	(b) moves per trial	(c) expansions per trial	(d) runtime [ms] per trial	(e) A* per trial	(f) SHP per trial
			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

Expert Systems (IEA/AIE-98), volume 1416 of *Lecture Notes in Computer Science*, pages 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*, 17(6):609–619, 1995.

[6] Sven Koenig and Maxim Likhachev. D* Lite. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 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)*, pages 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)*, pages 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)*, pages 281–288. ACM, 2006.

[10] Sven Koenig, Maxim Likhachev, and David Furcy. Lifelong Planning A*. *Artificial Intelligence*, pages 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)*, pages 1136–1143. IFAAMAS, 2007.

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

[13] 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)*, pages 2391–2397, 2007.

[14] 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)*, pages 469–476. IFAAMAS, 2008.

[15] 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)*, pages 615–620, 2009.

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

[17] 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)*, pages 855–861. IEEE, 2004.