

Masashi Shimbo

Real-Time Search with Nonstandard Heuristics

This thesis was submitted to
Faculty of Engineering, Kyoto University in December 1999
in fulfillment of the requirements for
the degree of Doctor of Engineering.

Doctoral Thesis Series of Ishida Laboratory
Department of Social Informatics
Kyoto University

Copyright © 1999 Masashi Shimbo

Abstract

Real-time heuristic search adopts a problem-solving model significantly different from traditional AI search. As opposed to these traditional *off-line search* methods devoted to *batch* global planning, real-time search handles problem solving as a whole, by interleaving partial planning and action execution. It performs a series of real-time planning/action execution cycle, each time taking into account the new surroundings it is facing. This distinctive feature of “the problem solver being *situated* in the environment” makes real-time search an attractive solution for problems involving uncertainty or dynamics, as it allows the problem solver to adapt itself to changing situations in a way which is never possible for off-line search.

In this thesis, we study various aspects of real-time search, with emphasis on two issues not previously pursued: the utility of nonstandard heuristic function and the analysis of convergence process.

The first issue addresses the effect of nonstandard heuristic functions on real-time search. In particular, we study the heuristic functions violating two properties that have been taken for granted: *admissibility* and *consistency*. Intuitively, admissibility is the property that the function never overestimates the actual cost, while consistency states that the values assigned by the function to adjacent states satisfy a form of triangle inequality. Most of the research efforts in real-time search have been directed towards the study of heuristic function satisfying these properties.

The second issue is concerned with the ability to *learn from experience*, enjoyed by R. Korf’s LRTA* and some other real-time search algorithms.

Even though real-time search algorithms generally cannot find an optimal action sequence in the first problem solving trial, these algorithms will eventually find, or learn, optimal solutions through repeated trials. The theorem of such asymptotic convergence has already been obtained, but it does not answer the question of in what way the convergence progresses.

Through the research on the following subtopics, we show that these seemingly independent issues are actually closely related; the use of nonstandard heuristics provides a remedy for the problems concerning the learning process.

1. A new technique for proving the convergence.

We present a new technique for proving the convergence of $LRTA^*$ algorithm. As the obtained proof is built upon the same lemmas as the existing proof of *completeness*, it constitutes the connection between these important properties of $LRTA^*$ that have been discussed somewhat separately.

2. Analysis of real-time search with overestimated heuristics.

We study the performance of *weighted $LRTA^*$* algorithm, which uses an inadmissible, or overestimated, heuristic function. We prove that although weighted $LRTA^*$ does not generally converge to optimal solutions, it does show a form of weaker convergence to suboptimal solutions. We also show that it is possible to predict from the amount of the overestimation of the initial heuristic function how much the quality of the converged solution may deteriorate.

3. A methodology for overcoming the instability of convergence process.

We point out the drawbacks of $LRTA^*$ algorithm during convergence: the failure to adequately balance the amount of exploration and exploitation, and the incurred instability of solution quality during convergence. To overcome the drawback, we propose a new search method that uses an inadmissible heuristic function for maintaining *upper bounds* of the actual cost. This new heuristic function is used along with the conventional admissible function for providing lower bounds. We

demonstrate that compared to LRTA^{*}, the algorithm guarantees more stable improvement of solution quality during convergence. With the algorithm, we obtain fine-grained control of the convergence process without losing the convergence to optimality.

4. Analysis of the effect of inconsistent heuristics on the moving-target search.

An interesting application of real-time search is the *moving-target search* (MTS) problem. In MTS setting, the goal changes during the course of the problem solving. We resolve the difference between LRTA^{*} and the MTS algorithm due to Ishida and Korf, on the use of inconsistent heuristic function. In particular, we show that the MTS algorithm is complete as well, even under inadmissible (and thereby inconsistent) heuristic function. This result contributes to extending the class of heuristic functions usable with moving-target search.

Acknowledgments

I would like to express my sincere gratitude to Professor Toru Ishida for taking the burden of supervising this thesis. He has always been helpful and generous, despite my occasional disobedience which I now deeply regret. Without his patient and continuous guidance, I would not have been able to accomplish this thesis.

I would also like to thank other members of my thesis committee, Professor Kazuo Iwama and Professor Shinji Tomita, for their valuable comments and criticism. Professor Tomita was also the director of the Center for Information and Multimedia Studies (CIMS), Kyoto University, where I completed some of the work included in the thesis.

I had a lot of support from all my colleagues at Professor Ishida's research group. Especially, Hirofumi Yamaki helped me to prepare the manuscript; Teruhisa Miura and Takayuki Yoshizumi computed the optimal solution cost of the fifteen-puzzle with their IDA* program for me; the robot drawing in Chapter 2 was created by Goichi Tanaka, which was originally inspired by the version by Professor Hiroshi Ishiguro.

I am deeply indebted to Professor Masatake Dantsuji for his encouragement, and for his kind and timely advice which really saved me out of trouble.

Professor Yuji Matsumoto gave me insightful comments on some of the material of this thesis. He was my advisor when I was an undergraduate student, and still is my mentor.

Masahiko Haruno has been a good friend of mine for almost ten years. He never fails to entertain me, not only because of his extensive knowledge of

machine learning, but also because of his superior sense of (sarcastic) humor. He also gave me a hint on how to correct a flaw of a proof included in the thesis.

Thanks are also due to my present colleagues at Ibaraki University for their support; in particular, Professor Yoshiki Kishi and Professor Tatsuhiro Yonekura kindly offered their computer facilities to me at the very day of my arrival at the university.

Last but not least, I would like to thank my loving family; my parents Masaru and Yayoe for their trust; my wife Hiroko for her devoted support and for enduring all the inconvenience incurred by my preoccupation in writing; and my son Atsushi, for making my life truly meaningful.

Contents

Abstract	i
Acknowledgments	v
Contents	vii
List of Figures	xi
List of Tables	xiii
1 Introduction	1
1.1 Background	1
1.2 Objectives	4
1.3 Outline of the Thesis	5
2 Preliminaries	9
2.1 State Space	9
2.2 Heuristics	11
2.3 The LRTA* Algorithm	13
2.4 Properties of Real-Time Search Algorithms	16
3 Convergence of Real-Time Search	21
3.1 Introduction	21
3.2 Preliminaries	22
3.2.1 Completeness Proof by Ishida and Korf	23

3.2.2	Original Convergence Proof by Korf	26
3.3	Convergence	29
3.3.1	Convergence of the Trajectory	30
3.3.2	Convergence of the Heuristic Estimates	33
3.4	Variations	35
3.4.1	Alternative Reset Model with Deadlines	35
3.4.2	Inadmissible Initial Heuristic Function	37
3.5	Discussion	37
3.6	Summary	39
4	Weighted Real-Time Search for Tolerating Suboptimal Solutions	41
4.1	Introduction	41
4.2	Preliminaries	42
4.3	Weighted LRTA*	43
4.4	Completeness of Weighted LRTA*	44
4.5	Convergence of Weighted LRTA*	46
4.6	Experiments	48
4.6.1	Single-Trial Problem Solving	48
4.6.2	Repeated Problem Solving Trials	49
4.7	Variations	51
4.7.1	Use of Naive Value-Update Rule	51
4.7.2	Measuring the Error Additively	53
4.8	Discussion	55
4.9	Summary	56
5	Controlling the Convergence Process	57
5.1	Introduction	57
5.2	Motivation	59
5.2.1	Learning Performance of LRTA*	59
5.2.2	Effect of Weighted LRTA* on Solution Stability	62
5.3	Preliminaries	65
5.4	Real-Time Search with Upper Bounds	65

5.4.1	Introducing Upper-Bound Heuristics	65
5.4.2	The δ -Search Algorithm	66
5.5	Properties of δ -Search	68
5.6	Experiments	72
5.7	Discussion	75
5.7.1	Multiple Initial and Goal States	75
5.7.2	State Space with Irreversible Actions	76
5.7.3	Note on Combining δ -Search and Weighted LRTA*	80
5.8	Summary	80
6	Completeness of the Moving-Target Search When Heuristics Overestimate	81
6.1	Introduction	81
6.2	Moving-Target Search	82
6.2.1	Informal Description of the Problem	82
6.2.2	Formal Description of the Problem	83
6.2.3	Modeling the Speed of Agents	85
6.2.4	The MTS Algorithm	86
6.3	Completeness of the MTS Algorithm	89
6.4	Related Work	93
6.5	Summary	93
7	Conclusions	95
7.1	Summary	95
7.2	Future Directions	97
	Bibliography	99

List of Figures

2.1	Pseudo-code of LRTA* algorithm	14
2.2	Robot navigated by LRTA* at a crossing	16
2.3	LRTA*: look ahead	17
2.4	LRTA*: value update	17
2.5	LRTA*: move	18
3.1	State space with inconsistent heuristic estimates	29
4.1	Workbench: fifteen-puzzle	49
4.2	Single-trial ϵ -LRTA* in fifteen-puzzle: number of moves	50
5.1	Gridworld with 35% random obstacles	60
5.2	LRTA* in gridworld: a typical episode	61
5.3	Repeated ϵ -LRTA* in gridworld: number of moves	63
5.4	Repeated ϵ -LRTA* in gridworld: number of expanded states	64
5.5	Pseudo-code of δ -search algorithm	67
5.6	δ -search in gridworld: number of moves	73
5.7	δ -search in gridworld: number of expanded states	74
5.8	$\epsilon\delta$ -search in gridworld: a typical episode	75
5.9	State space with irreversible actions	77
5.10	δ -search in the state space with irreversible actions	79
6.1	Pseudo-code of the Moving-Target Search algorithm	87

List of Tables

3.1	Episode in the state space with inconsistent estimates	30
4.1	Repeated ϵ -LRTA* in fifteen-puzzle: results	51

Chapter 1

Introduction

1.1 Background

Since 1960's, the *heuristic search* has been, and basically still is, one of the central issues in artificial intelligence (AI). Beginning with A* [10, 28], a number of heuristic search algorithms have been proposed. To solve a specific problem with these classic heuristic search methods, it is assumed that the problem solving task is decomposable into the following subtasks.

1. Convert the process description of the problem into state description [39].
2. Translate the *heuristic*, or the prior knowledge specific to the domain that helps to solve the problem, into the form of heuristic evaluation function.
3. Apply a suitable heuristic search algorithm to generate a plan to achieve goal (*planning*).
4. Commit to the generated plan (*action execution*).

From the standpoint of this framework, problem solving is intrinsically a batch process in which the planning and the commitment to the generated

plan are independent processes. Adopting the framework, most research efforts have been directed towards the development of the methods to find a complete (and in particular, optimal) action sequence to the goal; this is what classic, or “off-line” search algorithms are specialized for, confining themselves solely to the planning (Step 3).

Off-line search algorithms have seen success in many application domains, partly because of the generality and simplicity of the framework mentioned above. However, there are domains in which the framework fails to fit at all; such inadequacy stems from its being a batch process too centric on planning. To be more precise, they construct a complete action sequence even before the problem solver makes its first move. Since such construction often results in exhaustive search that takes a long time, there is no room left for taking the dynamics of the problem into account. In addition, it requires the complete model of the problem to be available before the first action. In some problems that involve uncertainty, it is more natural to assume that the precise model is available only after making specific actions at each stage. The problem is that the search algorithms rely so much on the assumption of static state space that it is quite difficult to build into them the means to cope with uncertainty or the changes that might happen to the state space during the search.

The new class of heuristic search algorithms, coined as “real-time search” by its inventor R. Korf, arose out of such observation. This new class includes Real-Time A* (RTA*) and Learning Real-Time A* (LRTA*) [23] and is more suited to searching in environments that involve uncertainty or dynamics. To deal with these factors, the real-time search adopts a framework significantly different from that of off-line search; it is not merely a planner in the sense of off-line search, but is rather a complete problem-solving architecture in which planning and action execution are interleaved. The problem solver is *situated* in the environment and is *reactive*, in the sense that it does not wait for a complete plan to be made up, but instead commits to an action based on the information gathered by local search (partial planning). After making an action, the problem solver observes its new surrounding situation, and repeats

the cycle of partial planning and action execution until it finally reaches the goal. This is reminiscent of two-player game search [29] that incorporates local search to handle limited time and resources available to each player. The resulting search algorithm is more flexible, featuring the abilities to perform planning and acting in turn or even in parallel, control the amount of planning based on the circumstances, or adapt itself to changing situations which is never possible for off-line search. A number of real-time search algorithms have been proposed [6, 7, 12, 13, 15, 30, 35] to take advantage of these unique features, and the range of application is growing even wider.

Another significant feature that LRTA* and some other real-time search algorithms enjoy is the *convergence* to optimal solutions; Although real-time search in general do not assure us to find an optimal action sequence, these algorithms have the property that if they are applied to the similar problems repeatedly, the action sequence executed by the problem solver will eventually converge to an optimal one. This property demonstrates that these algorithms have the ability to *learn from their experience*.

This learning ability of real-time search suggests its strong relationship with other machine learning algorithms. For instance, the similarity between LRTA* and reinforcement learning [40], especially Q-learning [18, 41], has been investigated by Barto et al. [2]. Reinforcement learning was originally intended to model an aspect of the humans' information processing mechanism, namely the adaptive learning through trial-and-error. It is also possible to view real-time search in the same way, which makes it interesting as a study model of autonomous agents with adaptive behavior.

An interesting topic concerning the learning through repeated problem solving is the so-called "exploration-exploitation trade-off." It stems from the fact that the algorithms have to achieve two often contradicting goals at the same time: (1) to reach goal safely (in each problem solving trial) and (2) to find better solution (through the trials). To achieve the first goal one has to make more use of the information it has collected before (exploitation), while to achieve the second requires eager trial of new actions (exploration) at the risk of not being able to arrive at a goal easily. The challenge is to

find a way to balance these in some decent ways.

1.2 Objectives

In this thesis, we study various aspects of real-time heuristic search, with emphasis on two issues not previously pursued:

1. the analysis of the effect of nonstandard heuristic function in real-time search, and
2. the analysis of learning process.

The first issue addresses the effect of nonstandard heuristic functions on real-time search, in particular, those violating two properties that have been taken for granted: *admissibility* and *consistency*. Intuitively, admissibility is the property that the function never overestimates the actual cost, while consistency states that the values assigned to neighboring states satisfy a form of triangle inequality. Most of the research efforts in real-time search have been directed towards the study of heuristic function with these properties, and little is known on the effect of the heuristics that are nonconforming to these criteria. In off-line search, on the other hand, the inadmissible heuristic function is often used to find near-optimal solutions, especially for the problems that are inherently intractable to seek for optimal solutions. Although most single-trial real-time search algorithms proposed so far are basically suboptimal algorithms that do not suffer from such intractability, they lack the nontrivial guarantee on the obtained solution quality to be usable for finding near-optimal solutions.

The second issue addressed by the thesis is concerned with the ability to learn from experience, enjoyed by some real-time search algorithms including LRTA*. As mentioned earlier, these algorithms are known to eventually find optimal solutions through repeated problem solving trials, but the mere fact that they eventually converge does not answer the question of how such convergence progresses. Though the learning ability of these algorithms is quite intriguing from the perspective of modeling adaptive autonomous agents, it

has not been paid much attention compared to their single-trial performance.

To achieve these goals, the thesis deals with the following topics.

1. Development of the comprehensive theory that accounts for both completeness and convergence in a unified manner, which is applicable regardless of the consistency of heuristic function.
2. Performance analysis of real-time search with inadmissible (and thereby inconsistent) heuristic function.
3. Development of the methodology for overcoming the instability of solution quality during convergence.
4. Resolution of difference between real-time search for stationary goals and moving-target search upon the effect of inconsistent heuristics.

1.3 Outline of the Thesis

The rest of the thesis is organized as follows.

Chapter 2 introduces basic concepts and notations used throughout the thesis, such as state spaces, actions, costs and paths. Because all the algorithms proposed in the subsequent chapters can be thought of as various extensions of Korf's LRTA* algorithm, we describe LRTA*, the most popular and fundamental of all the real-time search algorithms, with an illustration of how this basic algorithm works by a specific example.

Chapter 3 presents a new technique for proving the convergence of LRTA*. The chapter begins with a detailed survey of existing proof techniques for its completeness and convergence. We point out that the original convergence proof of LRTA* relies on an assumption that initial heuristics are consistent. Then, we propose an alternative proof of convergence which does not rest on the consistency assumption at all, based on a different technique. Besides its generality, the obtained proof is succinct and intuitive to understand, as it fully exploits the nature of LRTA*. In addition, as it is constructed as

a (nontrivial) extension of the proof of completeness, it helps to establish the link between these two important properties of LRTA*. Although these are closely related properties of the same algorithm LRTA*, they have been proven by somewhat different techniques. We also consider an alternative model of repeated trials, in which each trial may be cut off before it can reach the goal, either compelled by the environment or at the problem solver's own will. We show that our proof technique gives an equally satisfactory convergence result in this case. This new model is closely related to the δ -search algorithm which will be introduced in Chapter 5, where the proof technique is utilized to derive its property. This chapter is based on the material that appeared as [37].

In Chapter 4, we analyze the properties of the real-time search algorithm that uses overestimated, or inadmissible, heuristic function. One important question concerning such algorithms is whether they are convergent at all. We give an affirmative answer to this open question, by showing that if the amount of overestimation of initial heuristic estimates is at most a factor ϵ of the actual costs, then the converged solutions need not be optimal, but they are no worse than a factor $(1 + \epsilon)$ of the optimal cost. This results in the first suboptimal real-time search algorithm with nontrivial guarantee on solution quality. Since most of the problems addressed in AI are inherently intractable, the algorithms that relax optimal solutions have practical merits. The reported experimental results demonstrate that at the sacrifice of small degradation in solution quality, it is possible to reduce the amount of search effort dramatically. We also show a variation of the algorithm that measures the amount of overestimation additively relative to the actual costs, which brings about a different performance guarantee. This chapter is based on the material that appeared as [16, 17, 36].

In Chapter 5, we address the problem of the convergence process of LRTA* algorithm, namely, the instability of solution quality. Our interpretation of such instability is the LRTA*'s inability to balance exploration-exploitation trade-off. As a remedy to the problem, we propose a new real-time heuristic search algorithm called δ -search. This algorithm utilizes two heuristic func-

tions to guide the problem solver; the conventional, admissible heuristic function, and the new, often inadmissible heuristic function for maintaining the upper bounds of the actual costs. The upper-bound heuristic is obtained after the first trial, and is utilized to suppress excessive exploration. We not only analytically show δ -search's contribution to the stability of convergence process, but present experimental results demonstrating its effectiveness. This chapter is based on the material that appeared as [16] and [17].

Chapter 6 deals with the problem of moving-target search, in which the goal, or *target*, changes its location during the course of problem solving. It is an interesting application of real-time search, as the existence of such moving target renders off-line search algorithms powerless. Based on the insights gained in the previous chapters, especially Chapter 4, we discuss the use and the effect of overestimated, and hence inconsistent, heuristics, again, but in the case of moving-target search. In particular, we show that even if we use overestimated heuristics or the value-update rules that do not force nondecreasing property on the estimates, the completeness of Ishida and Korf's moving-target search algorithm are still preserved, on the contrary to what was believed. This chapter is based on the material that appeared as [38].

Finally, Chapter 7 concludes the thesis with the discussion of possible future research directions.

Chapter 2

Preliminaries

In this chapter, we introduce the basic concepts and notations used throughout the thesis.

2.1 State Space

Although real-time heuristic search removes the barrier between planning and commitment to actions, it is still built upon the state space representation. Therefore, we should first translate the process description of the problem at hand into state space representation¹, by associating a “state” to each stage of the problem solving process. Although the issue of how this translation should be done itself is another major problem, it is not our concern in this thesis. We assume such translation is given a priori, and concentrate on how search algorithms perform in the state space.

Let \mathbb{R} be the set of reals, and \mathbb{R}_+ be the set of positive reals. The state space we consider is formally defined as a quintuple (X, A, k, S, G) , which is basically a finite directed simple graph (X, A) without a loop (cycle of length one), where X is a nonempty finite set of states (nodes), and $A \subset X \times X - \{(x, x) \mid x \in X\}$ is a set of actions (edges)² with each edge labeled

¹See step 1 of the overall problem-solving architecture found in Section 1.1.

²The assumption that the underlying graph being simple and not containing loops is not a restriction, since in the LRTA* setting, the outcome of every action as well as the

by an action cost function $k : A \rightarrow \mathbb{R}_+$, together with two distinct nonempty subsets S and G of X , called initial and goal states, respectively. We use $k(x, y)$ as a shorthand for $k((x, y))$.

For $x, y \in X$, y is a *child* of x iff $(x, y) \in A$. We denote the set of children of x by $\text{Chd}(x)$; i.e., $\text{Chd}(x) = \{y \mid (x, y) \in A\}$. A *path* is a nonempty (possibly infinite) sequence $(x_0, x_1, \dots, x_i, \dots)$ of states, with every pair (x_i, x_{i+1}) of successive states in the sequence belonging to A . For each successive states (x_i, x_{i+1}) in the path, if the index i is clear from context, x_{i+1} is said to be the *successor* of x_i . The *length* of a path is the cumulative number of actions involved in the path; i.e., one less than the cumulative number of states involved. The *cost* of a path is the sum of the action costs associated with each successive state pair involved in the path. If a path is an infinite sequence, its cost is ∞ , reflecting $k(\cdot, \cdot)$ being always positive. The cost of a state is the minimum cost of the paths from the state to a goal state. The path that gives such minimum cost is called an *optimal path* from the state³. Respecting the convention of off-line search community, we use $h^*(x)$ to denote the optimal cost of state x . In particular, $h^*(g) = 0$ for each goal $g \in G$. We assume for every non-goal state $x \in X - G$ in the state space, there exists at least one path from x to a goal state. This assumption implies

- the optimal cost function $h^*(\cdot)$ is bounded from above, and
- $\text{Chd}(x) \neq \emptyset$ for every non-goal state $x \in X - G$.

In such a state space, the following relation holds for every state $x \in X$.

$$h^*(x) = \begin{cases} 0, & \text{if } x \in G; \\ \min_{y \in \text{Chd}(x)} \{k(x, y) + h^*(y)\}, & \text{otherwise.} \end{cases} \quad (2.1)$$

This relation is well-known as *Bellman optimality equation* [3, 5], and is solvable with dynamic programming or with Dijkstra's method [4]. Unfortunately, they are basically breadth-first algorithms that involve computation action itself is observable by the problem solver. For example, even when there are loops, we can safely ignore them.

³Note that there may be more than one optimal paths.

over all states in X ; heuristic search methods, in contrast, tries to reduce the search effort by carrying out computation at a limited number of relevant states. The computation is done in on-demand manner only when and where it becomes necessary with the help of prior knowledge specific to each domain. In this sense, heuristic search methods try to capture and mimic an aspect of humans' capability of problem solving.

2.2 Heuristics

Most of the AI heuristic search methods, including those discussed in this thesis, are intended to be domain-independent. Therefore, to be usable by these methods, the prior knowledge of how to accomplish the task efficiently in each domain should be translated into some more generic form, just as the problem solving task was translated into the state space description.

Such generic form is called a “heuristic evaluation function” and is defined in terms of the translated state space. This function gives an estimated difficulty of problem solving at each stage of problem solving. Since problem solving stage is represented by a state in the state space, the estimated difficulty at each stage are naturally encoded as an estimated distance from each state to the nearest goal. Formally speaking,

Definition 2.1 A *heuristic (evaluation) function* h is a mapping from states in the state space to nonnegative reals, i.e., $h : X \rightarrow \mathbb{R}_+ \cup \{0\}$. The value $h(x)$ is called the *heuristic estimate* of state x (by the heuristic function $h(\cdot)$).

It should be emphasized that $h(\cdot)$ gives only an *estimate* of the optimal cost $h^*(\cdot)$, therefore it may or may not be accurate. Conversely, when it is necessary to emphasize that the quantity $h^*(\cdot)$ is not an estimate but is determined solely by the topology of the state space (i.e., without intervention of specific heuristic function or search algorithm), we call it the *actual cost* or *correct cost*.

For the convenience of the later discussions, we also introduce a notion describing the heuristic function associates a correct estimates to a particular

state.

Definition 2.2 The heuristic estimate $h(x)$ of state x is *correct* iff $h(x) = h^*(x)$. Such a state x is called correct state.

There are several classes of heuristic function that are of interest to heuristic search, both off-line and real-time. The first class, called “admissible” heuristic function always gives an optimistic estimate of the distance to the goal.

Definition 2.3 The heuristic estimate $h(x)$ of the state x is said to be *admissible* iff the following condition is satisfied.

$$h(x) \leq h^*(x).$$

We call such state x an admissible state. The heuristic function that assigns admissible values to all the states is called admissible heuristic function.

Since we have $h^*(g) = 0$ for every goal state $g \in G$, we immediately have $h(g) = 0$ for all $g \in G$, provided the heuristic function h is admissible.

The “consistency” of heuristic function states that a form of triangle inequality holds between the heuristic values of any two adjacent states. The class of consistent heuristic functions is known to possess many advantages when it is used with some heuristic search algorithms.

Definition 2.4 The heuristic function $h(\cdot)$ is *consistent* iff for any pair (x, y) of states in A , $h(x) \leq k(x, y) + h(y)$.

There is another class of heuristic function, called “monotonic” heuristic function, which defines another form of triangle inequality. However, as it was shown [29] that this class is actually equivalent to consistent heuristics, we do not include its definition here, but use these two terms interchangeably throughout the thesis.

The following theorem clarifies the relationship between admissibility and consistency (or monotonicity).

Theorem 2.1 [29] Consistent (monotonic) heuristics are admissible.

An important corollary of this theorem that is of great interest to the thesis is its contrapositive.

Corollary 2.1 Inadmissible heuristics are inconsistent.

2.3 The LRTA* Algorithm

The Learning Real-Time A* (LRTA*) algorithm proposed by Korf [23] is probably the most popular and fundamental of all the real-time heuristic search algorithms. For this reason, the subsequent chapters discuss mainly the properties of LRTA* and its extensions. This section reviews the LRTA* algorithm. Although Korf has given the name LRTA* to a *family* of real-time search algorithms, we concentrate on the simplest version of LRTA* algorithm with search (look-ahead) horizon of one⁴ throughout the thesis.

A pseudo-code of the LRTA* algorithm is shown in Figure 2.1. The problem solver maintains its current location in the variable x . It departs from an initial state $x = x_0 \in S$, chosen arbitrarily from the set S of initial states, and iterates Steps 3a and 3b, until it reaches a goal state, i.e., until $x \in G$. In this algorithm, each state $z \in X$ is associated with an estimate $h(z)$ of the optimal cost $h^*(z)$. The problem solver performs a series of moves updating the estimate $h(x)$ of its currently occupying state x to make it more accurate. We assume the initial heuristic estimates are given by some admissible heuristic function $h_0(\cdot)$.

If the branching factor of the whole states is upper-bounded by some small constant (which is a reasonable assumption in many cases), one iteration of LRTA* algorithm can be accomplished within a constant-time. This is why the search methods are coined as *real-time* search; by using real-time search methods, it is possible for the problem solver to commit to an action in real

⁴It is not so difficult a task to extend the results of the subsequent chapters to the whole family in which search horizon is not limited to one. Unfortunately, because of the variations of the look-ahead, the arguments become too lengthy without much merit.

1. For each state z , set $h(z)$ to be its initial value; i.e., $h(z) \leftarrow h_0(z)$ for every $z \in X$.
2. Set the current state x to some initial state $x_0 \in S$; i.e., $x \leftarrow x_0$.
3. Repeat the following steps until x is a goal state; i.e., $x \in G$;
 - (a) Look ahead and value update:
Update the heuristic estimate $h(x)$ of the problem solver's current state x .

$$h(x) \leftarrow \min_{y \in \text{Chd}(x)} [k(x, y) + h(y)]. \quad (2.2)$$

- (b) Action execution:

Move to a child $y \in \text{Chd}(x)$ of the current state x such that

$$y \in \operatorname{argmin}_{z \in \text{Chd}(x)} [k(x, z) + h(z)]. \quad (2.3)$$

If there are more than one such state, choose among them arbitrarily. This operation is called a *tie break*.

Figure 2.1: Pseudo-code of LRTA* algorithm

time at each stage of problem solving, leaving the room for adapting itself to dynamics involved in the problem.

Remark 2.1 Notice that the algorithm lacks the routine to check whether there is no children available at Step 3b. Actually, such a check is not needed at all; since $\text{Chd}(x) \neq \emptyset$ for every non-goal state $x \in X - G$ in our setting, there should always be a place to move in Step 3b. It follows that the problem solver will either

1. stray in the state space forever, or,
2. arrive at a goal state and terminate the search.

Section 3.2.1 contains the proof that Situation 1 cannot actually occur, i.e., LRTA* will never fail to reach a goal.

Let us illustrate how LRTA* algorithm works by an example. Figures 2.2 through 2.5 depicts one iteration inside Step 3 of LRTA*. In the figures, the problem solver is represented by a robot, and the states by the square floor tiles. The current situation it is facing is depicted in Figure 2.2. The state currently occupied by the robot has four children, and all the transition costs are equally one. The number on the flag at each state indicates the heuristic estimates of the state at the instant.

1. Look-Ahead:

Figure 2.3 illustrates how “look-ahead” step is performed. The robot looks around the child states’ heuristic estimates, to compute the minimum of these values⁵.

2. Value Update:

Then it updates the estimate of the current state by the sum of the computed minimum and the cost of action which, in this case, is 1 (Figure 2.4).

⁵In general, we have to take the cost of actions into account as well; See value update formula (2.2). The process is omitted here because all the actions have uniform unit cost in this case.

3. Action Execution:

After updating the value, the robot moves towards the state that gives the minimum value used for the updating (Figure 2.5); in this case to the state with the estimate of 2. In this sense, the problem solver acts greedily in LRTA*. It would not be possible to escape from a loop or a dead-lock, if we used stationary heuristic estimates; LRTA* avoids such situation thanks to the value update of the estimates.

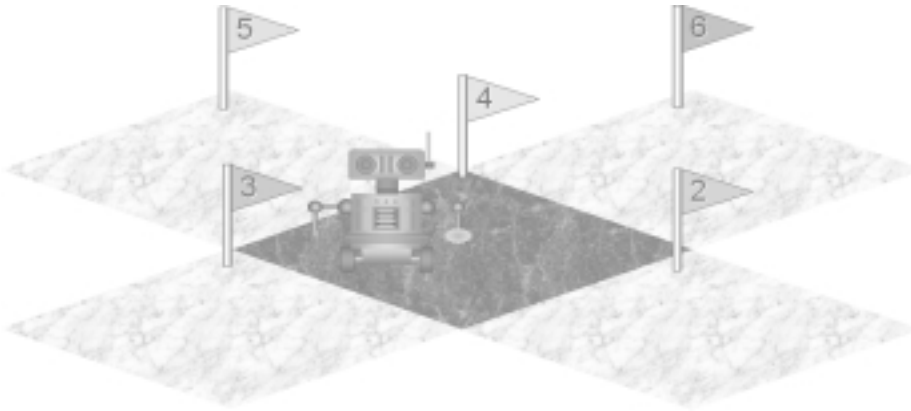


Figure 2.2: Robot navigated by LRTA* at a crossing

2.4 Properties of Real-Time Search Algorithms

Introduction of some more additional terms and concepts specific to real-time search closes this chapter. The main objective of this section is to present, informally but intuitively, two basic property addressed by real-time heuristic search algorithms. The first one, which is probably the most fundamental, is called completeness. It is addressed by almost all the real-time search algorithms proposed so far.

Definition 2.5 (Completeness) A real-time search algorithm is *complete*, iff it has the property of never failing to arrive at a goal state.

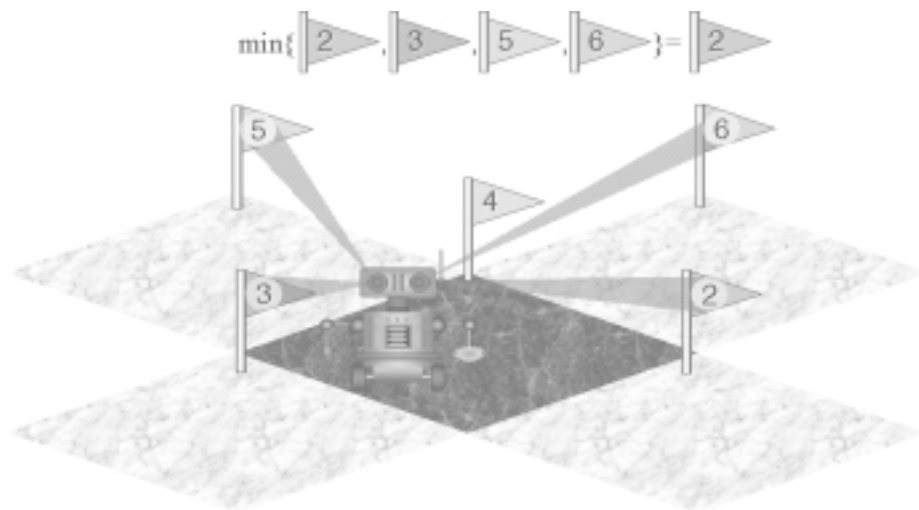


Figure 2.3: LRTA*: look ahead

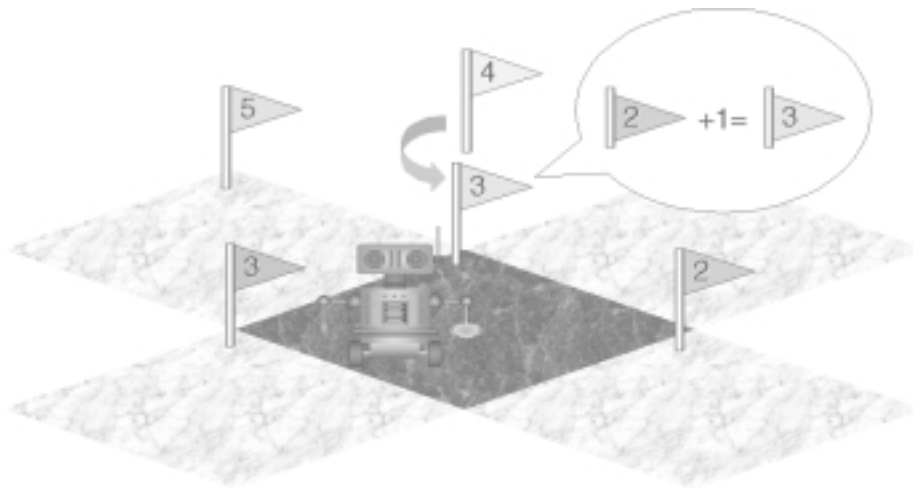


Figure 2.4: LRTA*: value update

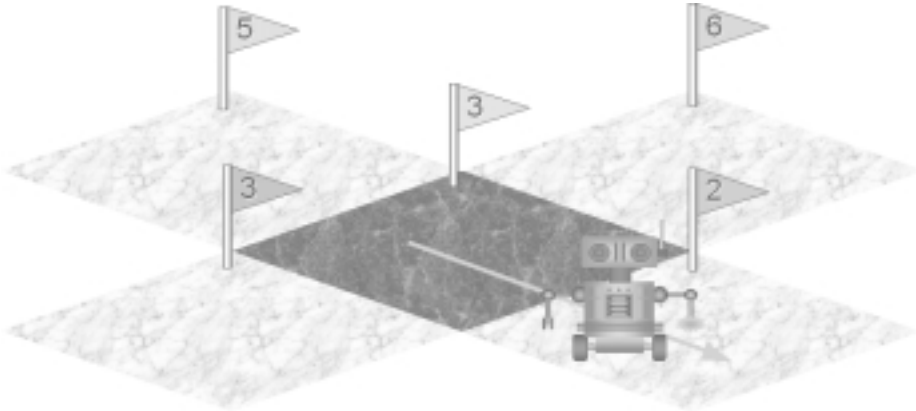


Figure 2.5: LRTA*: move

By a *problem solving trial*, or simply *trial*, we mean a single application of a real-time search algorithm to the given problem; i.e., a session beginning with the problem solver's departing from an initial state, and, if the algorithm is complete, terminating on its arrival at a goal. We call the problem solver's trajectory (the path departing from an initial state and arriving at a goal) in each trial the *solution* obtained by the problem solver in the trial.

If a real-time search algorithm is complete, it is conceivable to repeatedly apply the algorithm by resetting the problem solver to an initial state after it reaches a goal, and by reusing the modified heuristic estimates in the next trial. A series of such repeated application is called an *episode*. Now we are ready to introduce the second property.

Definition 2.6 (Convergence) A real-time search algorithm is *convergent*, iff it has the property such that after performing some number of trials, the problem solver continues to traverse only optimal paths.

A real-time search algorithm that possesses this property exhibits that it has the ability to learn from experience.

Note that the completeness and the convergence are basically independent properties. While LRTA* algorithm enjoys both these properties, not all of

the real-time search algorithms do. For example, RTA* [23] is a complete search algorithm but is not convergent in general.

Chapter 3

Convergence of Real-Time Search

3.1 Introduction

One of the significant features of real-time heuristic search algorithms is their explicit interleaving of planning and action execution. Since the planning is based on the partial information available at each stage of problem solving, it is not guaranteed that the algorithms execute an optimal sequence of actions. Yet, they still possess the favorable property of never failing to arrive at a goal (*completeness*). In addition, some of the algorithms including LRTA* can eventually identify an optimal path to the goal, if they are repeatedly applied to the similar problems (*convergence*). This property is the result of the way local information is maintained; at each stage (which is represented by a state in the state space), the information gathered by the local look-ahead search is accumulated and propagated to other stages via the value-updating of heuristic estimates.

The formal proof of this convergence property of LRTA* was derived by Korf in his pioneering paper [23] on real-time search. Although his proof covers the case of initial heuristic function being *consistent*¹, it seems non-

¹The consistency of initial heuristic function is assumed implicitly in the convergence

trivial to extend it to the whole class of admissible heuristic functions. In this chapter, we present an alternative convergence proof applicable regardless of the consistency of initial heuristic function. The new proof technique is built upon the lemmas used by Ishida and Korf for proving the completeness of their moving target search algorithm [14, 15], and not upon Korf’s convergence proof. To show that our proof is not a trivial extension of these previously known techniques, we review both Korf’s convergence proof and Ishida and Korf’s completeness proof in detail. We also compare our technique briefly with that of Barto *et al.* [2] who also established the general convergence result by reducing the problem to asynchronous dynamic programming.

The technique presented in this chapter has specific applications in Chapters 4 and 5.

3.2 Preliminaries

In this section, we clarify the background and the motivation of the present chapter, by reviewing the previous results on the properties of LRTA*.

The LRTA* algorithm is already depicted in Figure 2.1. To analyze its properties, we introduce the quantity τ to count how many iterations of Step 3 of the algorithm (see Figure 2.1) have been performed since the problem solver has departed from the initial state x_0 . Defined this way, τ represents the number of moves² made by the problem solver. In the following, by *time (instant)* τ , we refer to the instant immediately after τ -th iteration; thus, for example, at time 0, the problem solver is at an initial state.

For time $j = 0, 1, 2, \dots$, let x_j be the state occupied by the problem solver at time j , and for any $x \in X$, let $h_j(x)$ be the heuristic estimate associated with x at the same instant; note that these notations are consistent with our previous definitions of x_0 and $h_0(\cdot)$. From the value update formula (2.2) and

proof. On the other hand, it is explicitly stated that consistency is not required for the completeness proof.

²Note the problem solver makes one move per iteration at Step 3b.

the action-selection formula (2.3), we see that the following equation holds for every $j = 1, 2, \dots$

$$\begin{aligned} h_j(x_{j-1}) &= \min_{y \in \text{Chd}(x_{j-1})} [k(x_{j-1}, y) + h(y)] \\ &= k(x_{j-1}, x_j) + h_{j-1}(x_j). \end{aligned} \tag{3.1}$$

On the other hand, the update accompanying j -th move does not alter the heuristic estimates of the states besides state x_{j-1} . Formally speaking,

$$h_j(x) = h_{j-1}(x) \tag{3.2}$$

for each $x \in X - \{x_{j-1}\}$. Since identity relation is excluded from A , we have $x_j \neq x_{j-1}$ for each time $j \geq 1$. It follows that $h_j(x_j) = h_{j-1}(x_j)$ and hence equation (3.1) can be restated as

$$h_j(x_{j-1}) = k(x_{j-1}, x_j) + h_j(x_j). \tag{3.3}$$

3.2.1 Completeness Proof by Ishida and Korf

LRTA* is *complete*, in the sense that the problem solver never fails to reach a goal after executing a finite number of moves. Since the convergence proof we will develop in Section 3.3 shares the lemmas with the completeness theorem, we will review it in this section.

Historically, the technique we review below was originally developed by Ishida and Korf [14, 15] for their *Moving-Target Search* (MTS) algorithm for chasing changing goals³. It is applicable to LRTA* as well, since it is subsumed by MTS as a special case when the target (goal state) does not change its location. We describe the proof based on the presentation due to Koenig [19], with slight modification.

We begin with an important lemma due to Korf, which is also used by his convergence proofs and ours. The lemma says that the value-updates of the problem solver never violates the admissibility of heuristic estimates.

³The basic idea is already present in [23].

Lemma 3.1 [23] Value-updates with formula (2.2) preserve the admissibility of heuristic function. That is, if the initial heuristic function $h_0(\cdot)$ satisfies $h_0(x) \leq h^*(x)$ for all $x \in X$, then for every time $j = 1, 2, \dots$, and for each $x \in X$, $h_j(x) \leq h^*(x)$ still holds.

Proof. We show by induction on time j . By assumption, obviously true at $j = 0$. Suppose at time j , the heuristic estimates are all admissible; i.e.,

$$h_j(x) \leq h^*(x) \tag{3.4}$$

for every $x \in X$. Since $x_j \neq x_{j+1}$ by the assumption that the state space does not contain a loop by (3.2),

$$h_{j+1}(x) = h_j(x) \leq h^*(x)$$

for every $x \in X - \{x_j\}$. It remains to show that the state x is admissible after update. Let $y \in \text{Chd}(x_j)$ be an optimal child of x_j . Since x_{j+1} is chosen as the destination, and since it is chosen rather than y ,

$$h_{j+1}(x_j) = k(x_j, y) + h_j(x_{j+1}) \leq k(x_j, y) + h_j(y) \tag{3.5}$$

holds⁴. Since y is admissible by formula (3.4),

$$h_j(y) \leq h^*(y). \tag{3.6}$$

Substituting inequality (3.6) in (3.5) yields

$$h_{j+1}(x_j) \leq k(x_j, y) + h^*(y),$$

or, the value $h_{j+1}(x_j)$ after update is also admissible. □

Lemma 3.2 [14, 15, 19] For each time instant $\tau = 0, 1, 2, \dots$, the following relation (3.7) holds.

$$\sum_{j=1}^{\tau} k(x_{j-1}, x_j) = \sum_{x \in X} h_{\tau}(x) - h_{\tau}(x_{\tau}) - \sum_{x \in X} h_0(x) + h_0(x_0). \tag{3.7}$$

⁴Formula (3.5) even holds when $y = x_{j+1}$.

Proof. Obviously true for $\tau = 0$. On j -th move, x_{j-1} is the only state whose heuristic estimate is updated, and other states remain unaffected. Therefore, from equations (3.2) and (3.3),

$$\begin{aligned} \sum_{x \in X} h_j(x) &= \sum_{x \in X} h_{j-1}(x) - h_{j-1}(x_{j-1}) + h_j(x_{j-1}) \\ &= \sum_{x \in X} h_{j-1}(x) - h_{j-1}(x_{j-1}) + k(x_{j-1}, x_j) + h_j(x_j). \end{aligned} \tag{3.8}$$

Substituting $H_j = \sum_{x \in X} h_j(x) - h_j(x_j)$, we can restate equation (3.8) as

$$k(x_{j-1}, x_j) = H_j - H_{j-1}. \tag{3.9}$$

Summing equation (3.9) over $j = 1, 2, \dots, \tau$ yields

$$\sum_{j=1}^{\tau} k(x_{j-1}, x_j) = H_{\tau} - H_0,$$

which is the statement of the lemma. □

The following theorem establishes the completeness of LRTA^{*}. It does not have the direct relationship with the convergence proof we will develop in Section 3.3; the above lemmas suffice for our purpose. We include it here for the following reasons as well as for self-containedness.

- The completeness is a prerequisite for our model of repeated problem solving trials. If LRTA^{*} were not complete, the trials could not be repeated, because we assume “reset operation” that brings the problem solver back to an initial state is only available at the goal states⁵.
- Again, since the completeness is a prerequisite for the repeated application of LRTA^{*}, if one wants to prove the consistency is not required for convergence, it is necessary to make sure that the consistency is not required for completeness, either.

⁵We will discuss an alternative “reset” model in Section 3.4.1

- We believe it helps to convince the reader that our convergence proof is *not* a trivial application of that of completeness, although both are built upon the same set of lemmas.

Theorem 3.1 (Completeness of LRTA^{*}) [14, 15, 19] If the initial heuristic function is admissible, LRTA^{*} is complete. In other words, the problem solver reaches a goal state after a finite number of moves.

Proof. Based on Remark 2.1, it suffices to derive an upper bound on the number τ of the moves by the problem solver for proving this theorem.

Let $k_{\min} = \min_{(x,y) \in A} k(x,y)$ be the minimum action cost in the state space. Then, we have

$$\begin{aligned}
\tau k_{\min} &\leq \sum_{j=1}^{\tau} k(x_{j-1}, x_j) \\
&= \sum_{x \in X} h_{\tau}(x) - h_{\tau}(x_{\tau}) - \sum_{x \in X} h_0(x) + h_0(x_0) \\
&\leq \sum_{x \in X} h^*(x) - \sum_{x \in X} h_0(x) + h_0(x_0). \tag{3.10}
\end{aligned}$$

The equality in the second line follows from Lemma 3.2, and the inequality in the last line from Lemma 3.1. Since $|X|$ and $h^*(\cdot)$ are both upper bounded, RHS and consequently LHS, τk_{\min} , are also upper bounded. As the positive cost assumption implies $k_{\min} > 0$, τ is upper bounded. \square

Note that neither Lemmas 3.1, 3.2, nor Theorem 3.1 makes the assumption of the consistency of initial heuristic function.

3.2.2 Original Convergence Proof by Korf

Theorem 3.1 showed that LRTA^{*} arrives at a goal state after making a finite number of moves. In the remaining part of the chapter, we consider *successive* application of LRTA^{*} algorithm to the same problem; once the problem solver reaches a goal and a trial ends, it is reset to an initial state chosen arbitrarily from the set S , and the search is continued. In this setting, we do not discard the heuristic estimates at the end of each trial. Rather, we inherit

and reuse them as the initial heuristics of the next trial. It is possible to repeat such problem solving trials ad infinitum; from Lemma 3.1, we know that the admissibility of heuristics is always maintained, and hence the prerequisite for the completeness of LRTA* is never violated during the repeated trials.

The rest of the section describes the original convergence proof of LRTA* algorithm by Korf. His proof, however, relies on the consistency of heuristic estimates, and is not general enough to cover the class of admissible heuristic functions as a whole.

The following lemma says that the correctness of the state propagates with the move by LRTA*. It holds regardless of the admissibility or consistency.

Lemma 3.3 [23] Let x be a state, and y be a child of x . If state y is correct, and if problem solver has made a transition from state x to y , state x becomes correct after the transition.

Proof. Let $h(x)$ be the heuristic value of x after transition. Since state y is correct, $h^*(y) = h(y)$. Therefore,

$$h^*(x) \leq k(x, y) + h^*(y) = k(x, y) + h(y) = h(x).$$

The inequality follows from the optimality requirements, and the last equality from the value update formula (2.2). By Lemma 3.1, we have

$$h(x) \leq h^*(x).$$

It follows that state x is correct. □

At this point, the following assumption is made implicitly in [23].

Assumption 3.1 [23] In LRTA*, a state remains correct once it becomes correct.

If this assumption is in fact true, then the number of correct states is nondecreasing with number of trials. It is naturally seen that there will be a trial after which all the states along the paths traversed by the problem solver are correct, because the correctness of states propagates backwards along the

paths, beginning with the goal states which are initially correct by assumption.

Theorem 3.2 [23] Under Assumption 3.1, by repeating LRTA* problem solving trials infinite number of times, the cost of the path departing from $s \in S$ and following the state y that gives the minimum value of $k(x, y) + h(y)$ among the children of the current state x , will be all optimal, provided that the ties are broken randomly.

Proof. Consider the path $\pi = (x_0, x_1, \dots, x_{\tau-1}, x_\tau)$ from an initial state $x_0 \in S$ to a goal state $x_\tau \in G$ traversed infinitely many times by the problem solver after infinite number of trials. We prove π is indeed optimal, by induction on the length of π . It is trivial when $\tau = 0$. Thus, let $\tau > 0$. For all $1 \leq j \leq \tau$, we have

$$\begin{aligned} k(x_{j-1}, x_j) + h^*(x_j) &= \min_{y \in \text{Chd}(x_{j-1})} [k(x_{j-1}, y) + h(y)] \\ &\leq h^*(x_{j-1}), \end{aligned}$$

or,

$$k(x_{j-1}, x_j) \leq h^*(x_{j-1}) - h^*(x_j),$$

since by Lemma 3.1 and by Assumption 3.1, every state along π is correct. Now, summing this inequality over $j = 1, 2, \dots, \tau$ yields

$$\sum_{j=1}^{\tau} k(x_{j-1}, x_j) \leq h^*(x_0) - h^*(x_\tau) = h^*(x_0),$$

where $x_\tau \in G$ and $h^*(x_\tau) = 0$ is used. □

Let us return to the discussion on Assumption 3.1. When initial heuristics are consistent, Assumption 3.1 holds, but otherwise it need not. For example, see the state space with initially inconsistent heuristic function, depicted in Figure 3.1. There are six states u, v, w, x, y and z besides the initial state s and the goal g , each illustrated with a circle. The initial heuristic estimates of those states, depicted in the circle representing the states, are admissible but

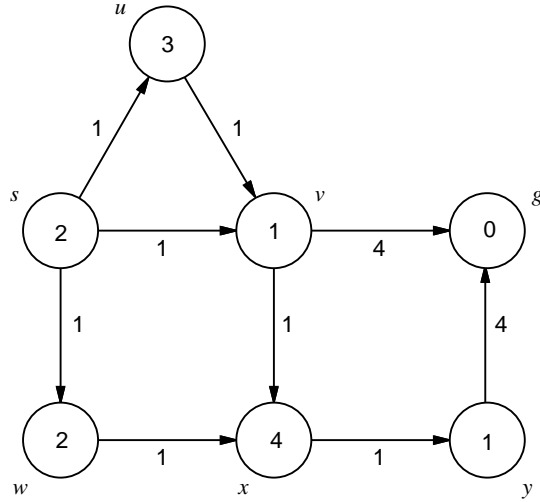


Figure 3.1: State space with inconsistent heuristic estimates

not consistent. The numbers labeling the edges are the cost associated with them. We invite the readers to verify that in this state space, state v does not remain correct even after it becomes, by the virtue of Lemma 3.3, correct. The trajectories of the problem solver is shown in Table 3.1. After the first trial, state v becomes correct, reflecting the problem solver's transition from v to g . However, the decrease in the estimate of x in the second trial redirects the problem solver from v to x and not to g in the third trial, rendering v into an incorrect state again.

3.3 Convergence

In this section, we show that LRTA^* is convergent even under inconsistent heuristics, based on a different technique from Korf's. Our strategy is to first show that after some trial, the set of the paths traversed by the problem solver in the subsequent trials will consists only of optimal paths. Once this property is derived, it is easier to derive the convergence of heuristic estimates to the optimal values. We will refer to the first type of the convergence as the

Table 3.1: Episode in the state space with inconsistent estimates

Trial	Traversed Path
1	$s \rightarrow v \rightarrow g$
2	$s \rightarrow w \rightarrow x \rightarrow y \rightarrow g$
3	$s \rightarrow u \rightarrow v \rightarrow x \rightarrow y \rightarrow g$
4	$s \rightarrow v \rightarrow g$
5	$s \rightarrow v \rightarrow g$
\vdots	$\vdots \quad \vdots$

“convergence of the trajectory⁶,” and the second type as the “convergence of the heuristic estimates,” and prove these properties in that order.

3.3.1 Convergence of the Trajectory

Let us introduce some notations first. For each trial $i = 1, 2, \dots$, let $\tau(i)$ be the number of moves the problem solver made in the i -th trial, and let $c(i)$ be the cost of the path traversed by the problem solver in the same trial. We denote by x_j^i the state at which the problem solver has arrived by the j -th move in the i -th trial (hence $0 \leq j \leq \tau(i)$), and for any $x \in X$ denote by $h_j^i(x)$ the heuristic estimate $h(x)$ at the same instant. Using these definitions, we have

$$c(i) = \sum_{j=1}^{\tau(i)} k(x_{j-1}^i, x_j^i). \quad (3.11)$$

The fact that the heuristic estimates on arriving at a goal is reused as the initial heuristics of the next trial is stated as follows: For any trial $i = 1, 2, \dots$, and any state $x \in X$,

$$h_{\tau(i)}^i(x) = h_0^{i+1}(x). \quad (3.12)$$

⁶This notion corresponds to the convergence of *policy* in reinforcement learning literature.

Lemma 3.4 (Corollary of Lemma 3.2) The following equation holds for each trial $i = 1, 2, \dots$:

$$c(i) = \sum_{x \in X} h_{\tau(i)}^i(x) - \sum_{x \in X} h_0^i(x) + h_0^i(x_0^i). \quad (3.13)$$

Proof. In the i -th trial, after making $\tau(i)$ moves, the problem solver arrives at a goal state $x_{\tau(i)}^i \in G$, at which $h_{\tau(i)}^i(x_{\tau(i)}^i) = h^*(x_{\tau(i)}^i) = 0$ by the admissibility of heuristic function. By noting this fact, and taking the definition of $c(i)$ (equation (3.11)) into account, it can be seen that equation (3.13) is a particular case of equation (3.7). \square

Lemma 3.5 For each trial $n = 1, 2, \dots$, equation (3.14) holds.

$$\sum_{i=1}^n c(i) = \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) + \sum_{i=1}^n h_0^i(x_0^i). \quad (3.14)$$

Proof. The proof is by induction on the number n of trials. By summing equation (3.13) over $i = 1, 2, \dots, n$, we have

$$\begin{aligned} \sum_{i=1}^n c(i) &= \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) \\ &\quad + \sum_{i=1}^{n-1} \sum_{x \in X} [h_{\tau(i)}^i(x) - h_0^{i+1}(x)] + \sum_{i=1}^n h_0^i(x_0^i). \end{aligned}$$

From the assumption on the reuse of the heuristic values, (i.e., equation (3.12)), the third term on RHS is 0. \square

Theorem 3.3 There is a trial after which the problem solver continues to traverse only optimal paths.

Proof. From equation (3.14), we have

$$\begin{aligned}
& \sum_{i=1}^n [c(i) - h^*(x_0^i)] \\
&= \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) + \sum_{i=1}^n h_0^i(x_0^i) - \sum_{i=1}^n h^*(x_0^i) \\
&\leq \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) + \sum_{i=1}^n h^*(x_0^i) - \sum_{i=1}^n h^*(x_0^i) \\
&= \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) \\
&\leq \sum_{x \in X} h^*(x) - \sum_{x \in X} h_0^1(x), \tag{3.15}
\end{aligned}$$

for every n . Here, Lemma 3.1 is used to derive the inequalities. Since RHS is a constant, the partial sums of the series $\sum_i [c(i) - h^*(x_0^i)]$ of nonnegative terms has an upper bound, and hence the series is convergent. Thus we have

$$\lim_{n \rightarrow \infty} [c(n) - h^*(x_0^n)] = 0,$$

which means that for any $\delta > 0$, there exists m such that for all $n \geq m$, $c(n) - h^*(x_0^n) < \delta$. It implies that after some trial m , $c(n) = h^*(x_0^n)$ for all $n \geq m$, since we can make the quantity δ such that every suboptimal path has a cost at least δ greater than that of the optimal path. This establishes the statement of the theorem. \square

Note that the above proof is valid regardless of the consistency of the heuristic function; it is neither required for the convergence proof nor the completeness proof which is required for such repeated problem solving trials to be possible.

Note also the arbitrariness of tie-breaking rule and the choice of initial states. If we use some tie-breaking scheme that always prefers one state to the rest, it is possible that only one optimal path will eventually be found. On the other hand, if we decide to break ties randomly, and choose initial state randomly as well, then all the optimal paths from every initial state in S will eventually be identified with probability one, as stated by Korf in his original LRTA* paper.

3.3.2 Convergence of the Heuristic Estimates

The previous section showed that there exists trial m such that all the paths traversed by the problem solver in the subsequent trials are optimal. It does not, however, imply that the heuristic estimates are convergent to optimal values on every state along the path traversed after trial m . Therefore, we prove that the heuristic estimates of the states along the optimal paths that are eventually traversed by the problem solver will also converge to the exact cost to the nearest goals.

The statement follows from the following fact. Suppose the solution cost have already converged. If we take two arbitrary states along a path traversed by the problem solver in the subsequent trials, an order between these states can be defined in terms of which state is visited after the other in a trial. The argument is intuitive but a bit lengthy, so we omit the detail and sketch the outline only.

First, we introduce several definitions and a lemma. Let $X^* \subset X$ be the set of states that are visited by the problem solver after infinite number of trials. By Theorem 3.3, all the states belonging to X^* is on an optimal path from an initial state. Note, however, that this set need not coincide with the set of all states on optimal paths, because due to the arbitrariness of tie-breaking strategy and the way initial states are chosen.

Definition 3.1 For every $i = 0, 1, \dots$, define the sequence $\text{Strat}(i)$, which we call i -th stratum, of the set X^* inductively as follows.

1. $\text{Strat}(0) = G \cap X^*$,

And for $i = 1, 2, \dots$,

2. $\text{Strat}(i) = \{x \in X^* \mid \text{for all } y \in \text{Chd}^*(x) \cap X^*, y \in \text{Strat}(i - 1)\}$.

Here, $\text{Chd}^*(x)$ denotes the set of the optimal children of state x , i.e.,

$$\text{Chd}^*(x) = \underset{y \in \text{Chd}(x)}{\text{argmin}} [k(x, y) + h^*(y)]$$

when $x \in X - G$ is a non-goal state, and, for a goal state $x \in G$, $\text{Chd}^*(x) = \emptyset$ ⁷. Furthermore, let the level of the state x , denoted by $\text{Lev}(x)$, be the smallest index of the stratum in which x belongs; i.e.,

$$\text{Lev}(x) = \underset{i}{\text{argmin}} \{i \mid x \in \text{Strat}(i)\}.$$

The set $\text{Strat}(i)$ is nondecreasing with respect to set inclusion (\subset). On the other hand, by the definition and the finiteness of X^* , there exists an index i such that $\text{Strat}(i) = X^*$, hence, for every $x \in X^*$, $\text{Lev}(x)$ is well-defined and is finite.

Lemma 3.6 Suppose the problem solver follows only optimal paths. Then, the heuristic estimate of the state x on the path will only be affected by the change of the heuristic values of the states belonging to the level $\text{Lev}(x) - 1$.

From Lemmas 3.3 (propagation of correctness) and 3.6, we have the following corollary.

Corollary 3.1 Suppose all the states belonging to level i are correct. If a state at $(i + 1)$ -st level becomes correct, then it will remain correct in the subsequent trials.

Although it is rather intuitive after we saw Corollary 3.1, we show the theorem for convergence of heuristic estimates to the correct values.

Theorem 3.4 (Convergence of Heuristic Estimates)

After some trial, all the heuristic estimates of the states visited by the problem solver will be correct.

Proof. (Sketch) we prove by induction on level i that all the states in X^* will be correct after some trial. When $i = 0$, obviously true since $\text{Strat}(0)$ contains only goal states.

Suppose all the states in i -th level are correct after some trial at which the paths traversed by the problem solver in the subsequent trials consist only of

⁷By this definition, it follows that $\text{Strat}(0) \subset \text{Strat}(1)$.

optimal ones. The problem solver eventually visits every states in $(i + 1)$ -st level. By Corollary 3.1, if all the states at i -th level are correct, the states in $(i + 1)$ -st level will be correct sooner or later as well. \square

In particular, again, as in Korf’s original paper, when tie-break and the choice of initial state are done at random, the heuristic estimates of the states along all the optimal paths will eventually become correct with probability one, because the set X^* coincides with the set of all states on all optimal paths in this case.

3.4 Variations

In this section, we present several application of our convergence proof techniques to some nonstandard settings.

3.4.1 Alternative Reset Model with Deadlines

The model of repeated problem solving trials considered in the previous sections followed the way proposed by the original LRTA* paper [23]; the problem solver is reset to an initial state only after it reaches a goal.

In this section, we consider an alternative model of repeated problem solving trials. In this model, we set a deadline for arriving at a goal in each trial. That is, regardless of whether the problem solver has arrived at a goal or not, a trial is cut off and the problem solver is reset to an initial state after a certain period of time has elapsed since its departure from the initial state. The deadline could also be defined in terms of the distance traveled by the problem solver instead of elapsed time. In either case, such a model is obtained by simply removing the constraints $x_\tau(i) \in G$ and $h_{\tau(i)}^i(x_\tau(i)) = 0$ implied thereby. However, the deadline should not be too restrictive to make reaching a goal totally impossible; for instance, it is hopeless to expect the problem solver ever reach a goal, if the deadline, defined in terms of distance, is less than the cost of the optimal solution. This constraint is guaranteed by making an explicit assumption on the cost $c(i)$ of the path traversed in trial

i such that

$$c(i) \geq h^*(x_0^i) \quad (3.16)$$

for every i . Note that in the original search model, this assumption was implicit in the model itself.

For this new model, we have a corollary of Lemma 3.2, which is slightly different from (actually, more general than) Lemma 3.4.

Lemma 3.7 (Corollary of Lemma 3.2)

The following equation holds for each trial $i = 1, 2, \dots$:

$$c(i) = \sum_{x \in X} h_{\tau(i)}(x) - h_{\tau}(x_{\tau(i)}) - \sum_{x \in X} h_0^i(x) + h_0^i(x_0^i). \quad (3.17)$$

Lemma 3.8 For each trial $n = 1, 2, \dots$, equation (3.18) holds.

$$\sum_{i=1}^n c(i) = \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) + \sum_{i=1}^n \left[h_0^i(x_0^i) - h_{\tau(i)}^i(x_{\tau(i)}^i) \right]. \quad (3.18)$$

Proof. By summing equation (3.17) over $i = 1, 2, \dots, n$. \square

Theorem 3.5 Assume $c(i) > h^*(x_0^i)$ for every trial i as in (3.16); i.e., in each trial, the problem solver is not subject to reset until it travels the cost of the optimal path. Then, there is a trial after which the problem solver continues to traverse optimal paths.

Proof. From equation (3.18), we have

$$\begin{aligned} & \sum_{i=1}^n [c(i) - h^*(x_0^i)] \\ &= \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) + \sum_{i=1}^n [h_0^i(x_0^i) - h_{\tau(i)}^i(x_{\tau(i)}^i)] - \sum_{i=1}^n h^*(x_0^i) \\ &\leq \sum_{x \in X} h^*(x) - \sum_{x \in X} h_0^1(x), \end{aligned}$$

for every n . By condition (3.16), $\sum_i [c(i) - h^*(x_0^i)]$ is a series of nonnegative terms and hence converges. \square

The proof is almost identical to the one presented in Section 3.3, but we shall see that it has a direct application in the proof of convergence of real-time search with upper bounds that we will introduce in Chapter 5.

3.4.2 Inadmissible Initial Heuristic Function

Our proof technique does not depend on the consistency of heuristic function, but the admissibility is still indispensable. This is because the proof essentially relies on the convergence of series of nonnegative terms towards explicit limit. If the heuristic function violates admissibility, the converged solution need not be optimal. This implies that the problem solver may traverse a path whose cost is less than the final solution during convergence. This makes it not so straightforward to extend Theorem 3.3 to inadmissible case. We can still derive an upper bound on the *average* solution cost in this case, but we defer the discussion until Section 4.7.1.

3.5 Discussion

If we assume consistent heuristic function, then such function will guarantee the heuristic estimates to be nondecreasing, and to remain consistent. It follows that the number of correct states is also nondecreasing, which is an essential assumption underlying the convergence proof of Korf. Unfortunately, it is not necessarily true of *inconsistent* heuristic functions. In this case, the number of correct states may decrease during the search, as shown in the example of Figure 3.1.

One way to circumvent this problem is to explicitly *force* the heuristic estimates to be nondecreasing, by performing an update only when it gives greater value than the current one [19, 35].

$$h(x) \leftarrow \max \left\{ h(x), \min_{y \in \text{Chd}(x)} [k(x, y) + h(y)] \right\}. \quad (3.19)$$

We believe this form of value update formula has its root in the method to construct a consistent heuristic function out of inconsistent one, found in the

off-line search literature [25] (see also [22]). When this max-version of update formula is used with LRTA*, Korf’s argument becomes applicable regardless of the consistency of heuristic function. We mention that even when this max-version of update formula is in use, our proof technique applies as well.

Koenig [19] gives a detailed theoretical analysis of the worst-case performance of LRTA* algorithm, both at the single and the repeated problem solving frameworks. Although he used Ishida and Korf’s proof technique to analyze single-trial performance, a totally different technique is used to prove and analyze convergence. His convergence proof makes use of the extra mechanism of *tagging* states that are found to be correct, based on Korf’s Lemma 3.3⁸. A state is tagged when its estimate is updated on a transition to another tagged state. Furthermore, the problem solver should prefer non-tagged states over tagged ones on tie-breaking. This is indeed a nice mechanism to encourage efficient exploration of state space and to decide when to stop further trials. Note, however, that the tagged states remain tagged thereafter as the algorithm does not contain a procedure to remove tags from states. In this sense, his analysis of convergence carries the same assumption as Korf’s, thus requiring max-operation in the update rule to be indispensable. With the state space of Figure 3.1, it is possible to verify that the tagging mechanism does not always work without max-operation. If LRTA*’s original value update formula were used, state s would be tagged as “correct” after the fourth trial, which is wrong the correct cost of s should be 5, while the estimate is still 4 after tagging.

Barto *et al.* [2] proposed trial-based real-time dynamic programming (trial-based RTDP), which is a generalization of LRTA* to the stochastic state space (or Markov decision process) where the mapping from actions to transitions are not one-to-one relation, but occurs according to a predetermined distribution. It is also applicable to deterministic state space, in which case the algorithm reduces to LRTA*. In their paper, they showed the convergence

⁸It seems that the introduction of such a tag is intended for the removal of random tie-breaks. Using such mechanism, Koenig succeeds in deriving an upper bound on the number of moves required for LRTA* to converge.

of trial-based RTDP, and their convergence proof covers the case of inconsistent heuristics. They solved the problem by reducing the convergence of trial-based RTDP (which subsumes LRTA* as a special case) to that of asynchronous dynamic programming [5].

Their proof differs essentially from ours in the strategy. They first show that there is a set of states that are visited infinite number of times, without indicating it consists only of the states along optimal paths; later they turn out to be, as a result of the convergence of heuristics to optimal values. This strategy is reminiscent of Korf's original proof, and therefore can be considered as a direct extension of his proof to the inconsistent as well as stochastic case. In contrast, ours first explicitly shows the states visited infinite number of times will be only those along optimal paths; the convergence of heuristic estimates in each state follows naturally.

3.6 Summary

In this chapter, the convergence of LRTA* is proven by a new technique that does not rest on the consistency assumption at all. Since it is a natural extension of the proof of the *completeness*, it establishes the connection between these fundamental properties of LRTA* that have been discussed somewhat independently. We also compared our technique with that of Korf, Koenig, and Barto *et al.* Among these, Barto *et al.*'s proof via reduction to asynchronous dynamic programming, is the most general. Although our proof lacks such generality to cover the case of stochastic state space, it fully exploits the setting of LRTA*, leading to a more succinct and intuitive proof.

Chapter 4

Weighted Real-Time Search for Tolerating Suboptimal Solutions

4.1 Introduction

In this chapter, we remove the constraint on the heuristic function that it should be admissible and investigate the effect of such nonstandard heuristic function on real-time search algorithms.

In Chapter 3, we saw that LRTA* algorithm enjoys convergence to optimal solution as well as completeness, provided that the initial heuristic function is admissible. From the viewpoint, it is preferable that the heuristic function is admissible, but there are problem domains for which it is difficult to find a heuristic function that are both admissible and effective [29]. Should we abandon the use of real-time search in such domains? What will happen if we apply real-time search even though the heuristic estimates are initially inadmissible?

Although there have been a number of literature on real-time search, most of them concentrated on the search with admissible and consistent heuristic functions, and the use of inadmissible ones are completely neglected. In par-

ticular, it is not clear whether repeated application of LRTA* algorithm with initially inadmissible heuristic function shows any sort of convergence. The primary objective of this chapter is to give an answer to this open question.

4.2 Preliminaries

Let the state space be (X, A, k, S, G) as defined in Section 2.1; X is the finite set of the states, A is the set of actions, $k(\cdot, \cdot)$ defines the cost of every action in A , and $S, G \subset X$ are the sets of initial and goal states, respectively.

As we have seen in Section 2.3, LRTA* uses admissible heuristic function (such as the Manhattan distance in the gridworld domain and the sum of the Manhattan distance of each numbered tile in the sliding-tile puzzle domain) as the initial value of $h(x)$. In the following, we remove the constraint on heuristic function $h(\cdot)$ that it should be admissible. Even in this case, since $h(\cdot)$ is initially finite, there should be some constant $\epsilon \geq 0$ such that $h(x) \leq (1 + \epsilon)h^*(x)$ for every state $x \in X$. Let us define such weakened constraint more formally.

Definition 4.1 Let $h(\cdot)$ be some heuristic function. For a constant $\epsilon \geq 0$, if the heuristic value of state x does not overestimate the exact cost by a multiplicative factor ϵ , i.e.,

$$h(x) \leq (1 + \epsilon)h^*(x), \quad (4.1)$$

then the heuristic value $h(x)$ of state x is said to be ϵ -admissible. The heuristic function $h(\cdot)$ that assigns an ϵ -admissible value to every state is called an ϵ -admissible heuristic function. A state x satisfying the condition (4.1) is called an ϵ -admissible state.

Next, we introduce the notion of ϵ -correctness, which defines a stronger condition than ϵ -admissibility, yet weaker than correctness.

Definition 4.2 For some constant $\epsilon \geq 0$, if the state x satisfies the condition

$$h^*(x) \leq h(x) \leq (1 + \epsilon)h^*(x)$$

then, x is said to be an ϵ -correct state.

It is obvious that if a state is ϵ -correct, it is also ϵ -admissible, but not vice versa. Furthermore, by generalizing the notion of optimal path, we define ϵ -optimal path as follows:

Definition 4.3 Let π be a path from a state $x \in X$ to a goal $g \in G$. If the cost of π does not exceed the optimal cost¹ $h^*(x)$ of x by at most a factor $(1 + \epsilon)$, then, π is said to be ϵ -optimal path from state x . More precisely, if $\pi = (x_0, \dots, x_{n-1}, x_n)$ where $x = x_0$ and $g = x_n$, and if

$$\sum_{i=0}^{n-1} k(x_i, x_{i+1}) \leq (1 + \epsilon)h^*(x)$$

is satisfied, then π is an ϵ -optimal path from x .

4.3 Weighted LRTA*

For $\epsilon \geq 0$, the ϵ -weighted LRTA*, or ϵ -LRTA* for short, is a modification of LRTA* algorithm obtained by relaxing the requirement for initial heuristic function $h(\cdot)$ to be only ϵ -admissible.

Since original LRTA* is identical to 0-weighted LRTA*, this definition subsumes LRTA* as a special case. From the viewpoint of the algorithm, there is no difference between LRTA* and weighted LRTA*². However, one thing should be clarified before we proceed to the discussion of the properties of weighted LRTA*. Recall that in Section 3.5, we saw a variation of LRTA* which we called max-version LRTA*. This variation used a value update formula (3.19), recited here for convenience,

$$h(x) \leftarrow \max \left\{ h(x), \min_{y \in \text{Chd}(x)} [k(x, y) + h(y)] \right\}. \quad (4.2)$$

¹Note that the optimal path that gives $h^*(x)$ may be terminating at a goal state different from g .

²In the off-line search literature, it was interpreted that the heuristic function itself is part of the algorithm; For instance, the name A* was coined for a special instance of generic “ordered search algorithm” employing an admissible heuristic function [28].

ϵ -LRTA* uses this max-version value update formula instead of original LRTA*'s (2.2). The reason and the effect of this choice is discussed later in Section 4.7.1.

4.4 Completeness of Weighted LRTA*

In this section, we show the completeness of weighted LRTA*. At the outset, we prove weighted LRTA* preserves ϵ -admissibility of states. When initial heuristic function h is ϵ -admissible, it has the following property.

Lemma 4.1 Suppose at time instant t , at least one of the optimal children³ of state x_t is ϵ -admissible. If weighted LRTA* makes a transition from state x_t to state x_{t+1} which is also ϵ -admissible, then, after value-updating accompanying the transition, the state x_t becomes ϵ -admissible as well.

Proof. Among the optimal children of x_t , let $y \in \text{Chd}(x_t)$ be one that is ϵ -admissible. Note that we do not exclude the possibility of $x_{t+1} = y$ here. Since at time t , x_{t+1} is chosen as the successor rather than y ,

$$h_{t+1}(x_t) = k(x_t, y) + h_t(x_{t+1}) \leq k(x_t, y) + h_t(y). \quad (4.3)$$

Since y is ϵ -admissible,

$$h_t(y) \leq (1 + \epsilon)h^*(y) \quad (4.4)$$

Substituting equation (4.4) in equation (4.3) yields

$$\begin{aligned} h_{t+1}(x_t) &\leq k(x, y) + (1 + \epsilon)h^*(y) \\ &\leq (1 + \epsilon)[k(x, y) + h^*(y)] \\ &= (1 + \epsilon)h^*(x), \end{aligned}$$

where the last equality follows from the supposition of y being an optimal child of x . □

³As optimal paths may not be unique, there may be more than one such states satisfying this condition.

Corollary 4.1 If all states have ϵ -admissible initial heuristic estimates, this property is preserved by ϵ -LRTA^{*}; i.e.,

$$h_t(x) \leq (1 + \epsilon)h^*(x)$$

for every state x and time instant $t = 0, 1, 2, \dots$.

Proof. Since the heuristic estimate is initially ϵ -admissible at all states, the supposition of Lemma 4.1 is initially satisfied. By straightforward induction on time t , it follows that the number of ϵ -admissible states are nondecreasing throughout the trial. \square

Lemma 4.2 For each time instant $\tau = 0, 1, 2, \dots$, the following relation (4.5) holds.

$$\sum_{j=1}^{\tau} k(x_{j-1}, x_j) = \sum_{x \in X} h_{\tau}(x) - h_{\tau}(x_{\tau}) - \sum_{x \in X} h_0(x) + h_0(x_0). \quad (4.5)$$

Proof. This lemma is identical to Lemma 3.2. It holds regardless of admissibility of heuristic function. \square

Theorem 4.1 If the state space is finite, weighted LRTA^{*} reaches a goal and terminates after finite number of moves, provided that it is reachable from initial state.

Proof. Let $k_{\min} = \min_{(x,y) \in A} k(x, y)$ be the least action cost in the state space. Then, we have

$$\begin{aligned} \tau k_{\min} &\leq \sum_{j=1}^{\tau} k(x_{j-1}, x_j) \\ &= \sum_{x \in X} h_{\tau}(x) - h_{\tau}(x_{\tau}) - \sum_{x \in X} h_0(x) + h_0(x_0) \\ &\leq \sum_{x \in X} (1 + \epsilon)h^*(x) - \sum_{x \in X} h_0(x) + h_0(x_0). \end{aligned} \quad (4.6)$$

The equality in the second line follows from Lemma 4.2, and the inequality in the last line from Corollary 4.1. Since $|X|$ and $h^*(\cdot)$ are both upper bounded, RHS and consequently LHS, τk_{\min} , are also upper bounded. As the positive cost assumption implies $k_{\min} > 0$, τ is upper bounded. \square

4.5 Convergence of Weighted LRTA*

Weighted LRTA* is, in general, not convergent in the sense of Definition 2.6. In this section, we derive a weaker form of convergence theorem for weighted LRTA*, such that if we perform ϵ -LRTA* repeatedly, the problem solver will eventually continue to traverse only the suboptimal paths whose cost is not greater than a multiplicative factor $(1 + \epsilon)$ of the optimal cost. First, we show an analogous result of Lemma 4.1 such that *ϵ -correctness* instead of ϵ -admissibility propagates to the parent state through the value-updating accompanying the problem solver's move.

Lemma 4.3 Suppose one of the optimal children from state x_t is ϵ -admissible. If weighted LRTA* makes a transition to an ϵ -correct state x_{t+1} from state x_t at time t , then after transition, x_t becomes ϵ -correct.

Proof. Among the optimal children of x_t , let y be one of the ϵ -admissible one. Since state x_{t+1} is also ϵ -admissible, we have

$$h_{t+1}(x_t) \leq (1 + \epsilon)h^*(x_t). \quad (4.7)$$

by Corollary 4.1. Hence it remains to show $h^*(x_t) \leq h(x_t)$. By the supposition that state y is an optimal child of x_t ,

$$h^*(x_t) = k(x_t, y) + h^*(y) \leq k(x_t, x_{t+1}) + h^*(x_{t+1}) \quad (4.8)$$

Since the successor x_{t+1} is ϵ -correct by supposition,

$$k(x_t, x_{t+1}) + h^*(x_{t+1}) \leq k(x_t, x_{t+1}) + h_t(x_{t+1}). \quad (4.9)$$

Combining formulas (4.8) and (4.9) yields

$$h^*(x_t) \leq k(x_t, x_{t+1}) + h(x_{t+1}) = h(x_t). \quad (4.10)$$

By formulas (4.7) and (4.10), state x_t becomes ϵ -correct after transition on time t . \square

Now we are ready to prove the convergence of ϵ -LRTA*.

First one is the convergence of heuristic estimates, but this is almost immediate from Lemma 4.3.

Theorem 4.2 (Convergence of heuristic estimates) In a finite state space, By repeatedly applying ϵ -LRTA* repeatedly, every state along the path traversed by ϵ -LRTA* will eventually be ϵ -correct.

Proof. Since goal states are initially ϵ -correct, and from Lemma 4.3, a trial of ϵ -LRTA* changes at least one state to ϵ -correct, unless all the states visited by the problem solver are already ϵ -correct. As the state space is finite, the statement of the theorem follows immediately.

Theorem 4.3 (Convergence of trajectory) If we apply ϵ -LRTA* repeatedly, there will be a trial after which the problem solver continues to traverse only ϵ -optimal paths.

Proof. For each state x , since $h(x)$ is nondecreasing with time, and since it is upper-bounded by $(1 + \epsilon)h^*(x)$ by Corollary 4.1, $h(x)$ will eventually converge as the episode proceeds.

Let $\pi = (x_0, x_1, \dots, x_{\tau-1}, x_\tau)$ be a path traversed by the problem solver in a trial after all the heuristic estimates have converged, where $x_0 \in S$ and $x_\tau \in G$. Further let $h(x)$ be the converged heuristic estimate at each $x \in X$. We show that the path π is indeed ϵ -optimal by induction on the length τ of π . It is trivial when $\tau = 0$, so let $\tau > 0$. Since the problem solver has moved from x_{j-1} to x_j , for all $1 \leq j \leq \tau$, we have

$$k(x_{j-1}, x_j) + h(x_j) = \min_{y \in \text{Chd}(x_{j-1})} [k(x_{j-1}, y) + h(y)].$$

From update-formula (4.2),

$$\min_{y \in \text{Chd}(x_{j-1})} [k(x_{j-1}, y) + h(y)] \leq h(x_{j-1}).$$

Combining above two formulas yields

$$k(x_{j-1}, x_j) \leq h(x_{j-1}) - h(x_j).$$

Summing this inequality over $j = 1, 2, \dots, \tau$ yields

$$\sum_{j=1}^{\tau} k(x_{j-1}, x_j) \leq h(x_0) - h(x_\tau) = h(x_0),$$

where the last equality follows from $x_\tau \in G$. Since LHS is the definition of the cost of π , and since RHS is further upper-bounded by $(1 + \epsilon)h^*(x_0)$ by Corollary 4.1, π is ϵ -optimal. \square

Note, however, that the above theorem does not imply the solution cost tends to a *fixed* value not greater than $(1 + \epsilon)$ of the optimal one. Even the state spaces with single initial and goal states are not exempt from this. This phenomenon will be observed in the experiments with the fifteen-puzzle domain we demonstrate in the next section. The theorem guarantees that all the solution costs (which may be oscillating among several different values because of random tie-breaking), fall within a factor $(1 + \epsilon)$ of the optimal cost once the heuristic estimate at every state converges.

4.6 Experiments

To examine the performance of weighted LRTA*, we conducted an experiment with the fifteen-puzzle⁴. Our test bed is a randomly-generated fifteen-puzzle instance depicted in Figure 4.1. We used as the initial heuristic function the sum of the Manhattan distance of each numbered tiles between the initial and the goal states. This instance has the optimal solution cost (length) of 43 units, with the initial estimate of 33.

4.6.1 Single-Trial Problem Solving

Figure 4.2 shows the first trial performance of ϵ -LRTA* with various ϵ , applied to the fifteen-puzzle instance. The horizontal axis of the figure shows the parameter ϵ , and the vertical axis shows the average solution cost, or number of moves in this case, over 1000 independent trials. The reason for taking the

⁴For extensive survey and analysis of the eight-tile puzzle and its extensions, see [1].

14		7	3
15	1	2	8
4	10	5	11
9	12	13	6

(a) Initial state

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

(b) Goal state

Figure 4.1: Workbench: fifteen-puzzle

average is to compensate for the effect of random tie-breaking. We can see from this figure that adequately overestimated heuristic function may improve the first-trial performance. Note that at $\epsilon = 0.5$, the average solution cost reduces to half that of LRTA^* ($\epsilon = 0$). Although we do not include the material here, we have also verified that almost identical results are observed on the hundred instances of randomly generated fifteen-puzzle listed in [22], with best results obtained at the values of ϵ ranging from 1.2 to 1.6.

4.6.2 Repeated Problem Solving Trials

Next, to measure the learning performance of ϵ - LRTA^* , we applied it repeatedly to the same fifteen-puzzle instance, varying the value of ϵ .

In this experiment, the original LRTA^* (which is identical to ϵ - LRTA^* with $\epsilon = 0$) could not find the optimal solution within a reasonable amount of time and resource⁵. Thus we needed IDA^* to compute the optimal solution cost of the problem. In contrast, it was possible to observe the convergence of ϵ - LRTA^* with $0.2 \leq \epsilon \leq 1.0$.

Table 4.1 shows the data obtained from the final results of the experiment; the value of ϵ vs. the number of trials required for solutions to converge,

⁵Our implementation of LRTA^* is written in C and run on a Linux box with an Intel Celeron 500MHz processor and 256MB of memory.

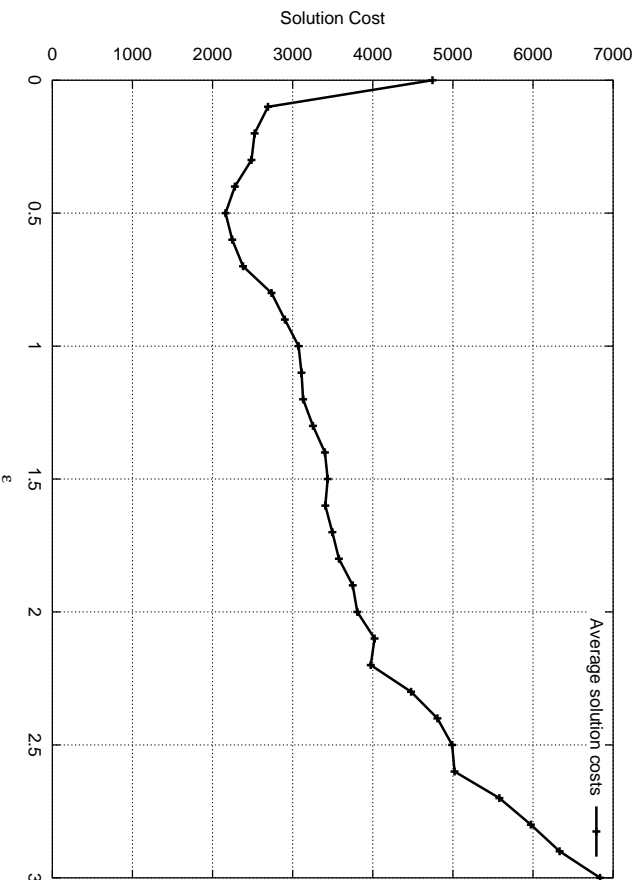


Figure 4.2: Single-trial e -LRTA* in fifteen-puzzle: number of moves

the number of expanded states, and the converged solution cost. In the table, the row of $\epsilon = 0.0$ (original LRTA*) does not hold the data after final convergence, but merely indicates the number of expanded states obtained after 10000 trials (thus a ‘+’ sign after the numbers, such as ‘10000+’ in the trial column), for the reason stated previously.

Compared with the single-trial performance which showed only modest improvements, the effect of e -LRTA* is dramatic in this experiment. Small ϵ value (0.2 through 0.6) significantly (about order of 10^2 and possibly more⁶) reduces both the number of trials and the amount of memory consumption required for convergence.

On the other hand, as ϵ increases, the quality of converged solution tends to be worse. In the above results with $\epsilon = 0.6$ and greater, the solution cost oscillates among several values. This is due to the use of max-version update

⁶We cannot tell the exact order here, as LRTA* could not find optimal solution in our experiment. The braced number 43 in the table is obtained by IDA*.

Table 4.1: Repeated ϵ -LRTA* in fifteen-puzzle: results

ϵ	# of Trials	# of Expanded States	Solution Costs
0.0 (LRTA*)	10000+	39924343+	N/A (43)
0.2	4404	1757537	43
0.4	976	560457	43
0.6	4213	1444669	43,45
0.8	18508	2527144	45,51
1.0	39990	3922826	45,49,51,53

formula, but as Theorem 4.3 tells, all the solutions are guaranteed to fall within the multiplicative factor $(1 + \epsilon)$ of optimal solution, which is 43.

4.7 Variations

Several variations of ϵ -LRTA* are conceivable and are discussed here.

4.7.1 Use of Naive Value-Update Rule

Unlike original LRTA*, the ϵ -LRTA* algorithm described above uses the max-version update formula (4.2). If we used LRTA*'s original update formula (2.2) instead, the nondecreasing property of heuristic estimates would be lost. This would also make the convergence proof difficult, as nondecreasing property plays the fundamental role in the proof.

While we were working on this, we expected at first that the proof technique of Section 3.3 might help, but it was not at all straightforward; the cost of the converged solution cannot be told a priori when initial heuristic estimates are overestimating, while the proof technique relies on the convergence of a series of nonnegative terms towards explicit limit. Even worse, if the heuristic function violates admissibility, the converged solution need not be optimal. This means that there is a possibility that the problem solver traverses a path whose cost is less than the final solution during convergence,

violating the requirement that the difference between the solution cost of each trial and the converged solution be nonnegative (See formula (3.15)).

However, a weaker but still meaningful performance guarantee is obtainable by extending the proof technique. For brevity, we assume the singleton set $S = \{x_0\}$ of initial states. Note that the proof relies on Lemma 3.5 (and equation (3.14)), which holds regardless of initial admissibility.

Theorem 4.4 If initial heuristic function is ϵ -admissible, the average solution cost will eventually fall within a factor $1 + \epsilon$ of the optimal cost.

Proof. Let $w = 1 + \epsilon$. From invariant (3.14), we have

$$\begin{aligned}
& \sum_{i=1}^n [c(i) - wh^*(x_0)] \\
&= \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) + \sum_{i=1}^n h_0^i(x_0) - \sum_{i=1}^n wh^*(x_0) \\
&\leq \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) + \sum_{i=1}^n wh^*(x_0) - \sum_{i=1}^n wh^*(x_0) \\
&= \sum_{x \in X} h_{\tau(n)}^n(x) - \sum_{x \in X} h_0^1(x) \\
&\leq \sum_{x \in X} wh^*(x) - \sum_{x \in X} h_0^1(x), \tag{4.11}
\end{aligned}$$

for every n , where Corollary 4.1 is used twice to derive inequalities. Note that in this case, the terms in the series at the LHS, i.e. $c(i) - wh^*(x_0)$ need not be nonnegative. Now we divide both sides of (4.11) with the number n of the trials to obtain

$$\frac{1}{n} \sum_{i=1}^n [c(i) - wh^*(x_0)] \leq \frac{1}{n} \left[\sum_{x \in X} wh^*(x) - \sum_{x \in X} h_0^1(x) \right],$$

where LHS is the difference between the average cost of solutions and $wh^*(x_0)$. Since RHS tends to 0 as the number n of trials increases, it follows that the average solution cost will eventually be no worse than $wh^*(x_0)$. \square

The theorem is too weak to imply that the solution cost converges to some fixed real p satisfying $h^*(x_0) \leq p \leq (1 + \epsilon)h^*(x_0)$, even if the sets S and G of initial and goal states are both singleton; for instance, we cannot exclude the possibility of the solution costs oscillating between $h^*(x_0)$ and some quantity greater than $(1 + \epsilon)h^*(x_0)$. Nevertheless, the above theorem guarantees, at least, that the problem solver will never fail to traverse a path whose cost is less than or equal to $(1 + \epsilon)h^*(x_0)$ in some future trial.

4.7.2 Measuring the Error Additively

It is possible to measure the amount of possible overestimation of each heuristic estimates *additively* relative to the correct cost, instead of multiplicatively as in ϵ -LRTA*. Such reformulation also establishes a different bound on the converged solutions. We begin with a definition.

Definition 4.4 If the following condition is met, heuristic estimate $h(x)$ of state x is said to be *e-additively admissible*.

$$h(x) \leq h^*(x) + e. \tag{4.12}$$

The state x satisfying formula (4.12) is called an *e-additively admissible state*. A heuristic function $h(\cdot)$ is *e-additively admissible* iff it assigns *e-additively admissible* heuristic values to all non-goal states, and 0 to each goal state.

For $e \geq 0$, *e-additive LRTA** is a variation of max-version LRTA* that uses *e-additively admissible* initial heuristic function instead of admissible one. In particular, when $e = 0$, *e-additive LRTA** is identical to (max-version) LRTA*, which amounts to another generalization of LRTA*.

Below, we show the convergence property of *e-additive LRTA**. Completeness is obvious and thus omitted, since by making ϵ sufficiently large, every *e-additively admissible* heuristic function can be conceivable as ϵ -admissible. Concerning the *e-additive admissibility*, the following lemma holds.

Lemma 4.4 When all the children of state x are *e-additively admissible*, and if problem solver moves from x , as the effect of the update of heuristic value by formula (4.2) accompanying the move, x becomes *e-additively admissible*.

Proof. By assumption, for each child state y of state x , $h(y) \leq h^*(y) + e$. Hence, if we let $h(x)$ be heuristic value of x after update,

$$\begin{aligned} h(x) &= \min_{y \in \text{Chd}(x)} [k(x, y) + h(y)] \\ &\leq \min_{y \in \text{Chd}(x)} [k(x, y) + h^*(y) + e] \\ &\leq \min_{y \in \text{Chd}(x)} [k(x, y) + h^*(y)] + e. \end{aligned}$$

Therefore, substituting equation (2.1) yields

$$h(x) \leq h^*(x) + e.$$

or, after update, x is e -additively admissible. \square

Corollary 4.2 If for all state, heuristic values are initially e -additively admissible, this property is preserved by e -additive LRTA*.

The notions of e -additively optimal path, and e -additively correct heuristic estimate and state are defined analogously to their multiplicative counterparts.

Definition 4.5 The path $\pi = (x_0, x_1, \dots, x_{n-1}, x_n)$, from state $x = x_0$ to a goal state $x_n \in G$ is e -additively optimal iff following inequality is satisfied.

$$\sum_{i=0}^{n-1} k(x_i, x_{i+1}) \leq h^*(x) + e.$$

Definition 4.6 The heuristic estimate $h(x)$ of state x is said to be e -additively correct if the following inequality is satisfied.

$$h^*(x) \leq h(x) \leq h^*(x) + e.$$

The state x satisfying the above formula is called an e -additively correct state.

Theorem 4.5 (Convergence of trajectory) If we repeatedly apply e -LRTA* to the same problem, there will be a trial after which the problem solver continues to traverse only e -additively optimal paths.

Proof. By Corollary 4.2, for each state x , $h(x) \leq h^*(x) + e$, and by the definition of update formula (4.2), $h(x)$ is nondecreasing. It follows that if we repeat problem solving trials, all the heuristic estimates will eventually converge.

Suppose $\pi = (x_0, x_1, \dots, x_{n-1}, x_n)$, where $x_0 \in S$ and $x_n \in G$, be the path traversed by the problem solver after convergence of all the heuristic estimates. It is possible to show that the cost of π is at most $h(x_0)$, i.e,

$$\sum_{i=0}^{n-1} k(x_i, x_{i+1}) \leq h(x_0),$$

by similar argument as Lemma 4.3. By e -admissibility of x_0 , the RHS is upper-bounded by $h^*(x_0) + e$, and hence π is indeed e -additively optimal. \square

4.8 Discussion

ϵ -LRTA* and (max-version) LRTA* differ only in the class of heuristic function they use and are algorithmically identical. In particular, ϵ -LRTA* subsumes LRTA* when $\epsilon = 0$.

Pohl's weighted off-line search (heuristic path algorithm) [8, 31, 32] is an off-line heuristic search algorithm that allows to find suboptimal solution⁷.

If LRTA* is a real-time counterpart of the off-line search algorithm A*, ϵ -LRTA* is an online version of Pohl's weighted off-line search. However, conceptually similar as they may be, it should be noted that the real-time and off-line search rest in completely different search models and thus their properties do not necessarily transfer over one another. For example, in LRTA*, the set of expanded states after convergence need not be identical to that of A*. Furthermore, in real-time search, even if we search by satisfying

⁷Weighted off-line search uses heuristic estimate function $f(x) = (1 - w)g(x) + wh(x)$, where $g(x)$ is an upper-bound of the minimum cost from the initial state to x , and $h(x)$ is the admissible heuristic function as in real-time search. This is equivalent to $f(x) = g(x) + Wh(x)$ with $W = w/(1 - w)$. It is proven that by setting $W = (1 + \epsilon)$ in this equation, suboptimal solution with error at most ϵ will be found. Furthermore, it was empirically reported that the amount of search dramatically reduces.

local consistency, completeness is not guaranteed in an infinite state space [23]. Therefore, although weighted off-line search and repeated application of ϵ -LRTA* achieve the same goal of relaxing optimality requirements with similar performance guarantee, their behavior is totally different and is not evident from the property of one another.

Harris’s bandwidth search [9] is another version of off-line heuristic search algorithm that makes use of overestimated heuristic functions. The difference between Harris’ bandwidth search and Pohl’s heuristic path algorithm is that while the latter composes heuristic function by multiplying $(1 + \epsilon)$ to admissible heuristic function, the bandwidth search assumes that in every state, the overestimated heuristic value is within some constant additive factor e of exact values (i.e. $\forall x. h(x) \leq h^*(x) + e$). We showed that composition of similar extension to LRTA* is also possible.

4.9 Summary

Tolerating suboptimal solution is often indispensable, especially if finding optimal solution for the problem is known to be inherently intractable. For example, finding optimal solution of sliding-tile puzzles (extension of the fifteen-puzzle to $n \times n$ boards) is NP-complete [33], although it is considered to be a “toy” problem.

The results obtained in this chapter are summarized as follows.

- The use of overestimated, or inadmissible heuristics, often improves the learning performance of real-time search, at the sacrifice of possible degradation of converged solution.
- It is possible to predict how much the quality of converged solution may deteriorate if one knows the amount of overestimation of the initial heuristic function.
- The amount of overestimation can be measured by two means, multiplicatively or additively relative to the actual cost function $h^*(\cdot)$, each resulting in different measure on the degradation of solution quality.

Chapter 5

Controlling the Convergence Process

5.1 Introduction

In this chapter, we discuss the performance of real-time search algorithms during convergence. Although there have been a number of literature on real-time search, most of them are concerned with the first-trial performance, and did not pay much attention to the learning process. A few exceptions include:

- Korf in his pioneering paper on real-time search [23] provides a proof of asymptotic convergence of LRTA* to an optimal solution.
- Koenig[19] reviews Korf's proofs for LRTA* and analyzes the worst-case complexity of LRTA*'s convergence property as well as completeness.
- Mizuno and Ishida [26] empirically compare the learning performance of RTA*, LRTA* and Local Consistency Maintenance search [30] in the simulation of robot-arm manipulation.

The first two are concerned with the asymptotic behavior of the search, and hence do not answer in what way the solution costs converge to the optimal

value. The last reports the instability of learning process. This is the issue addressed in this chapter. In particular, when LRTA* is repeatedly applied, the following phenomena are observed.

- The algorithm tries to find *every* optimal path.

Even after a suboptimal solution with satisfactory quality is found, the algorithm tries to find the optimal solution. What is worse, when there are more than one optimal solutions, it continues to find *all* the optimal paths even after an optimal solution is obtained.

Since the algorithm maintains only the lower bounds of the optimal distance to the goal, the algorithm cannot decide whether the traversed path is optimal or not.

In uncertain environments to which real-time search is appealing, it is not so important to obtain an optimal path, let alone to find all the optimal paths. The problem is that the algorithm has no means to decide when and where to stop exploration.

- It does not guarantee the continuous improvement of the solution quality.

Since the algorithm uses admissible heuristic function, the estimates in the state space are initially much smaller than their optimal values. Consequently, as the learning progresses, the estimates in the explored area increase. This makes the problem solver more directed towards unexplored area, where the estimates remain its original smaller values¹. As a result, it frequently traverses much costly paths than the previous trials.

What is guaranteed by the convergence theorem is merely the eventual convergence to the optimal solution, and the stability of the solution quality during the convergence is out of its scope.

¹The similar phenomenon is observed in reinforcement learning literature [27] (see also [20]), in which it is termed as “optimism in the face of uncertainty.”

To overcome these deficiencies of LRTA* we present a new real-time search method. The algorithm, which we call *real-time search with upper bound δ* , or *δ -search* for short, is intended to control the amount of exploration by maintaining a new heuristic function. This new heuristic function supplies the problem solver the *upper bounds* of the exact costs. Such upper bound estimates are typically obtainable after the first problem-solving trial, and approach towards their exact costs through the repeated problem solving.

We will demonstrate the effectiveness of the algorithm by running it in a randomly-generated gridworld.

5.2 Motivation

5.2.1 Learning Performance of LRTA*

To clarify the motivation of the present chapter, we report the results of an experiment we conducted to evaluate the learning performance of LRTA*.

In this experiment, the state space is a 100×100 (10,000 states) gridworld depicted in Figure 5.1. In this state space, 35% of the states, which are chosen at random, are occupied with obstacles. The problem solver is allowed to move south, north, east, or west of its current state, unless it is blocked by an obstacle. The problem solver's unique initial and goal states (indicated by s and g in Figure 5.1, respectively) are 100-unit apart measured by the Manhattan distance. The exact cost of the optimal path from s to g is 122 units.

Figure 5.2 shows a typical episode of LRTA* applied to the gridworld of Figure 5.1 using Manhattan distance initial heuristics. The vertical axis of the figure measures the solution cost² and the horizontal axis displays the number of problem solving trials. From Figure 5.2, we observe the following two characteristics of LRTA*'s learning performance.

1. Even after a near-optimal solution is found, the problem solver con-

²In our gridworld, this equals to the solution length, or the number of actions performed by the problem solver to reach the goal.

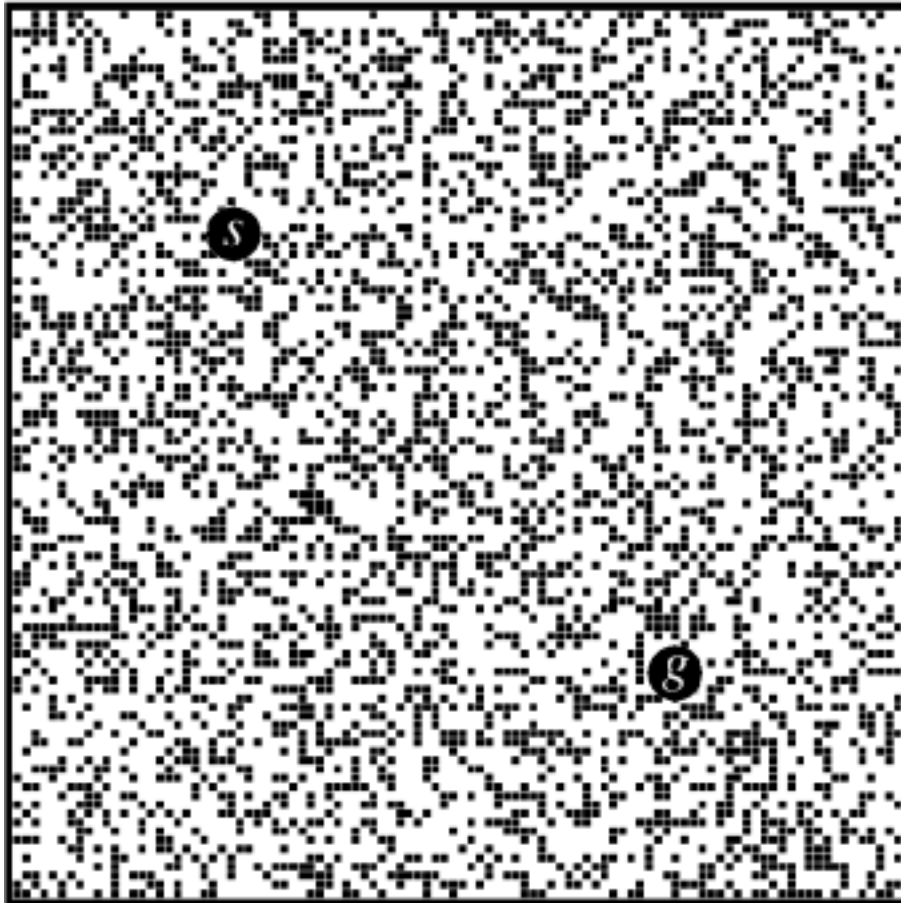


Figure 5.1: Gridworld with 35% random obstacles

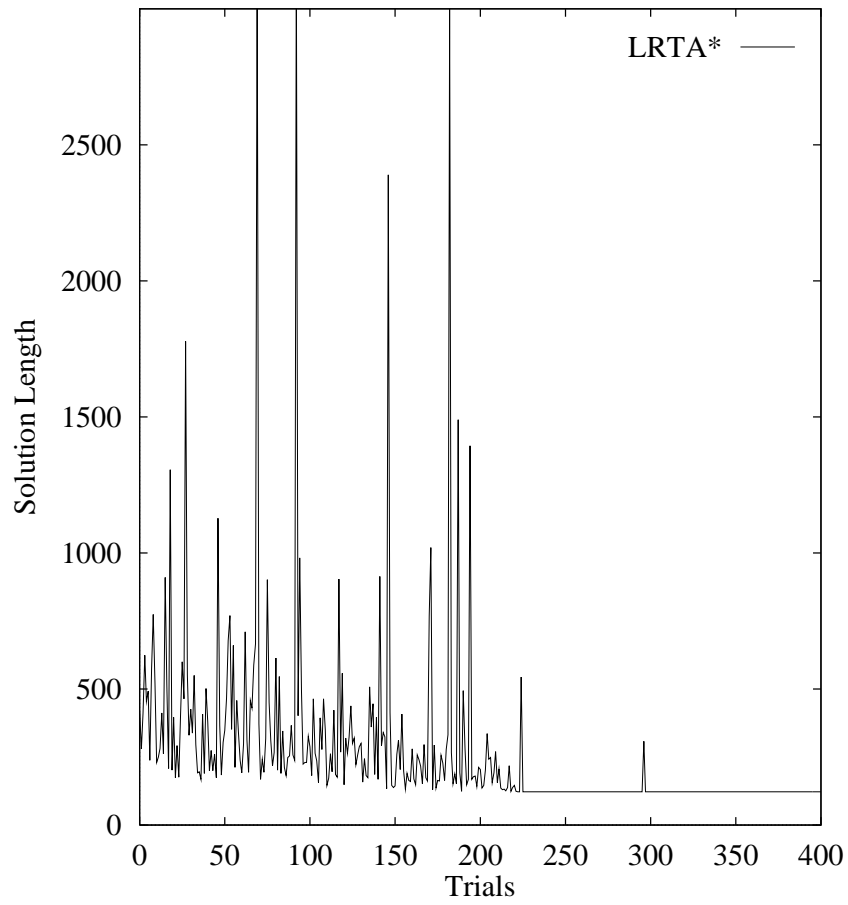


Figure 5.2: LRTA* in gridworld: a typical episode

tinues to explore the state space trying to find better solutions. From Figure 5.2, we can observe that it frequently happens that inferior paths are traversed even after near-optimal solution is found in the previous trial.

2. The quality of solution may suddenly deteriorate even in the later stage of convergence process. Figure 5.2 shows the number of moves is totally unstable during convergence.

5.2.2 Effect of Weighted LRTA* on Solution Stability

From the experimental results of Section 4.6 in the fifteen-puzzle domain, tolerance of suboptimal solution may often reduce the amount of exploration during the convergence, leading to the faster convergence at the sacrifice of the converged solution quality. This suggests that it is possible to use weighted LRTA* to overcome the first problem of the previous section.

Weighted LRTA* with multiplicative factor ϵ (ϵ -LRTA*) is repeatedly applied to the same gridworld of Figure 5.1 with various ϵ . The results are shown in Figures 5.3 and 5.4. Figure 5.3 plots the solution costs, and Figure 5.4 plots the total number of expanded states since the beginning of episode, which is roughly proportional to the amount of memory required to store heuristic estimates. These figures were created by averaging 50 independent episodes (or convergence processes), trial by trial, to compensate for the effects of random tie-breaking and to smooth the graph to make it easy to grasp their tendencies. It should be noted that in each episode, the performance graph is far more jagged than the averaged graph; compare the graph for LRTA* of Figure 5.3 (averaged) with that of Figure 5.2 (single episode).

Each graph shows the cases of $\epsilon = 0, 0.2, \text{ and } 0.5$. As mentioned earlier, at the extreme case of $\epsilon = 0$ the algorithm reduces to ordinary LRTA*. We can see from these graphs:

- As ϵ increases, the number of expanded states decreases. Thus, even in gridworld domain as well as the fifteen-puzzle, introduction of ϵ -admissible heuristics is effective to reduce the amount of computation

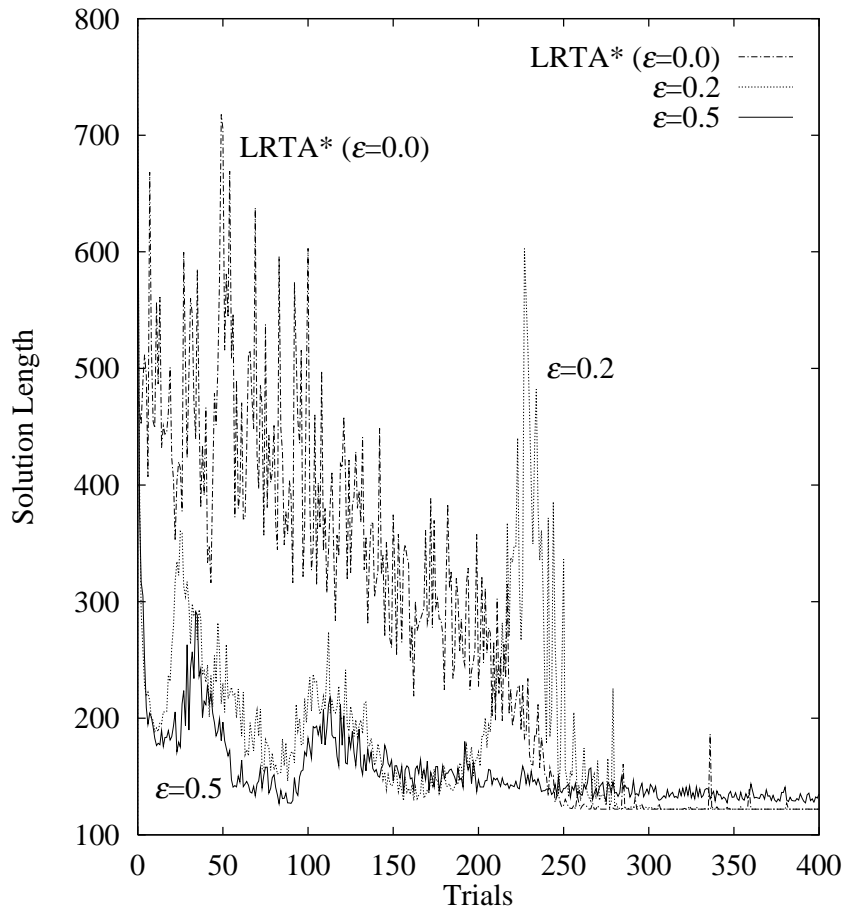


Figure 5.3: Repeated ϵ -LRTA* in gridworld: number of moves

necessary for convergence.

- On the other hand, as ϵ increases, it becomes harder to obtain optimal solutions. As the analysis in Chapter 4 shows, the converged solution may be worse than the optimal solution by at most a factor ϵ . For instance, Figure 5.3 shows that when $\epsilon = 0.5$ the solution cost does not convergence to optimal solution, but indeed fall within the factor 1.5 of it.
- As ϵ increases, the average solution costs decrease dramatically during

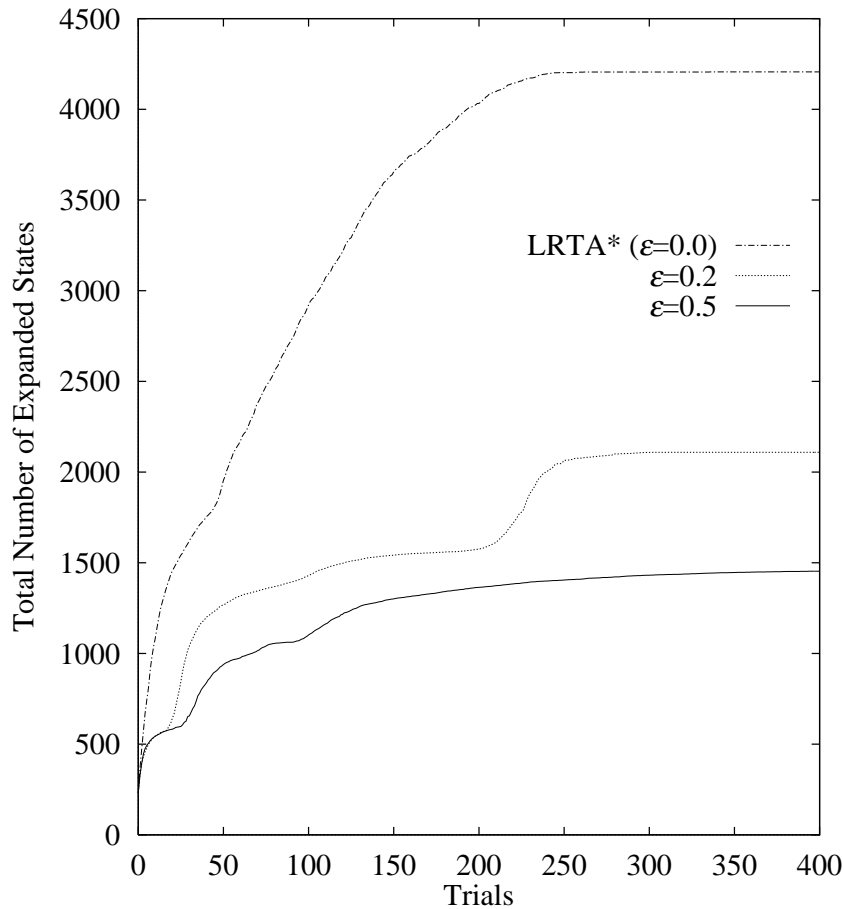


Figure 5.4: Repeated ϵ -LRTA* in gridworld: number of expanded states

the convergence. However, in the case of $\epsilon = 0.2$, for instance, the solution costs often increase drastically all of a sudden after 200 trials. This is because the problem solver at this point decided to seek for another solution. This behavior is not specific to ϵ -LRTA*, and can be seen in original LRTA*, but in either case, the solution stability is still not guaranteed in the convergence process.

To summarize, the use of overestimated initial heuristic function (with adequate parameter ϵ) may reduce the amount of computation required for

convergence, but it does not contribute to the stability of the solution quality during the convergence. We tackle this problem of instability of convergence process, in the rest of the chapter.

5.3 Preliminaries

Let the state space be (X, A, k, S, G) as defined in Section 2.1; X is the set of the states, A is the set of actions, $k(\cdot, \cdot)$ defines the cost of every action in A , and S and G are the sets of initial and goal states. Furthermore, this chapter makes the following additional assumptions.

- Let the sets S and G be singleton, and denote the elements of each by s and g , respectively; i.e., $S = \{s\}$ and $G = \{g\}$.
- Let every action be *reversible*; That is, if $k(x, y)$ is defined for some states x and y , then $k(y, x)$ is also defined (i.e., $(x, y) \in A$ implies $(y, x) \in A$ corresponding to the underlying graph (X, A) being undirected) and furthermore $k(x, y) = k(y, x)$.

To emphasize that every action is reversible, we use the term *adjacent* (or *neighboring*) states instead of *parent* or *child*; for instance, we say x is adjacent to y (or vice versa) to emphasize that both actions (x, y) and (y, x) are in the set A and $k(x, y) = k(y, x)$ holds.

Not all the problem domains satisfy these assumptions, but typical AI benchmark problems such as the gridworld and the sliding-tile puzzle fall into this category. The effect of these additional assumptions are discussed in Section 5.7.

5.4 Real-Time Search with Upper Bounds

5.4.1 Introducing Upper-Bound Heuristics

In this section, we propose the method to overcome the instability of solution quality. The proposed algorithm, which we call δ -search, uses two heuristic

functions to guide the problem solver; in addition to the ordinary admissible heuristic function $h(\cdot)$, we introduce another heuristic function $u(\cdot)$. While $h(x)$ gives a lower bound of exact cost $h^*(x)$ of each state x , the estimate $u(x)$ gives an *upper bound* of $h^*(x)$, and hence is generally inadmissible.

Let us see how this new heuristic estimates are maintained. First, as the initial values, $u(x) = \infty$ for every non-goal state x , and $u(g) = 0$ at the goal state g . Based on Bellman's optimality equation (2.1), we use the following value update operation.

$$u(x) \leftarrow \min\{u(x), \min_{y \in \text{Chd}(x)} [k(x, y) + u(y)]\}. \quad (5.1)$$

Thus, the upper-bound heuristic estimates start from infinity and are non-increasing by nature.

Definition 5.1 The set of *safe children* of state x with respect to parameter θ , denoted by $\text{Safe}(x, \theta)$, is defined as follows.

$$\text{Safe}(x, \theta) = \{y \in \text{Chd}(x) \mid k(x, y) + u(y) \leq \theta\}. \quad (5.2)$$

This definition states that even if we made a transition from x to one of the members of $\text{Safe}(x, \theta)$, we would still be able to reach a goal within the cost of θ , by following the least of the upper-bound heuristic estimates. The idea behind δ -search is that the problem solver avoids transition to non-safe children, trying to maintain the guarantee of reaching a goal by the predetermined deadline.

5.4.2 The δ -Search Algorithm

δ -search is a real-time search algorithm that never makes the solution cost in each trial exceed the initial upper-bound heuristic estimate of the initial state by a multiplicative factor $(1+\delta)$. Such performance guarantee is realized with the help of upper-bound heuristic function.

The pseudo code of δ -search is depicted in Figure 5.5. The algorithm repeats the cycle of value updates of heuristic estimates and action selection in Step 4, just as in LRTA*. The value-update rules (5.4) and (5.5)

1. Initialize for each state $z \in X$, $h(z)$ with some admissible heuristic function $h_0(z)$, and $u(z)$ with ∞ , except for the goal state $g \in G$, where $u(g) \leftarrow 0$.
2. Set the problem solver's current state x to the initial state s ; i.e., $x \leftarrow s$
3. Set threshold $\theta \leftarrow (1 + \delta)u(s)$.
4. Repeat the following steps until $x = g$.

(a) Forward propagation of upper bounds:

For each adjacent state y of the problem solver's current state x , update $u(y)$ by

$$u(y) \leftarrow \min [u(y), k(y, x) + u(x)]. \quad (5.3)$$

(b) Look-ahead and value update:

Update estimates $h(x)$ and $u(x)$ with

$$h(x) \leftarrow \max \left\{ h(x), \min_{y \in \text{Chd}(x)} [k(x, y) + h(y)] \right\}, \quad (5.4)$$

$$u(x) \leftarrow \min \left\{ u(x), \min_{y \in \text{Chd}(x)} [k(x, y) + u(y)] \right\}. \quad (5.5)$$

(c) Action selection:

Move to a child y satisfying

$$y \in \underset{y \in \text{Safe}(x, \theta)}{\text{argmin}} [k(x, y) + h(y)]; \quad (5.6)$$

i.e., $x \leftarrow y$. If there are more than one such state, choose among them arbitrarily (*tie break*).

(d) Recomputation of the threshold:

Set $\theta \leftarrow \theta - k(x, y)$;

Figure 5.5: Pseudo-code of δ -search algorithm

in Step 4b are the the same as LRTA* for lower-bounds, and is identical to (5.1) described earlier for upper-bounds. There is another value-update operation (5.3) in Step 4a, which partially computes (5.1). It propagates the upper-bound estimates towards the children of the current state, in the opposite direction of (5.5). This step is essential for δ -search to work properly; without this step, exploration is suppressed too excessively and it becomes hardly possible to learn better solutions in the subsequent trials³. The algorithm also maintains the threshold θ to be the remaining distance allowed to move in the present trial, initialized at Step 3 and updated accordingly at Step 4d. This threshold value is used for filtering the child states allowed to move, with the formula (5.6) in Step 4c.

Note that initially the algorithm sets initial threshold θ to be $(1 + \delta)u(s)$. It follows that as an extreme case when $\delta = \infty$, condition (5.2) holds unconditionally, and δ -search corrupts to ordinary LRTA*.

5.5 Properties of δ -Search

This section analyzes the properties of δ -search algorithm. At the outset, we introduce some new notations used for the analysis. Let d_j be the cost of the path the problem solver has traversed in the present trial, departing from initial state s and moving j times to reach current state x . Formally, at time instant j (i.e., immediately after j -th move in the trial), let the path traversed so far in the present trial be (x_0, x_1, \dots, x_j) where $s = x_0$ and $x = x_j$. Then,

$$d_j = \sum_{i=1}^j k(x_{i-1}, x_i).$$

Using this notation the value of threshold θ in the algorithm at time j can be rewritten as

$$\theta = (1 + \delta)u_0(s) - d_j. \tag{5.7}$$

³If the state space contained irreversible actions, this update rule would have no effects. See the discussion in Section 5.7.2.

Furthermore, we use $\text{Safe}_j(x, \theta)$ to denote the set $\text{Safe}(x, \theta)$ at the time instant j . This notation is necessary because even for fixed state x and distance θ , $\text{Safe}(x, \theta)$ may change as the upper-bound heuristic values are updated during problem solving.

Note that by the way upper-bound estimates are maintained, on deciding j -th move at state x_{j-1} there should be at least one neighbor y of x_{j-1} satisfying $y \in \text{Safe}_j(x_{j-1}, (1 + \delta)u_0(s) - d_{j-1})$; otherwise, δ -search might not have chosen x_{j-1} as the destination on $(j - 1)$ -st move.

Note also that δ -search is complete, by the following arguments. First of all, by the way upper-bounds are maintained, if $u_0(x)$ is finite, then at any time in the subsequent trials, the state occupied by the problem solver has finite $u(\cdot)$ value. Secondly, if $u(x)$ of the problem solver's current state x is finite, the problem solver can reach the goal by following the states that gives the minimum of $k(x, y) + u(y)$ among the neighbors y of the current state x . As the trial proceeds, the threshold θ decreases, and hence such path will be eventually traversed.

The following theorem states that δ -search contributes to the claimed stability of solutions.

Theorem 5.1 Suppose $\delta \geq 0$, and let the initial upper-bound heuristic $u_0(s)$ of initial state s be finite. Then, the solution cost obtained by performing δ -search is at most $(1 + \delta)u_0(s)$.

Proof. Let the path followed by the problem solver be (x_0, x_1, \dots, x_j) where $x_0 = s$ and $x_j = x$. Furthermore, let the path from x to goal g about to be traversed by the problem solver be $(x_j, x_{j+1}, \dots, x_{\tau-1}, x_\tau)$ where $x_\tau = g$. By equation (5.7), at time instant j , we can write condition (5.2) as

$$d_j + u_j(x_j) \leq (1 + \delta)u_0(s). \quad (5.8)$$

By induction on j over inequality (5.8) and using the fact $u_\tau(x_\tau) = 0$, we have

$$d_\tau \leq (1 + \delta)u_0(s),$$

which shows that the cost of the path traversed by the problem solver in this trial is not more than factor $(1 + \delta)$ of initial upper-bound $u_0(s)$. \square

By this theorem, we see that once we have a finite $u(s) = u_0(s)$ on the initial state s at the beginning of the trial, we can guarantee the solution costs in the subsequent trials to be at most a factor $(1 + \delta)$ of $u_0(s)$. Note that $u(\cdot)$ should be maintained carefully so that traversing towards the minimum of $k(x, y) + u(y)$ of the current state x eventually leads the problem solver to the goal; otherwise, the completeness in each trial is not guaranteed.

At this point, we hit upon a question: with all the extra mechanisms incorporated in the algorithm, does δ -search generally converge to optimal solution? The answer is affirmative for $\delta \geq 2$, but not necessarily so for $\delta < 2$.

The proof for affirmative case is based on the following observation. When $\delta \geq 2$, the problem solver behaves just like repeated LRTA* with reset discussed in Section 3.4.1, until it travels the distance $u_0(s)$. The updates in the subsequent moves are, actually, not needed for convergence at all.

Lemma 5.1 Let $\pi = (x_0, x_1, \dots, x_\tau)$ be the path traversed by the problem solver, where $x_0 = s$ and $x_\tau = g$. The following relation holds for time $j = 0, 1, 2, \dots$

$$u_j(x_j) \leq d_j + u_0(s) \tag{5.9}$$

Proof. The proof is by induction on time j . The base case when $j = 0$ is trivial. Suppose inequality (5.9) holds at some time $j \geq 0$. By update formula (5.3) and by the supposition, we have

$$\begin{aligned} u_{j+1}(x_{j+1}) &\leq k(x_{j+1}, x_j) + u_j(x_j) \\ &\leq k(x_{j+1}, x_j) + d_j + u_0(s) \\ &\leq d_{j+1} + u_0(s), \end{aligned}$$

where the last inequality follows from the definition of d_j and the symmetry of the cost function, i.e., $k(x_j, x_{j+1}) = k(x_{j+1}, x_j)$. \square

Theorem 5.2 When $\delta \geq 2$, the actions executed by δ -search are identical to LRTA* applied to the same state space and the same initial (lower-bound) heuristic function $h_0(\cdot)$, until the cost of the traversed path exceeds $u_0(s)$ and up to the outcomes of tie-breaking.

Proof. Suppose at j -th move, δ -search problem solver is about to make a transition from state x_{j-1} to x_j , whereas LRTA* might choose another child $y \neq x_j$ of x_{j-1} as the destination. By assumption,

$$d_{j-1} + k(x_{j-1}, y) \leq u_0(s). \quad (5.10)$$

Suppose on deciding j -th move, y is excluded from the transition candidates by δ -search; i.e., $y \notin \text{Safe}_j(x_{j-1}, (1 + \delta)u_0(s) - d_{j-1})$. By this and by Lemma 5.1, we have

$$(1 + \delta)u_0(s) < d_{j-1} + k(x_{j-1}, y) + u_j(y) \leq u_0(s) + u_j(y).$$

By assumption that $\delta \geq 2$, we have

$$2u_0(s) \leq \delta u_0(s) < u_j(y). \quad (5.11)$$

On the other hand, by update formula (5.3) for forward propagation of upper bounds, we have

$$u_j(y) \leq k(y, x_{j-1}) + u_{j-1}(x_{j-1}) \leq k(y, x_{j-1}) + d_{j-1} + u_0(s).$$

However, substituting (5.10) in this formula yields $u_j(y) \leq 2u_0(s)$, which contradicts formula (5.11). It follows that $y \in \text{Safe}_j(x_{j-1}, (1 + \delta)u_0(s) - d_{j-1})$. Since LRTA* would chosen y as the successor, $k(x_{j-1}, y) + h(y) \leq k(x_{j-1}, x_j) + h(x_j)$. It follows that δ -search could choose y as the successor as well, unless the tie-breaking strategy have said otherwise. \square

Since the distance $u_0(s)$ is sufficiently large to satisfy the prerequisite for the convergence with reset, i.e., $u_0(s) \geq h^*(s)$, the same argument is applicable⁴.

⁴The updates of $h(\cdot)$ that might occur after traveling $u_0(s)$ distance do not interfere with convergence property. This can be most easily understood with the help of Barto et al.'s convergence proof via reduction to asynchronous dynamic programming [2].

5.6 Experiments

δ -search guarantees the worst solution cost of each trial. Unlike ϵ -LRTA* (except for the trivial case of $\epsilon = 0$), the paths followed by the problem solver will converge to optimal solutions if suitable δ is chosen. Note, however, that since an upper-bound estimate is updated only locally in section 5.4.2, the δ -search is identical to LRTA* until the upper-bound estimate finally propagates from goal state g (where $u(g) = 0$) to initial state s . Therefore, to take advantage of the feature of δ -search, the upper bound of initial state s must be finite and updated as soon as possible to reflect the results of the previous search trials. For this reason, in the following experiments, every time problem solving trial ends, we backtracked the obtained path in the trial, i.e., backward from goal state towards initial state, updating the upper bound heuristic estimates along the path using formula (5.3). Due to this extra mechanism, the solution cost in each trial becomes at most factor $(1+\delta)$ of the cost of the path traversed in the previous trial.

Figures 5.6 and 5.7 display the performance of δ -search (augmented with the extra mechanism of propagating upper bounds between each trials, as mentioned above) repeatedly applied to the gridworld of Figure 5.1, again using the Manhattan distance heuristic function initially as $h(\cdot)$. Figure 5.6 plots the solution cost (number of moves), and Figure 5.7 shows the number of expanded states (memory usage). All the figures contain the results of $\delta = 0, 1, 2,$ and ∞ . When $\delta = 0$, the problem solver is satisfied with the path obtained in the first trial, and does not dare to explore anymore. On the other hand, the behavior of the problem solver when $\delta = \infty$ completely coincides with ordinary LRTA*. From these figures, we can observe the following.

- As the parameter δ decreases, the number of expanded states decreases as well, because the use of upper bounds has successfully restricted the search region. Unfortunately, the restriction of search region does not necessarily mean the speed-up of convergence process. For example, if we compare the case with $\delta = 0$ and the one with $\delta = 2$ in Figure 5.6, $\delta = 2$ requires more trials for convergence. This is because δ -search uses

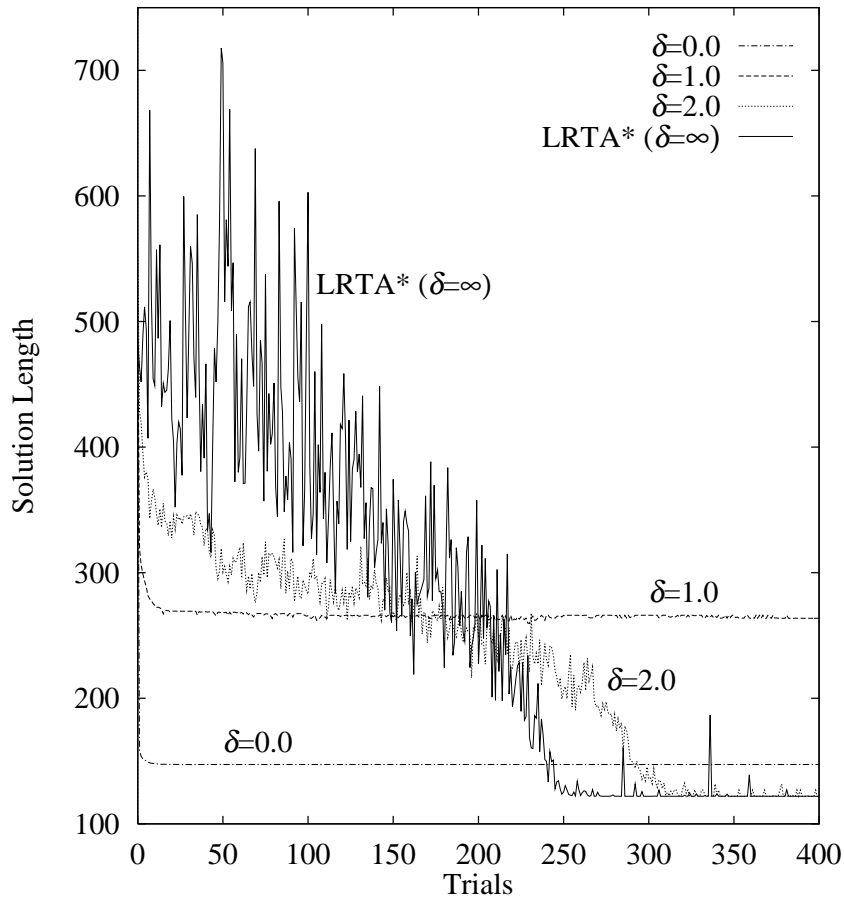


Figure 5.6: δ -search in gridworld: number of moves

upper-bounds not only to restrict the overall amount of exploration, but also to limit the amount of exploration in each trial.

- As δ decreases, the solution quality stabilizes dramatically. On the other hand, as δ decreases, it becomes harder to obtain the optimal solution. Figure 5.6 shows, for example, when $\delta = 1$, the path traversed by the problem solver is not an optimal solution.
- When $\delta < 2$, the greater δ does not necessarily lead to better final solution. For example, the solution quality is worse with $\delta = 1$ than

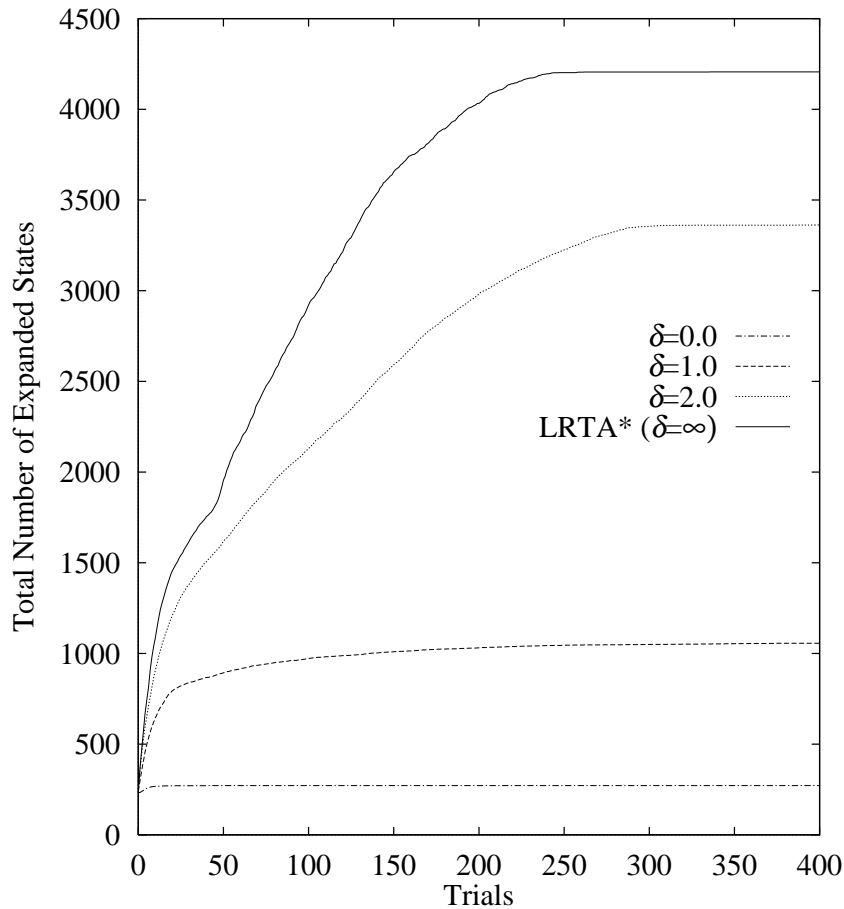


Figure 5.7: δ -search in gridworld: number of expanded states

with $\delta = 0$. This is because the value of δ in this case is not sufficiently large to improve solution quality.

The explanation of the last two results can be given by Theorem 5.2: to find paths better than the one already found, we have to allow the problem solver to traverse at least the cost necessary to cover round-trip of the best path already found, i.e., $\delta \geq 2$.

The weighted LRTA* and upper-bound search are compatible; i.e., these methods can be combined to take advantage of the both. We call the search

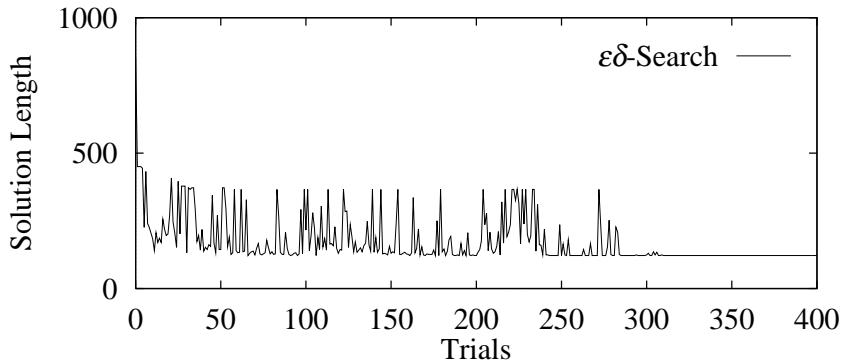


Figure 5.8: $\epsilon\delta$ -search in gridworld: a typical episode

algorithm $\epsilon\delta$ -search. $\epsilon\delta$ -search is identical to δ -search, but an ϵ -admissible heuristic function $h(\cdot)$ is used initially instead of admissible one. $\epsilon\delta$ -search controls the amount of learning by the mechanism of ϵ -search, and intends to stabilize the convergence process by the use of upper-bound heuristic estimates. Figure 5.8 shows the performance of $\epsilon\delta$ -search (with $\epsilon = 0.2$ and $\delta = 2$) in the same gridworld of Figure 5.1. By comparing this graph with the result of LRTA* in Figure 5.2, we can observe the effectiveness of $\epsilon\delta$ -search. It demonstrates that the weighted real-time search and the upper-bound search can be combined to get a more liberty in control of the convergence process.

5.7 Discussion

5.7.1 Multiple Initial and Goal States

The reason we assume a single initial and goal state is just for brevity. We believe it should not be difficult to construct δ -search for the state space with many initial and goal states. For example, even if there exist two or more goals, the situation with single goal state is easily obtained by transforming the state space as follows:

1. Add a virtual goal state (which constitutes the only goal state in the transformed state space).

2. Make virtual edges from each real goal state to the virtual goal state just created.
3. Assign uniform cost to each virtual edge created.
4. Treat real goal states as ordinary non-goal states in the transformed state space.

To be precise, a state space (X, A, k, S, G) with multiple ($|G| > 1$) goal states is transformed into $(X \cup \{g\}, A \cup A', k', S, \{g\})$ where $g \notin X$, $A' = \{(x, g) \mid x \in G\}$ and $k(x, g) = \kappa$ for every $x \in G$, where κ is some constant.

5.7.2 State Space with Irreversible Actions

The other assumption we made specifically in this chapter is the reversibility of all the actions. At a glance, it may seem that δ -search in a state space with irreversible actions incurs only performance degradation but preserves convergent property as in LRTA*. Unfortunately, it does not⁵.

The following simple example exhibits that the δ -search fails to converge to the optimal solution if the state space contains irreversible actions. Consider the state space given by $(X, A, k, \{s\}, \{g\})$, where

- Set of states $X = \{s, g, x, y, z\}$
- Set of actions $A = \{(s, x), (s, y), (y, z), (x, g), (z, g)\}$
- Cost function $k(s, x) = k(s, y) = k(y, z) = 3$, $k(x, g) = k(z, g) = 1$
- The sets of initial states and goal states are both singleton.

Figure 5.9 illustrates the state space described above. The nomenclature is as follows.

- A circle denotes a state, and an edge denotes an action.

⁵I would like to thank one of audience at my presentation at AAAI-96 who made a question on the effect of irreversible actions. It reminded me to reconsider my misjudgment that they incur only corruption to LRTA* at worst but preserve convergence.

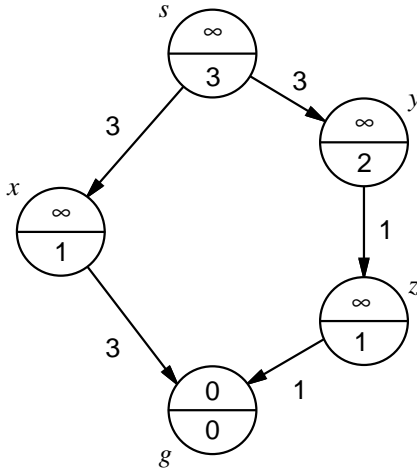


Figure 5.9: State space with irreversible actions

- All the actions in the state space of this figure are irreversible. Transition is possible only to the direction of the arrow head (i.e., it is not possible to move backwards towards the tail of the arrow). For example, the existence of an arrow $s \rightarrow x$ in the figure, together with nonexistence of $x \rightarrow s$, depicts $(s, x) \in A$, and also $(x, s) \notin A$, thus direct transition from x to s is disallowed.
- The number labeling each edge (action) is the cost associated with the action by $k(\cdot, \cdot)$.
- The numbers in the circle (state) display the initial heuristic estimates assigned to the state. The one at the upper side shows the upper-bound heuristic $u(\cdot)$, and the lower one is the lower-bound heuristic $h(\cdot)$.

The optimal path in this state space is (s, y, z, g) with the cost of 5. This $h(\cdot)$ does satisfy the admissibility as well as consistency (or monotonicity), and this phenomenon is not because we used unnatural heuristic values as $h(\cdot)$.

Figure 5.10 illustrates how the problem solving episode proceeds when δ -search is repeatedly applied to the state space of Figure 5.9. Nomenclature

is the same as Figure 5.9, but with the following modifications.

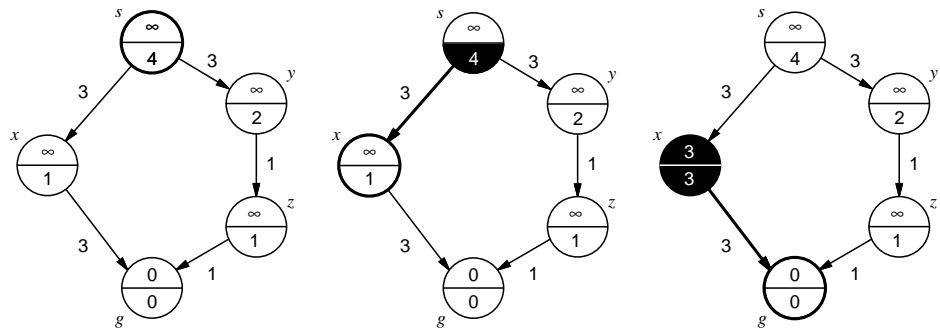
- The circle with bold trim indicates the state currently occupied by the problem solver.
- The numbers in the circle represent the heuristic estimates at the instant, and not the initial values.
- An arrow with bold line indicates selected action, and a dotted arrow indicates an action that are excluded from selection because they violate formula (5.2) of δ -search.
- White numbers indicate that the value is updated and changed.

When second trial terminates (Figure 5.10(f)), $h(\cdot)$ and $u(\cdot)$ values of each state are all convergent. Therefore, in the subsequent trials, the problem solver continues to traverse the path (s, x, g) without performing any updates of the heuristic estimates. Unfortunately, however, the path is not optimal.

All actions are irreversible in the above state space, but even when every action were reversible, if the costs associated with an action and its counter-action were different, the same phenomenon could still occur. For example, consider the state space constructed by adding for each action in the state space of Figure 5.9, a counter-action with a gigantic cost, e.g, uniform cost of 100. In this case, at the first move in the second trial, upper-bound value $u(y)$ of state y is updated with the upper-bound estimate of s . The new value is then $u(y) = k(y, s) + u(s) = 100 + 6 = 106$. Therefore, unless $(1 + \delta)u(s) \geq k(s, y) + u(y) = 3 + 106 = 109$ (i.e., $\delta \geq 17.166666\dots$) the problem solver never tries to move from s to y .

Remark 5.1 Neither irreversible property nor asymmetry cost of actions does not affect off-line search in which the problem solver is not situated in the environment. It is not an obstacle for LRTA* either, since the objective of LRTA* is merely reaching goal state, no matter how long it takes.

In domains with irreversible or actions with asymmetric costs, if one wants to find an optimal solution, one has to try one of the irreversible actions

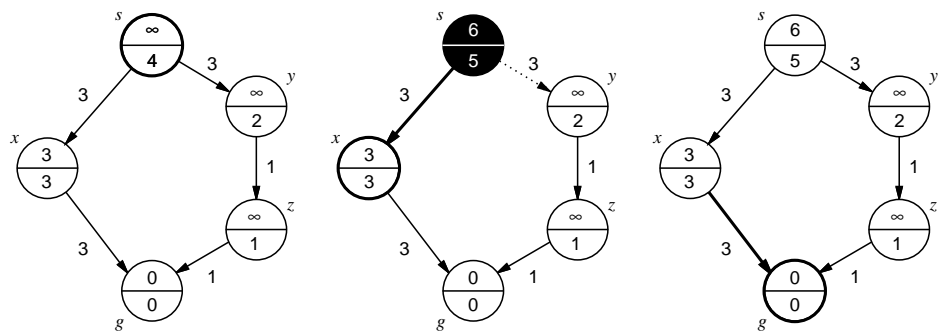


(a) Before the trial

(b) After first move

(c) After second move

First Trial



(d) Before the trial

(e) After first move

(f) After second move

Second Trial

Figure 5.10: δ -search in the state space with irreversible actions

at the risk of losing the stability of solution quality. The above example suggests that the task of improving the solution quality while maintaining stability is fundamentally difficult in such domains. We might need some relaxed definition for *stability*, for example, by allowing the problem solver to perform a small number of reset operation to cut off the trial at its will.

5.7.3 Note on Combining δ -Search and Weighted LRTA*

Besides $\epsilon\delta$ -search depicted in Section 5.6, there is another possibility in the way ϵ -LRTA* and δ -search are combined. This new approach preserves the property of convergence to optimal solution, and still achieves the stability of convergence process that δ -search provides. In this method, we apply ϵ -LRTA* in the beginning of the episode until convergence to find a good reference path to the goal, and then switching to the upper-bound search. In Chapter 4, it was shown that ϵ -LRTA* often arrives at the goal faster than normal LRTA* by adding more depth-first flavor to the behavior of the problem solver. And since the amount of degradation of the converged solution is known analytically in advance, this gives us more liberty to control the convergence process.

5.8 Summary

In this chapter, we showed that with δ -search it is possible to achieve fine-grained control of convergence process. If we use δ -search with appropriate parameter δ , stable performance can be guaranteed after the first trial. This stability is obtained by the virtue of upper-bound heuristics we have introduced. This shows that δ -search acts as an effective balancer of the future investments and the present problem solving efficiency. δ -search is also compatible with ϵ -LRTA* of Chapter 4, which avoids paying too much effort on minor performance improvement.

Chapter 6

Completeness of the Moving-Target Search When Heuristics Overestimate

6.1 Introduction

Real-time search algorithms enjoy the flexibility stemming from their interleaving of planning and action execution. This feature opens the possibility of adapting themselves to changing situations in a way which is impossible for traditional off-line search methods.

The Moving-Target Search (MTS) algorithm, developed by Ishida and Korf [14, 11, 12, 15], is a real-time heuristic search method that takes advantage of this feature to cope with goals changing their locations during the course of problem solving. Under certain reasonable assumptions, the algorithm is *complete*, in the sense that it never fails to capture a goal (target). Although it can naturally be seen as an extension of the stationary goal search algorithm LRTA* [23], they differ in several points. This raises the following questions.

- MTS first compares the current estimate with those from the neighboring states before an update in a similar manner as in max-version

LRTA* (formula (3.19)); the estimate is updated only when one of the latter is greater than the former, hence is forced to be nondecreasing. On the other hand, Korf's original LRTA* lacks this process of comparison, yet is known to preserve completeness. Is it really necessary to force heuristic estimates to be nondecreasing for the completeness of MTS?

- It is known that LRTA* is complete even if the initial heuristic estimates are *inadmissible*, or, overestimating the exact costs. In contrast, Ishida and Korf claim the use of admissible initial heuristic estimates is essential for the completeness of MTS [15, Section X, last sentence]. What makes the difference between the two methods, although they seem to share the same technique for the proof of their completeness?

In this chapter, we extend Ishida and Korf's proof technique, to answer the above questions with the following discoveries.

Proposition 6.1 On updating a heuristic estimate, there is no need to compare the current value with those from the adjacent states.

Proposition 6.2 The completeness is *not* affected by the use of inadmissible heuristics.

6.2 Moving-Target Search

In this section, we review the problem addressed by the MTS algorithm, which we later refer to as MTS problem. We start with the informal and intuitive description of the problem, and later develop it into the formal definition.

6.2.1 Informal Description of the Problem

The MTS problem involves two agents, called a problem solver and a target, both traveling in a deterministic state space of which

1. both the numbers of states and actions are finite;
2. every state is reachable from every state; and
3. all actions are reversible (i.e., has a counter-action) and cost uniformly one.

In addition,

- (4) the state space contains no actions that does not change the location of the agent.

The task of the problem solver, for whom we are responsible, is to catch (i.e., to occupy the same state as) the target who is totally out of control of the problem solver. Stated differently from the standpoint of the algorithm design, our objective is to find a strategy for the problem solver that can effectively and efficiently catch the target. The problem solver has no clue on the behavior of the target; it can only observe the position of the target, which actions are available at their present states, and the outcome of these actions. On the other hand, the target may or may not have the complete knowledge of the problem solver. Since it is hopeless to always catch the target that moves much faster than the problem solver, it is assumed that the problem solver has a speed advantage over the target, however small it might be.

As usual with the real-time search model, the problem solver is required the *real-timeliness*, in the sense that there is a limit on the time and resources allowed to commit to an action. in each state; In this chapter, this constraint will be modeled by restricting the problem solver to the look-ahead of depth one.

6.2.2 Formal Description of the Problem

Now we formulate the MTS problem formally. The present chapter is concerned with the situation with two agents, and therefore requires introduction of new concepts as well as the redefinition of the already introduced concepts.

Let \mathbb{N} be the set of natural numbers including 0, and \mathbb{R} be the set of reals.

The state space in which the agents are traveling is formulated by a graph (X, A) , which is finite, undirected, simple, and connected. Here, X is the finite set of nodes, or *states*, and reflexive relation $A \subset X \times X$ is the edges between states. Since the graph is simple, we identify the set of *actions* with the set A of edges; the state pair $(u, v) \in A$ means there is an action available at state u that makes the agent move to v (we hereby call such action as action (u, v)). It follows that since there is no action that leaves the agent at the same state, there is no loop (i.e., cycle of length one) contained in the graph; i.e., $(v, v) \notin A$ for any $v \in X$.

In such a state space, it is well known that the optimal (or exact) cost function $h^* : X \times X \mapsto \mathbb{R}$ is definable, where $h^*(u, v)$ denotes the shortest distance between states $u, v \in X$.

Generally, the problem solver does not know the optimal cost function h^* , however, it has at hand an inexpensive mechanism to compute an estimate $h(u, v)$ of $h^*(u, v)$ for each state pair (u, v) . This estimate is called *heuristic estimates*, or simply *heuristics*.

When $u, v \in X$ and $w = (u, v) \in X^2$ is a pair of states, we use $h(w)$ and $h^*(w)$ as the shorthands for $h(u, v)$ and $h^*(u, v)$, respectively. If no information is available as to the optimal costs, we can use zero-heuristic function that gives $h(u, v) = 0$ for all $(u, v) \in X^2$.

Admissibility is a property of a heuristic function that it never overestimates the optimal cost. In other words, such heuristics always give optimistic estimates of the distance between the problem solver and the target. Ishida and Korf's MTS algorithm is known to catch up a target when the heuristic function enjoys this property. In the following discussion, however, we do not require the heuristic function be admissible¹.

To measure the quality of (possibly overestimated) heuristics, the notion of admissibility is extended to allow a fixed amount of error. The following

¹In this chapter, the only property required for the heuristic function is that it always return a finite value.

definition involves two similar but slightly different formulation, concerning how the amount of overestimation is evaluated relative to the actual cost.

Definition 6.1 The heuristic estimate $h(u, v)$ of state pair $(u, v) \in X^2$ is said to be *e -additively-admissible* iff it does not overestimate the optimal cost $h^*(u, v)$ up to an additive factor e ; i.e., $h(u, v) \leq h^*(u, v) + e$.

The heuristic estimate $h(u, v)$ is *ϵ -multiplicatively-admissible* iff it does not overestimate $h^*(u, v)$ up to a multiplicative factor of ϵ ; i.e., $h(u, v) \leq (1 + \epsilon)h^*(u, v)$.

Heuristics that assign e -additively-(resp. ϵ -multiplicatively)-admissible value to every state are called e -additively-(resp. ϵ -multiplicatively)-admissible heuristics.

0-additively-(or -multiplicatively)-admissible heuristic function is simply called *admissible*.

6.2.3 Modeling the Speed of Agents

We model the speed advantage of the problem solver in a manner as follows.

A problem solver's move and a target's move are performed *alternately* except when the target skips (or fails to make) its move. The problem solver, on the other hand, never fails to make a move. Formally, we introduce the set of fictitious discrete time instants identified with \mathbb{N} , and partition it into two disjoint sets \mathbb{N}_T of odd numbers and \mathbb{N}_P of even numbers. \mathbb{N}_T corresponds to the set of times at which the target is allowed to move, and \mathbb{N}_P the times when the problem solver is allowed to move.

As a result, we only take into account the situation where the target moves first, unless it skips its first move. This is not a restriction, since the case where the problem solver moves first can be identified with the case where the target skips its first move.

To formulate the target's skipping its move, we further introduce the set $\text{Skip} \subset \mathbb{N}_T$ of times at which the target could have moved but actually would not (or would fail to do so). In this case, both the problem solver and the

target remain at their present states, and no extra deliberation is allowed for the problem solver.

We make the assumption that the target skips its move *ad infinitum*, i.e., the problem solver retains its speed advantage, stated as follows.

Assumption 6.1 (Continuing Skips) The set Skip is infinite. \square

Let $\text{Skip}(t)$ be the set of times at which the target is skipping its move, before and including time t . Formally,

$$\text{Skip}(t) = \{\tau \in \text{Skip} \mid \tau \leq t\}.$$

In particular, we define $\text{Skip}(0) = \emptyset$. This implies

$$\text{Skip}(2n) = \text{Skip}(2n - 1) \quad \text{for all } n \in \mathbb{N}. \quad (6.1)$$

and if we let $\mathcal{I}_{\text{Skip}}$ be the indicator function of Skip , i.e.,

$$\mathcal{I}_{\text{Skip}}(t) = \begin{cases} 1, & \text{if } t \in \text{Skip}, \\ 0, & \text{if } t \notin \text{Skip}, \end{cases} \quad (6.2)$$

then, observe that

$$|\text{Skip}(t)| = \sum_{\tau=1}^t \mathcal{I}_{\text{Skip}}(\tau), \quad (6.3)$$

which we will shortly use in our completeness proof.

Note that Assumption 6.1 implies

$$\lim_{t \rightarrow \infty} |\text{Skip}(t)| = \infty. \quad (6.4)$$

6.2.4 The MTS Algorithm

Ishida and Korf proposed MTS algorithm as a solution to the above mentioned MTS problem. The algorithm, formulated under our previous definition of the problem, is depicted in Figure 6.1. The variables x and y are used to denote respectively the locations of the problem solver and the target, and t is the number of iterations.

1. For each $u, v \in X$, initialize $h(u, v)$ with some initial heuristic function h_0 ; i.e., $h(u, v) \leftarrow h_0(u, v)$.
2. Observe the initial location $x = x_0$ of the problem solver.
3. Observe the initial location $y = y_0$ of the target.
4. Repeat for $t = 1, 2, \dots$, until $x = y$,

- If it is problem solver's turn to move, i.e., $t \in \mathbb{N}_P$, update $h(x, y)$ by

$$h(x, y) \leftarrow \min_{(x, z) \in A} h(z, y) + 1, \quad (6.5)$$

and move to an adjacent state x' s.t.

$$x' \in \operatorname{argmin}_{(x, z) \in A} h(z, y), \quad (6.6)$$

i.e., set $x \leftarrow x'$.

- Otherwise, if the target has successfully moved from y to another state y' , i.e., $t \in \mathbb{N}_P - \text{Skip}$, then update $h(x, y)$ by

$$h(x, y) \leftarrow h(x, y') - 1. \quad (6.7)$$

Set $y \leftarrow y'$.

Figure 6.1: Pseudo-code of the Moving-Target Search algorithm

Remark 6.1 The algorithm of Figure 6.1 contains different value-update formulas from the original version by Ishida and Korf. Their original formulas are

$$h(x, y) \leftarrow \max \left[h(x, y), \min_{(x,z) \in A} h(z, y) + 1 \right]$$

when the problem solver moves, and

$$h(x, y) \leftarrow \max [h(x, y), h(x, y') - 1]$$

when the target moves, instead of formulas (6.5) and (6.7), respectively. These original formulas force the heuristic estimates to be nondecreasing. For the proof of Proposition 6.1, we removed the max operation from the update formulas. The following argument is equally valid with trivial modification when the heuristics are forced to be nondecreasing as in the original paper.

Throughout the subsequent analysis, we use x_t and y_t respectively to denote the locations of the problem solver and the target at the beginning of t -th iteration of Step 4, and $h_t(u, v)$ to denote the heuristic estimate $h(u, v)$ at the same instant². Combining the above notation with the formulas (6.5) and (6.6), we obtain

$$h_t(x_{t-1}, y_{t-1}) = h_t(x_t, y_t) + 1 = h_{t-1}(x_t, y_t) + 1 \quad (6.8)$$

when the problem solver moves at t -th iteration. Similarly, when the target moves at t -th iteration, we have from equation (6.7),

$$h_t(x_{t-1}, y_{t-1}) = h_t(x_t, y_t) - 1 = h_{t-1}(x_t, y_t) - 1. \quad (6.9)$$

In either case,

$$h_t(u, v) = h_{t-1}(u, v)$$

holds for any $(u, v) \neq (x_{t-1}, y_{t-1}) \in X^2$.

²This notation is compatible with the one we have already introduced, viz. the initial states x_0 and y_0 and the initial heuristic function h_0 .

6.3 Completeness of the MTS Algorithm

A heuristic search algorithm is said to be *complete* if the problem solver never fails to arrive at a goal state. In the MTS setting, since the target changes its location during the course of the search, an algorithm is complete if the problem solver eventually catch up a target.

We will establish the completeness of MTS through a series of lemmas. First, we show that updates of heuristic estimates preserve e -additive-admissibility. Lemma 6.1 is the main achievement of this chapter, that generalizes the result of the original MTS paper to the case of inadmissible heuristics.

Lemma 6.1 For any constant $e \geq 0$, if the initial heuristic estimate is e -additively-admissible for every pair of states, then updates of MTS preserve the same property.

Proof. The proof is by induction on time t . The base case is trivial. Assume that at time instant t , we have $h_t(u, v) \leq h^*(u, v) + e$ for each pair (u, v) of states before an update. We will show that $h_{t+1}(u, v) \leq h^*(u, v) + e$ holds for each $(u, v) \in X^2$.

Case (i) Update accompanying a problem solver's move: Suppose t -th iteration belongs to problem solver's turn, i.e., $t \in \mathbb{N}_P$, and the problem solver has moved from state x_t to state x_{t+1} , while the target is at state $y = y_t = y_{t+1}$. In this case, the heuristic estimate affected by the update is only the one at the state pair (x_t, y) . We have

$$\begin{aligned}
 h_{t+1}(x_t, y) &= h_t(x_{t+1}, y) + 1 \\
 &= \min_{(x_t, z) \in A} h_t(z, y) + 1 \\
 &\leq \min_{(x_t, z) \in A} [h^*(z, y) + e] + 1 \\
 &= h^*(x_t, y) + e.
 \end{aligned}$$

Case (ii) Update to compensate for the target's move:

Suppose t -th iteration belongs to the target's turn, and the target has succeeded in moving from state y_t to state y_{t+1} , i.e., $t \in \mathbb{N}_T - \text{Skip}$, while the problem solver remains at state $x = x_t = x_{t+1}$. Since y_t and y_{t+1} is only one unit apart,

$$h^*(x, y_{t+1}) \leq h^*(x, y_t) + 1.$$

It follows that

$$\begin{aligned} h_{t+1}(x, y_t) &= h_t(x, y_{t+1}) - 1 \\ &\leq h^*(x, y_{t+1}) + e - 1 \\ &\leq h^*(x, y_t) + e. \end{aligned}$$

In either case, since the heuristic estimates for any other state pair besides (x_t, y_t) do not change with the t -th move, the statement of the lemma follows. \square

Remark 6.2 The multiplicative counterpart of Lemma 6.1 does not hold. To be specific, ϵ -multiplicative-admissibility may be violated by the updates accompanying target's moves. \square

Now let us define the quantity H_t as

$$H_t = \sum_{w \in X^2} h_t(w) - h_t(w_t),$$

where $w_t = (x_t, y_t)$ is the pair of states occupied by the problem solver and the target at time instant t . This definition is reminiscent of the *heuristic disparity* of the original paper [15], and H_t essentially plays the role of the latter in our restated proof.

Since there is an upper bound of the heuristic estimates, we have an upper bound of the quantity H_t as well.

Corollary 6.1 If the initial heuristic function h_0 is finite (but *not* necessarily admissible), then for all $t \in \mathbb{N}$, H_t has an upper bound s ; i.e.,

$$s = \sum_{w \in X^2} [h^*(w) + e],$$

where e is the maximum amount of overestimation, i.e.,

$$e = \max \left\{ 0, \max_{w \in X^2} [h_0(w) - h^*(w)] \right\}.$$

The following lemma states that the amount of speed advantage of the problem solver accumulates upon the quantity H_t .

Lemma 6.2 The following relation holds for all $t \in \mathbb{N}$.

$$|\text{Skip}(t)| = \begin{cases} H_t - H_0 + 1, & \text{if } t \in \mathbb{N}_T, \\ H_t - H_0, & \text{if } t \in \mathbb{N}_P. \end{cases} \quad (6.10)$$

Proof. The proof is by induction on t . If $t \in \mathbb{N}_T$ (target's move),

- and if $t \in \text{Skip}$, then

$$H_t = H_{t-1}, \quad (6.11)$$

since the target does not move and hence update does not take place.

- Otherwise, $t \notin \text{Skip}$, thus

$$\begin{aligned} \sum_{w \in X^2} h_t(w) &= \sum_{w \in X^2} h_{t-1}(w) - h_{t-1}(w_{t-1}) + h_t(w_{t-1}) \\ &= \sum_{w \in X^2} h_{t-1}(w) - h_{t-1}(w_{t-1}) + h_t(w_t) - 1, \end{aligned}$$

or, equivalently,

$$H_t = H_{t-1} - 1. \quad (6.12)$$

Using the indicator function $\mathcal{I}_{\text{Skip}}$ (see equation (6.2)), equations (6.11) and (6.12) can be restated as a single equation

$$H_t = H_{t-1} - 1 + \mathcal{I}_{\text{Skip}}(t). \quad (6.13)$$

On the other hand, if $t \in \mathbb{N}_P$ (problem solver's move),

$$\begin{aligned} \sum_{w \in X^2} h_t(w) &= \sum_{w \in X^2} h_{t-1}(w) - h_{t-1}(w_{t-1}) + h_t(w_{t-1}) \\ &= \sum_{w \in X^2} h_{t-1}(w) - h_{t-1}(w_{t-1}) + h_t(w_t) + 1, \end{aligned}$$

or,

$$H_t = H_{t-1} + 1. \quad (6.14)$$

Summing equations (6.13) and (6.14) over $t = 1, 2, \dots$, we have

$$H_t = \begin{cases} H_0 - 1 + \sum_{\tau=1}^t \mathcal{I}_{\text{Skip}}(\tau), & \text{if } t \in \mathbb{N}_T, \\ H_0 + \sum_{\tau=1}^t \mathcal{I}_{\text{Skip}}(\tau), & \text{if } t \in \mathbb{N}_P. \end{cases}$$

The statement of the lemma is derived by using equation (6.3). \square

Finally, the following theorem extends the result of Ishida and Korf's completeness theorem to the case of inadmissible heuristics.

Theorem 6.1 MTS is complete, if the initial heuristic estimate is finite (but not necessarily admissible) for every pair of states.

Proof. Assume on the contrary that the problem solver cannot catch the target forever. Then, by Assumption 6.1, $|\text{Skip}(t)|$ increases without bound (see equation (6.4)). Hence, from equation (6.10) we have

$$\lim_{t \rightarrow \infty} H_t = \lim_{t \rightarrow \infty} |\text{Skip}(t)| = \infty,$$

but this contradicts the boundedness of H_t implied by the finiteness of initial heuristic estimates and Corollary 6.1. \square

6.4 Related Work

The use of overestimated heuristics dates back to Pohl [31] who analyzed their effect on the off-line search. The idea was later applied to real-time search for stationary goals by Ishida and Shimbo [16] with the aim of improving the behavior of convergence process. The detailed result of their work, including the completeness property as well, and several extensions are discussed in Chapter 4 of the thesis. The idea of measuring the amount of overestimation additively is due to Harris [9], again for off-line search. Based on decision-theoretic analysis, Koenig and Simmons [21] introduced a beautiful idea to cope with non-determinism in the state space. As the existence of the moving target inherently incurs non-determinism, they briefly mention the application of their min max-LRTA* algorithm in the MTS setting as well. They use the max-version update formula to force consistency and assume admissibility. It should be interesting to analyze whether inadmissible heuristics work with min max-LRTA*.

6.5 Summary

The contribution of this chapter is summarized as follows.

1. On updating the heuristic estimates, there is no need to compare and take the maximum of the current value with those from the adjacent states, insofar as the completeness is concerned.
2. The completeness is *not* affected by the use of overestimated (or inadmissible) heuristics.

The second item extends the class of the heuristic function usable by the Moving-Target Search algorithm. The impact of the first item might seem relatively weaker compared to the second. But this clarified that forcing the estimates to be nondecreasing is not required by the problem setting of MTS, but is introduced for the sake of efficiency. Furthermore, it may have a potential merit when the heuristics are inadmissible, since removing the

max operation may sometimes lead to more accurate heuristic values in that case; even though forcing the heuristics to be nondecreasing can be justified from the standpoint of efficiency when the heuristics are admissible, it may sometimes make the value more inaccurate when they are inadmissible. This might lead to worse converged policy through the repeated problem solving trials.

Chapter 7

Conclusions

7.1 Summary

This thesis centered on two issues of real-time heuristic search that had not been pursued previously.

1. The utility of nonstandard heuristic function.

We studied the effect of heuristic functions that violate two properties that have been taken for granted: admissibility and consistency.

2. The analysis of learning process.

We conducted an analysis of how learning progresses.

In the pursuit of the above topics, the following results were obtained.

1. A new technique for proving the convergence.

We presented a technique for proving the convergence of LRTA* algorithm. The derived proof, being an extension of the proof of the completeness, is not only succinct and intuitive to understand, but also helps to clarify the link between the two important properties of real-time search, namely, the completeness and the convergence.

2. Weighted LRTA* for tolerating suboptimal solutions.

We discovered the previously unknown property of LRTA* algorithm that it can converge to a suboptimal solution instead of optimal one, under overestimated heuristic function. It was also found that it is possible to predict the amount of deterioration of the converged solution if one knows the amount of overestimation of the initial heuristic function.

The experimental result showed that it is also possible to reduce the search effort needed for convergence at the sacrifice of optimality. This result appeals to the application domains in which it is inherently intractable to seek for optimal solution, but suboptimal solution is equally feasible in practice. Most of the applications, even “toy” problems such as the sliding-tile puzzle, fall into this category.

3. The δ -search algorithm for stabilizing the convergence process.

We pointed out the two intrinsic drawbacks of real-time search during the convergence process: failure to balance exploration-exploitation trade-off, and instability of solution quality during convergence.

We showed that these drawbacks can be overcome by using an inadmissible heuristic function along with the conventional admissible one. The new heuristic is used for maintaining the upper bounds of the actual costs to the goal, while the admissible heuristic maintains the lower bounds as in LRTA*. We proved both theoretically and experimentally that δ -search, our new algorithm featuring the above mechanisms, guarantees stable improvement of solution quality during convergence. When this algorithm is combined with weighted LRTA*, we obtain fine-grained control the convergence process.

4. Analysis of the effect of inconsistent or overestimated heuristics in the case of moving target search.

In moving target search (MTS) problem, the state space is completely different from that of single-agent search because of the existence of the target, or, the changing goal.

We resolved the difference between the stationary-goal search algorithm LRTA* and the MTS algorithm by Ishida and Korf on the use of inconsistent heuristic function. In particular, we showed that the MTS algorithm is complete even under inadmissible heuristic function, on the contrary to what was believed. This result contributes to extending the class of heuristic functions usable for moving target search problem.

These results contribute to further improving the flexibility and applicability of real-time search; it is now proven that even nonstandard heuristic functions have a role in real-time search, as they often provide effective means for controlling the learning process.

7.2 Future Directions

We conclude this thesis with the list of possible future research directions.

- General convergence proof that covers the case of inadmissible heuristic function.

Although the proof technique we proposed in Chapter 3 covers the whole class of admissible heuristic function, it fails to prove the convergence when the function is inadmissible. The difficulty lies in that the converged solution cost is not known in advance, as opposed to admissible case.

- A theory that well accounts for the performance improvements achieved by weighted LRTA*.

Although we have demonstrated experimentally that excessive exploration can be suppressed by weighted LRTA*, we do not have the theory that well describes this fact. For example, although the bound obtained from inequality (4.6) is worse than that of LRTA*, adequate increase in the initial heuristic estimates do improve the performance of real-time search in the domains such as the gridworld and the fifteen-puzzle.

- Methodology for controlling the convergence process in more uncertain domains

The state spaces involving irreversible or asymmetric cost actions are intrinsically difficult to obtain a stable learning process in the sense of Chapter 5. It seems that some other criterion is needed to redefine what “stable” performance improvement means. One possible solution would be to incorporate the ideas from computational learning theory, such as mistake-bound online learning [24] and “homing sequence” [34].

- A theory on the effect of heuristic function on real-time search.

Although we have derived several upper bounds on the number of moves performed by real-time search algorithms (e.g., formulas (3.10) or (3.15)) in this thesis, they only give a guarantee for the worst case performance. We need a theory that accounts for how the performance of real-time search is affected by the topologies of the state space and of initial heuristic function.

Bibliography

- [1] Shinji Araya, Taketoshi Momohara, Yoshihiro Okamoto, and Kenji Yamaguchi. Problem space analysis of the 8-puzzle and its complete solution. *Journal of the Japanese Society for Artificial Intelligence*, 11(3):130–137, 1996. In Japanese.
- [2] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, 72(1–2):81–138, 1995.
- [3] Richard E. Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, USA, 1957.
- [4] Dimitri P. Bertsekas. *Dynamic Programming: Deterministic and Stochastic Models*. Prentice Hall, Englewood Cliffs, NJ, USA, 1987.
- [5] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation: Numerical Methods*. Prentice Hall, Englewood Cliffs, NJ, USA, 1989.
- [6] Blai Bonet, Gábor Loerincs, and Héctor Geffner. A robust and fast action selection mechanism for planning. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, pages 714–719, Providence, RI, USA, 1997. AAAI Press.
- [7] Fumihiko Chimura and Mario Tokoro. The trailbrazer search: a new method for searching and capturing moving targets. In *Proceedings of*

- the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, pages 1347–1352, Seattle, WA, USA, 1994.
- [8] Henry W. Davis, Anna Bramanti-Gregor, and J. Wang. The advantages of using depth and breadth components in heuristic search. In *Proceedings of the Third International Symposium on Methodologies for Intelligent Systems*, pages 19–28, Turin, Italy, 1988. North-Holland.
- [9] Larry R. Harris. The heuristic search under conditions of error. *Artificial Intelligence*, 5(3):217–234, 1974.
- [10] P. E. Hart, Nils J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost path. *IEEE Transactions on Systems Science and Cybernetics (SSC)*, 4(2):100–107, 1968.
- [11] Toru Ishida. Moving target search with intelligence. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 525–532, San Jose, CA, USA, 1992. AAAI Press/MIT Press.
- [12] Toru Ishida. Real-time bidirectional search: coordinated problem solving in uncertain situations. *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 18(6):617–628, 1996.
- [13] Toru Ishida. *Real-Time Search for Learning Autonomous Agents*. Kluwer Academic Publishers, Dordrecht, the Netherlands, 1997.
- [14] Toru Ishida and Recharad E. Korf. Moving-target search. In *Proceedings of the Twelfth International Conference on Artificial Intelligence (IJCAI-91)*, pages 204–210, Sydney, Australia, 1991. Morgan Kaufmann.
- [15] Toru Ishida and Richard E. Korf. Moving-target search: a real-time search for changing goals. *IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI)*, 17(6):609–619, June 1995.
- [16] Toru Ishida and Masashi Shimbo. Improving the learning efficiencies of real-time search. In *Proceedings of the Thirteenth National Conference*

- on *Artificial Intelligence (AAAI-96)*, volume 1, pages 305–310, Portland, OR, USA, 1996. AAAI Press.
- [17] Toru Ishida and Masashi Shimbo. Path learning by real-time search. *Journal of Japanese Society for Artificial Intelligence*, 11(3):411–419, May 1996. In Japanese.
- [18] Tommi Jaakkola, Michael I. Jordan, and Satinder P. Singh. On the convergence of stochastic iterative dynamic programming algorithms. *Neural Computation*, 6(6):1185–1201, 1994.
- [19] Sven Koenig. The complexity of real-time search. Technical Report CMU-CS-92-145, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, April 1992.
- [20] Sven Koenig. Exploring unknown environments with real-time search or reinforcement learning. In *Neural Information Processing Systems (NIPS★99)*, 1999.
- [21] Sven Koenig and Reid G. Simmons. Real-time search in non-deterministic domains. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95)*, pages 1660–1667, Montreal, Quebec, Canada, 1995.
- [22] Richard E. Korf. Depth-first iterative-deepening: an optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.
- [23] Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2–3):189–211, 1990.
- [24] Nick Littlestone. Learning when irrelevant attributes abound: a new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.
- [25] László Mérő. A heuristic search algorithm with modifiable estimate. *Artificial Intelligence*, 23:13–27, 1984.

- [26] Norifumi Mizuno and Toru Ishida. Evaluation on learning efficiencies of real-time search. *Journal of the Japanese Society for Artificial Intelligence*, 10(2):306–313, March 1995. In Japanese.
- [27] Andrew W. Moore and Christopher G. Atkeson. Prioritized sweeping: reinforcement learning with less data and less time. *Machine Learning*, 13:103–130, 1993.
- [28] Nils J. Nilsson. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill, New York, NY, USA, 1971.
- [29] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, Reading, MA, USA, 1984.
- [30] J. Pemberton and Richard E. Korf. Making locally optimal decisions on graphs with cycles. Technical report, Computer Science Department, University of California at Los Angeles, Los Angeles, CA, USA, 1992.
- [31] Ira Pohl. First results on the effect of error in heuristic search. In *Machine Intelligence*, volume 5, pages 219–236. Edinburgh University Press, Edinburgh, UK, 1970.
- [32] Ira Pohl. Heuristic search viewed as path finding in a graph. *Artificial Intelligence*, 1:193–204, 1970.
- [33] Daniel Ratner and Manfred K. Warmuth. Finding a shortest solution for the $n \times n$ extension of the 15-puzzle is intractable. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI-86)*, volume 1, pages 168–172, Philadelphia, PA, USA, 1986. Morgan Kaufmann.
- [34] Ronald L. Rivest and Robert E. Schapire. Inference of finite automata using homing sequence. *Information and Computation*, 103(2):299–347, 1993.
- [35] Stuart J. Russell and Eric Wefald. *Do the Right Thing: Studies in Limited Rationality*. MIT Press, Cambridge, MA, USA, 1991.

- [36] Masashi Shimbo and Toru Ishida. Weighted real-time search. Technical Report AI95-43, Institute of Electronics, Information and Communication Engineers (IEICE), Tokyo, Japan, January 1996. In Japanese.
- [37] Masashi Shimbo and Toru Ishida. On the convergence of real-time search. *Journal of the Japanese Society for Artificial Intelligence*, 13(4):631–637, July 1998. In Japanese.
- [38] Masashi Shimbo and Toru Ishida. The completeness of moving-target search with inconsistent heuristics. *Journal of the Japanese Society for Artificial Intelligence*, 14(2):342–348, March 1999. In Japanese.
- [39] Herbert A. Simon. *The Sciences of the Artificial*. MIT Press, Cambridge, MA, USA, third edition, 1996.
- [40] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA, 1998.
- [41] Christopher J. C. H. Watkins and Peter Dayan. Technical note: Q-learning. *Machine Learning*, 8(3–4):279–292, 1992.