Real-Time Search in Non-Deterministic Domains*

Sven Koenig and Reid G. Simmons
Carnegie Mellon University
School of Computer Science
Pittsburgh, PA 15213-3890
skoenig@cs.cmu.edu, reids@cs.cmu.edu

Abstract

Many search domains are non-deterministic. Although real-time search methods have traditionally been studied in deterministic domains, they are well suited for searching non-deterministic domains since they do not have to plan for every contingency – they can react to the actual outcomes of actions. In this paper, we introduce the min-max LRTA* algorithm, a simple extension of Korf's Learning Real-Time A* algorithm (LRTA*) to non-deterministic domains. We describe which non-deterministic domains min-max LRTA* can solve, and analyze its performance for these domains. We also give tight bounds on its worst-case performance and show how this performance depends on properties of both the domains and the heuristic functions used to encode prior information about the domains.

1 Introduction

Real-time (heuristic) search methods, a term coined by Korf [Korf, 1987], interleave search with action execution, limiting the amount of deliberation performed between action executions. After an action has been executed, the deliberation-act cycle is repeated – until a goal state is reached. [Korf, 1993] demonstrated that real-time search methods are powerful suboptimal search methods that can often outperform more traditional search methods in terms of total running time. For example, they are among the few search methods that can find suboptimal solution paths for the 24-puzzle, a domain with more than 7×10^{24} states.

Real-time search methods have usually been investigated in the context of traditional AI search domains: sliding tile puzzles such as the 8- or 24 puzzle, blocks worlds, grid worlds, and others. These domains are usually assumed to be deterministic: whenever an action is executed in the same state, the same successor state results. Many domains, however, are non-deterministic, such as many robotics, control, or scheduling domains. In this paper, we present a first step towards extending real-time search methods to non-deterministic single-agent search domains by viewing real-time search as a game where the search method selects the actions and nature, a fictitious opponent, chooses their outcomes.

We investigate suboptimal search, i.e. how to get the agent to any goal state. The path traversed by the agent does neither have to be optimal nor repeatable, which is a sufficient condition for many real-world problems. Real-time search methods appear to be well suited for suboptimal search in non-deterministic state spaces. In contrast to traditional (off-line) search techniques, which must plan for every possible outcome, real-time search methods only need to choose actions for those outcomes that actually occur. Thus, real-time search methods can potentially decrease search time, although possibly at the expense of action execution time: Since they do not plan exhaustively for every possible outcome of an action, one cannot be sure how good it really is to execute the action. It might well be that the action has an outcome that makes it hard for the agent to reach a goal state. In this paper we begin to quantify the tradeoff between search time and action execution time by analyzing the performance of real-time search in non-deterministic domains.

Our new technique, which we call min-max LRTA*, is based on Korf's Learning Real-Time A* algorithm (LRTA*) [Korf, 1987; 1988; 1990]. LRTA* is a single-agent real-time search algorithm that can be used to find suboptimal and optimal solution paths in deterministic domains. It performs only minimal computations between action executions (we constrain it to a lookahead of one), choosing only which action to execute next, and basing this decision only on information local to its current state. We extend LRTA* to non-deterministic domains, describe which non-deterministic domains it can solve, and analyze its performance for these domains. We also give tight bounds on its worst-case performance and show how this performance depends on properties of the domains and the heuristic functions. Our theoretical analysis, which suggests what constitutes easy and hard real-time search problems in non-deterministic domains, also applies to both deterministic domains and multi-agent domains (such as moving target search).

*This research was supported in part by NASA under contract NAGW-1175. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the U.S. government.

2 Assumptions and Notation

LRTA*-type real-time search algorithms differ from traditional search algorithms, such as the A* algorithm, in that they always maintain a current state. This is a state of the search space; it can only be changed by executing actions. The real-time search algorithm can choose the action freely from the actions that are applicable in its current state. While chronological backtracking is such a search method, it can only be used in undirected, deterministic state spaces. In non-deterministic state spaces, a real-time search algorithm might not be able to backtrack (i.e. undo action executions).

We view real-time search in non-deterministic domains as a two-player game. The action that the real-time search algorithm selects determines the possible successor states, from which some mechanism, which we call **nature**, has to choose one. We do not impose any restrictions on how nature makes its decisions (its **strategy**) and, furthermore, assume that we do not know nature's strategy. Although a second agent might indeed exist in some real-time search scenarios, our assumption of its existence is simply an analysis tool.

We use the following notation: S denotes the finite set of states of the state space (of size $n := |S|$), $s_{start} \in S$ is the start state, and G (with $\emptyset \neq G \subseteq S$) is the set of goal states. $A(s)$ is the finite set of actions that can be executed in state $s \in S$. Executing action $a \in A(s)$ causes a (potentially non-deterministic) state transition into one of the states $succ(s,a)$ (with $\emptyset \neq succ(s,a) \subseteq S$). **Identity actions** are actions $a \in A(s)$ with $s \in succ(s,a)$, i.e. those actions that might not result in a state change. We call a state space **deterministic** iff the cardinality of $succ(s,a)$ is one for all $s \in S$ and $a \in A(s)$. For deterministic state spaces, we use $succ(s,a)$ not only to denote the set of successor states, but also the only element of this set. Note that every deterministic state space is per definition non-deterministic as well. We call a state space **non-deterministic** if we want to stress that we do not require it to be deterministic.

The **distance** $d(s,s') \in [0,\infty]$ between $s \in S$ and $s' \in S$ is measured in action executions and defined to be the (unique) solution of the following set of equations (for all $s, s' \in S$)

$$d(s,s') = \begin{cases} 0 & \text{if } s = s' \\ 1 + \min_{a \in A(s)} \max_{s'' \in succ(s,a)} d(s'',s') & \text{otherwise} \end{cases}$$

In other words: the search algorithm can reach s' from s with at most $d(s,s')$ action executions (regardless of nature's strategy) given that the search algorithm knows the state space and acts optimally. The **goal distance** $gd(s)$ of $s \in S$ is defined to be $gd(s) := \min_{s' \in G} d(s,s')$. If $gd(s) = \infty$ then there exists a strategy for nature that prevents the search algorithm from reaching any goal state from state s (although nature might choose not to follow this strategy). Note that $gd(s) \leq n-1$ if $gd(s)$ is finite (otherwise nature could cause the search algorithm to loop indefinitely). We define the **diameter** (depth) of the state space with respect to G as $d := \max_{s \in S} gd(s)$. For deterministic state spaces, the definitions of $d(s,s')$ and $gd(s)$ simplify to the standard definitions of distance and goal distance, respectively.

Initially, $V(s) = f(s)$ for all $s \in S$, where f is a heuristic function for the goal distance. The search algorithm starts in state s_{start}.

1. $s :=$ the current state.
2. If $s \in G$, then stop successfully.
3. $a := \text{argmin}_{a \in A(s)} V(succ(s,a))$.
4. Set $V(s) := \max(V(s), 1 + V(succ(s,a)))$.
5. Execute action a. (As a consequence, the new current state becomes $succ(s,a)$.)
6. Go to 1.

Figure 1: Deterministic domains: LRTA*

Initially, $V(s) = f(s)$ for all $s \in S$, where f is a heuristic function for the goal distance. The search algorithm starts in state s_{start}.

1. $s :=$ the current state.
2. If $s \in G$, then stop successfully.
3. $a := \text{argmin}_{a \in A(s)} \max_{s' \in succ(s,a)} V(s')$.
4. Set $V(s) := \max(V(s), 1 + \max_{s' \in succ(s,a)} V(s'))$.
5. Execute action a. (As a consequence, nature selects the new current state from $succ(s,a)$ according to its strategy.)
6. Go to 1.

Figure 2: Non-deterministic domains: min-max LRTA*

---

3 Deterministic Domains: LRTA*

We describe a simple version of Korf's **LRTA* algorithm** that has lookahead one. It consists of a termination checking step (line 2), an action selection step (line 3), a value update step (line 4), and an action execution step (line 5), see Figure 1.

First, LRTA* checks whether it has reached a goal state and thus can terminate successfully. If not, it decides on the action to execute next. It looks only one action execution ahead, picking the action that leads to the successor state with the smallest state value $V(s)$, which approximates the goal distance $gd(s)$ (ties can be broken arbitrarily). Note that the algorithm is greedy since it always chooses the action that appears to be best locally. The algorithm then replaces the value $V(s)$ with the one-step lookahead value $\max(V(s), 1+V(succ(s,a)))$, which is a more accurate estimate. Finally, LRTA* executes the selected action and iterates.

Korf showed that LRTA* is correct for deterministic, strongly connected (i.e. $d(s,s') < \infty$ for all $s, s' \in S$) state spaces. That is, it reaches a goal state eventually and terminates; the sequence of executed actions is a suboptimal solution path.

4 Non-Deterministic Domains: Min-Max LRTA*

The extensions necessary to make LRTA* work in non-deterministic state spaces are fairly straightforward. Since we do not know which strategy nature uses, we use

a (worst-case) minimax approach and let the search algorithm act as if nature tries to maximize the goal distance of the search algorithm while the search algorithm tries to minimize it. If the search algorithm can reach a goal state and terminate for such a vicious strategy of nature, it will also reach a goal state if nature uses a different, and therefore less vicious, strategy. As a consequence, the search algorithm does not depend on assumptions about the strategy that nature actually uses.

The **min-max LRTA* algorithm** is shown in Figure 2. It uses $\max_{s' \in succ(s,a)} V(s')$ at the places in the action selection step (line 3) and value update step (line 4) where LRTA* uses $V(succ(s,a))$. In deterministic state spaces, min-max LRTA* reduces to the original LRTA* algorithm.

5 Performance Analysis

In this section, we analyze the performance of min-max LRTA*, which we measure as the total number of action executions until a goal state is reached. This is justified, because the time needed to execute an action in the world often dominates the minimal amount of computation that min-max LRTA* performs between action executions. Even if this is not the case, the total number of actions that min-max LRTA* executes can still be roughly proportional to its total running time, because it performs only a bounded and in many domains essentially constant amount of computation between action executions. We define its **complexity** to be an upper bound on the number of action executions that holds for all possible topologies of state spaces of a given size, start and goal states, tie breaking rules among actions that evaluate to the same value, and strategies of nature.

There exist state spaces in which every real-time search algorithm has infinite complexity. This is the case if the search algorithm can get trapped in a part of the state space that does not contain a goal state. Traditionally, researchers have therefore restricted their attention to strongly connected state spaces or, more generally, state spaces with $d < \infty$. We use the same assumption for non-deterministic state spaces and call state spaces with this property **safely explorable**. (To be more precise: The goal distances of all states *that the agent can reach from its start state without passing through a goal state* have to be finite.) Moore and Atkeson's partigame algorithm [Moore and Atkeson, 1993], for example, learns non-deterministic abstractions of spatial state spaces that are safely explorable.

Intuitively, we expect min-max LRTA* to do well in safely explorable state spaces when the state spaces are relatively small or contain many goal states. In the latter case, the we expect it to perform the better, the more the goal states are spread out over the state space. In the following, we analyze this intuition formally.

5.1 Complexity: Upper Bounds

In this section, we provide upper bounds on the complexity of min-max LRTA*. But first, we introduce some definitions of properties of the state values $V(s)$ that we need in order to be able to state our results.

Definition 1 *The state values of min-max LRTA* are* **consistent** *iff, for all* $s \in G$, $V(s) = 0$, *and, for all* $s \in S \backslash G$, $0 \leq V(s) \leq 1 + \min_{a \in A(s)} \max_{s' \in succ(s,a)} V(s')$.

Definition 2 *The state values of min-max LRTA* are* **admissible** *iff, for all* $s \in S$, $0 \leq V(s) \leq gd(s)$.

In deterministic state spaces, these definitions reduce to the standard definitions as used for traditional heuristic search algorithms, such as the A* algorithm [Pearl, 1985]. In particular, consistency (or, equivalently, monotonicity) means that the triangle inequality holds and admissibility means that the state values of min-max LRTA* never overestimate the goal distances. Note that zero-initialized values are consistent, and consistent values are admissible.

We can prove the following theorem. (For all of the following proofs, we provide outlines only.)

Theorem 1 *Initial state values that are consistent (or admissible) remain consistent (or admissible) after every action execution of min-max LRTA* and are monotonically non-decreasing.*

Proof sketch: The theorem can easily be proven by induction. (For admissible initial state values in deterministic state spaces, the theorem follows from [Ishida and Korf, 1991].) ■

This theorem enables us to simplify the value update step of min-max LRTA* if its initial state values are consistent. According to the theorem, the values remain consistent. This means according to the definition of consistent state values that $V(s) \leq 1 + \min_{a \in A(s)} \max_{s' \in succ(s,a)} V(s')$ for $s \notin G$. It follows that $\max(V(s), 1 + \max_{s' \in succ(s,a)} V(s')) = 1 + \max_{s' \in succ(s,a)} V(s')$ for $a := \operatorname{argmin}_{a \in A(s)} \max_{s' \in succ(s,a)} V(s')$ and $s \notin G$, and the value update step can be simplified to $V(s) := 1 + \max_{s' \in succ(s,a)} V(s')$ without changing the values assigned to $V(s)$.

How does min-max LRTA* work? Assume consistent state values and call $PG := \{s \in S : V(s) = 0\} \supseteq G$ the set of potential goal states. For example, if min-max LRTA* is zero-initialized, then PG is always the set of states which the search algorithm has not yet visited at least once. We can easily prove by induction that (for all $s \in S$)

$$0 \leq V(s) \leq \min_{s' \in PG} d(s, s')$$

The action selection step of min-max LRTA* can be interpreted as using $V(s)$ to approximate $\min_{s' \in PG} d(s, s')$. This means that it tries (sometimes unsuccessfully) to direct the search algorithm from its current state to the closest potential goal state with as few action executions as possible. Thus, it always executes the action that appears to be best according to its local view of the state space.

How efficient is min-max LRTA*? A time superscript of t in the following complexity result refers to the values of the variables immediately before min-max LRTA* executes the $(t+1)$st action, e.g. $s^{(t=)0} = s_{start}$ and $V^{(t=)0}(s) = f(s)$. Also, the theorem refers to identity

actions. These are actions whose execution might not result in a state change. $S_{id} := \{s \in S : \exists_{a \in A(s)} s \in succ(s,a)\} \subseteq S$ is the set of states in which identity actions can be executed.

Theorem 2 *For all $t = 0, 1, 2, \ldots$ (until termination) and a min-max LRTA* algorithm with an admissible heuristic function f it holds that*

$$t \leq \sum_{s \in S} \left[V^t(s) - V^0(s)\right] - (V^t(s^t) - V^0(s^0)) + loop^t \qquad (1)$$

and

$$loop^t \leq \sum_{s \in S_{id}} \left[V^t(s) - V^0(s)\right], \qquad (2)$$

where $loop^t := |\{t' \in \{0, \ldots, t-1\} : s^{t'} = s^{t'+1}\}|$ (the number of identity actions executed before t that did not change the state).

Proof sketch (by induction): For $t = 0$, the inequalities reduce to $t \leq 0$ and $loop^t \leq 0$, which is true. Now assume that the inequalities hold at time t and consider how the variables change from time t to $t+1$. The only state value that changes is $V(s^t)$. It increases according to Theorem 1 by $V^{t+1}(s^t) - V^t(s^t) \geq 0$, and so does $\sum_{s \in S} \left[V(s) - V^0(s)\right]$. We distinguish two cases:

1. $s^t \neq s^{t+1}$, i.e. the LHS of Inequality (1), t, increases by one and the LHS of Inequality (2), *loop*, does not change. Since $V^{t+1}(s^t) - V^t(s^t) \geq 1 + \max_{s' \in succ(s^t,a^t)} V^t(s') - V^t(s^t) \geq 1 + V^t(s^{t+1}) - V^t(s^t) = 1 + V^{t+1}(s^{t+1}) - V^t(s^t)$, the RHS of Inequality (1) increases by at least one. The RHS of Inequality (2) does not decrease. Thus, the two inequalities continue to hold at time $t+1$.
2. $s^t = s^{t+1}$, i.e. both the LHS of Inequality (1), t, and the LHS of Inequality (2), *loop*, increase by one. Since $V^{t+1}(s^t) = V^{t+1}(s^{t+1})$ and *loop* increases by one, the RHS of Inequality (1) increases by one as well. Since $s^t \in S_{id}$ and $V^{t+1}(s^t) - V^t(s^t) = 1 + \max_{s' \in succ(s^t,a^t)} V^t(s') - V^t(s^t) \geq 1 + V^t(s^t) - V^t(s^t) = 1$, the RHS of Inequality (2) increases by at least one. Thus, the two inequalities continue to hold at time $t+1$. ■

Theorem 3 *A min-max LRTA* algorithm with an admissible heuristic function reaches a goal state and terminates after at most $2\sum_{s \in S} gd(s)$ action executions (regardless of nature's strategy). If the state space has no identity actions, a min-max LRTA* algorithm reaches a goal state and terminates after at most $\sum_{s \in S} gd(s)$ action executions (regardless of nature's strategy).*

Proof sketch:

$$\begin{aligned}
t &\overset{\text{Th. 2}}{\leq} \sum_{s \in S} \left[V^t(s) - V^0(s)\right] - (V^t(s^t) - V^0(s^0)) + loop^t \\
&\overset{\text{Th. 2}}{\leq} 2 \sum_{s \in S} \left[V^t(s) - V^0(s)\right] - (V^t(s^t) - V^0(s^0)) \\
&= 2 \sum_{s \in S \setminus \{s^0\}} \left[V^t(s) - V^0(s)\right] + 2V^t(s^0) - V^0(s^0) - V^t(s^t) \\
&\overset{\text{Adm.}}{\leq} 2 \sum_{s \in S \setminus \{s^0\}} [gd(s) - 0] + 2\,gd(s^0) - 0 - 0 \\
&= 2 \sum_{s \in S} gd(s)
\end{aligned}$$

If the state space has no identity actions, then $loop^t = 0$ for all t and the second part of the theorem follows similarly. ■

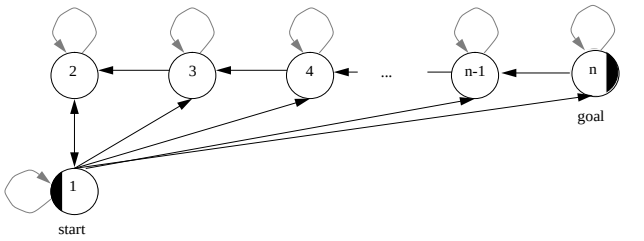

Figure 3: A worst-case example ($n \geq 1$)

As a consequence, min-max LRTA* is guaranteed to reach a goal state eventually (i.e. it is correct) if the state space is safely explorable ($d < \infty$). In this case, the algorithm reaches a goal state after at most $O(nd)$ action executions, because $2\sum_{s \in S} gd(s) \leq 2nd$. $O(nd) \leq O(n^2)$, since $d \leq n-1$ if $d < \infty$. Note that the goal distances of *all* states influence the complexity of min-max LRTA*, not only the goal distance of the start state.

5.2 Complexity: Lower Bounds

In this section, we prove, by example, that the upper bounds from the last section are tight for uninformed (i.e. zero-initialized) min-max LRTA* algorithms. For each example, rather than explaining what occurs, we provide pseudo-code that prints the sequence of states that the algorithm could traverse. (The scope of the for-statements is shown by indentation.) It is easy to determine from inspection of the code how many actions the algorithm executes.

All of our example state spaces are deterministic. This shows that the bounds remain tight for this important subclass of non-deterministic state spaces: deterministic state spaces are not easier to solve with min-max LRTA* than non-deterministic ones. Since min-max LRTA* reduces to the original LRTA* algorithm in deterministic domains, the bounds are tight for the original LRTA* algorithm as well, see also [Koenig, 1992].

Theorem 3 states that a zero-initialized min-max LRTA* algorithm reaches a goal state after at most $2\sum_{s \in S} gd(s)$ action executions. $2\sum_{s \in S} gd(s) \leq 2\sum_{i=0}^{n-1} i = n^2 - n$ for safely explorable state spaces. Now consider the state space in Figure 3 with the identity actions included. The following program shows one possible sequence that a zero-initialized min-max LRTA* algorithm can traverse (in this case, ties are broken by remaining in one's current state if possible and otherwise always choosing the state with the lowest label):

```
for i := 1 to n-1
```

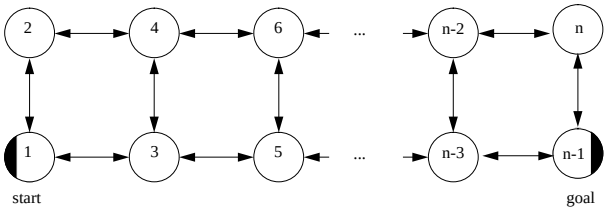

Figure 4: A rectangular grid world

```
    for j := 1 to i
      print i
    for j := i downto 1
      print j
  print n
```

In this case, the min-max LRTA* algorithm must execute a total of $n^2 - n$ actions. Thus, the complexity of $n^2 - n$ is tight.

Theorem 3 also states that a zero-initialized min-max LRTA* algorithm reaches a goal state after at most $\sum_{s \in S} gd(s)$ action executions if the state space has no identity actions. $\sum_{s \in S} gd(s) \leq 1/2n^2 - 1/2n$ for safely explorable state spaces. Now consider again the state space in Figure 3, this time with the identity actions removed. A zero-initialized min-max LRTA* algorithm can traverse the following state sequence (which is equal to the above state sequence, but with repeated occurrences of the same state deleted):

```
for i := 1 to n-1
  for j := i downto 1
    print j
print n
```

Since the min-max LRTA* algorithm executes $1/2n^2 - 1/2n$ actions in this case, the complexity of $1/2n^2 - 1/2n$ is tight for state spaces that have no identity actions. Note that identity actions can always be safely deleted from a state space, since their removal does not affect whether min-max LRTA* can solve a given search problem in the worst-case. Our results show, however, that their removal can at most halve the complexity of uninformed (i.e. zero-initialized) min-max LRTA*.

The state space used in the above examples was artificially constructed. However, the complexity of $O(n^2)$ is tight even for more realistic state spaces, such as grid worlds. They have often been used as testbeds for real-time search methods [Pemberton and Korf, 1992; Ishida and Korf, 1991]. Consider the grid world shown in Figure 4 and assume $n \geq 2$ with $n \bmod 4 = 2$. A zero-initialized min-max LRTA* algorithm can traverse the following state sequence:

```
for i := n-3 downto n/2 step 2
  for j := 1 to i step 2
    print j
  for j := i+1 downto 2 step 2
    print j
for i := 1 to n-1 step 2
  print i
```

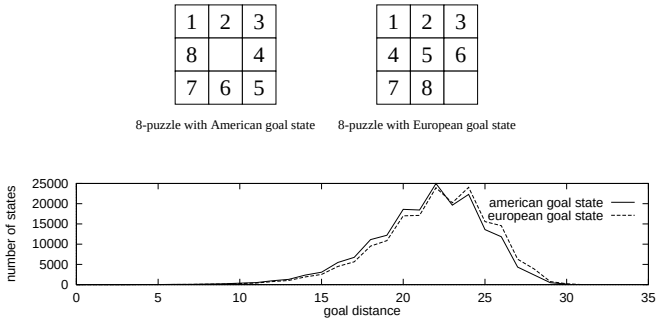

Figure 5: Goal distances of the 8-puzzle

In this case, the min-max LRTA* algorithm executes $3n^2/16 - 3/4$ actions before it reaches the goal state. This also shows that the complexity of $O(n^2)$ is tight for undirected state spaces for which the number of actions that can be executed in any state is bounded from above by a small constant (here: three).

5.3 Decreasing the Complexity

This section demonstrates how properties of the search domains and the heuristic functions can decrease the complexity of min-max LRTA*.

Domain Properties

Since the complexity of uninformed min-max LRTA* is tight at $2\sum_{s \in S} gd(s)$, our intuition was correct: the smaller the state space and/or the average goal distance, the smaller the complexity. Consider, for instance, the sliding tile puzzles. These deterministic domains are sometimes considered to be hard search problems, because they have a low goal density. The 8-puzzle, for example, has 181440 states (that are reachable from the start state), but only one goal state. Our complexity results, however, imply that average goal distance, not goal density, is among the factors that determine the hardness of a search problem for (min-max or original) LRTA*. Although increasing the goal density tends to decrease the average goal distance, there are search problems with low goal density *and* low average goal distance. The 8-puzzle is an example: Figure 5 shows for every goal distance how many of the 181440 states have this particular goal distance. It turns out that the average goal distance of the 8-puzzle with the American goal state is only 21.5 and the largest goal distance is 30. Similarly, the average goal distance of the 8-puzzle with the European goal state (that cannot be reached from the American goal state) is 22.0 and the largest goal distance is 31. (See [Reinefeld, 1993] for extensive statistics on the 8-puzzle.) In both cases, the average goal distances are much smaller than the size of the state space. Thus, sliding-tile puzzles are rather well-suited search problems for (min-max or original) LRTA* – compared to many grid worlds of the same size, for example. This does not imply, however, that LRTA* can solve sliding tile puzzles with a huge number of tiles, since the complexity of LRTA* does not only depend on the average

goal distance, but also on the number of states, which grows as a factorial in the number of tiles.

Properties of the Heuristic Functions
Prior knowledge in form of more informed, admissible heuristic functions decreases the complexity of min-max LRTA*. For example, in the totally informed case one initializes $V^0(s) := f(s)$ with $f(s) = gd(s)$ for all $s \in S$ and Theorem 2 predicts

$$\begin{aligned} t &\overset{\text{Th. 2}}{\leq} \sum_{s \in S} \left[V^t(s) - gd(s)\right] - (V^t(s^t) - gd(s^0)) + loop^t \\ &\overset{\text{Th. 2}}{\leq} 2 \sum_{s \in S} \left[V^t(s) - gd(s)\right] - (V^t(s^t) - gd(s^0)) \\ &\overset{\text{Adm.}}{\leq} 2 \sum_{s \in S} \left[gd(s) - gd(s)\right] - (0 - gd(s^0)) \\ &\leq gd(s^0) \end{aligned}$$

That is, Theorem 2 predicts correctly that the search algorithm needs only at most $gd(s)$ action executions to reach a goal state from any given $s \in S$ and, thus, that it follows a shortest path to a goal.

It is easy to determine admissible heuristic functions for non-deterministic state spaces if one can determine them for deterministic state spaces. One can simply assume that nature decides in advance which successor state $g(s,a) \in succ(s,a)$ to choose every time the agent executes action $a \in A(s)$ in state $s \in S$ – all possible assumptions about which particular actions nature chooses are fine. If nature really used this strategy and the agent found out about it, then the state space would effectively become deterministic for the agent. One can easily see that any admissible heuristic function for the goal distances in this deterministic state space is admissible for the original, non-deterministic search problem as well, regardless of the strategy that nature actually uses. Note, however, that the informedness of this heuristic function depends on how close the assumed behavior of nature is to its most vicious strategy.

6 Example Domains

In this section, we give two examples that demonstrate how min-max LRTA* can be applied to non-deterministic search problems. In particular, we discuss search problems with coarse models (for example, abstract state spaces) and moving target search.

6.1 Search with Coarse Models

Our complexity results for min-max LRTA* do not depend on how nature selects successor states. Thus, they apply to scenarios where the search algorithm is not able to make assumptions about nature's strategy. Assume, for example, that one can model a deterministic world only with low granularity. Then, one might not be able to identify one's current state uniquely, and actions can appear to have non-deterministic effects. Assume, for instance, that a search algorithm occupies either state 1 or state 2 in some state space, but cannot distinguish between these two states. Action a is a deterministic action that results in state 3 when it is executed in state 1 and in state 4 when it is executed in state 2. Thus, the execution of action a can result in either state 3 or 4, but the search algorithm has no way of predicting which of these states will result and could attribute this to nature having a strategy that is unknown to the search algorithm.

An application with these characteristics is the **tray-tilting problem** [Christiansen, 1992; Kadie, 1991; Erdman and Mason, 1988]: One puts an object into a tray in a given starting position and then slides it repeatedly by tilting the tray until it is in a given goal position. In our version of the tray-tilting problem, one can observe the position of the object with an overhead camera before deciding on a tilting action. The corresponding state space is non-deterministic, because one can neither observe the position of the object precisely nor control the motion of the tray precisely. Min-max LRTA* can be used to control the tilting actions directly and eventually orients the object in the desired position if the state space is safely explorable. (We have performed experiments to verify that a large number of tray tilting problems are indeed safely explorable.) Our complexity results provide an upper bound on the number of tray tilting actions needed.

6.2 Moving Target Search

When deriving the complexity results, we assumed the existence of a fictitious opponent "nature." But the results also apply to scenarios where there is a real opponent. In a way, applying single-agent real-time search methods to two-player games closes a loop, since real-time search was originally inspired by time-constraints present in antagonistic two-agent domains such as game playing. Consider, for example, **moving target search** – the task for a hunter is to catch an independently acting prey. Both agents move on a known directed graph. The hunter moves first, then they alternate moves to adjacent vertices. (If the agents can pass their moves, one can model this by adding identity actions to the graph.) Both agents can always sense the current vertices of themselves and the other agent, but the hunter does not know where the prey will move. The hunter catches the prey if both agents occupy the same vertex.

In our framework, the agent is the hunter and nature is the prey. It is straightforward to map the moving target search problem to a non-deterministic single agent search problem against nature (Figure 6). The hunter can catch the prey for sure if the derived state space is safely explorable. If the hunter uses min-max LRTA* with lookahead one, then we can utilize our complexity result to derive an upper bound on the number of actions that the hunter executes before it catches the prey.

[Ishida and Korf, 1991] have also applied real-time search methods to moving target search, but utilize LRTA* for the hunter in a different way. Their MTS algorithm learns the following strategy for the hunter (until it catches the prey): always move to an adjacent vertex that is on a shortest path to the current vertex of the prey. They prove that the hunter eventually catches the prey on a strongly connected graph if it is faster than the prey. Note the differences between the two ap-

Figure 6: A simple moving target search problem

proaches: Obviously, one has to make some assumptions to ensure that the prey can not force the hunter into a cycle in which the hunter cannot decrease its distance to the prey. Ishida and Korf do not restrict the topology of the graph, but have to assume that the hunter has a speed advantage over the prey. Consider for example the graph in Figure 6 (note that one of its edges is directed) and assume that both agents are equally fast. Korf and Ishida's algorithm always follows the prey at the same distance if the algorithm is totally informed and the prey runs around in an anti-clockwise cycle. Min-max LRTA*, however, will eventually go left until the prey takes the one-way street and later go right until the prey is caught, no matter which strategy the prey uses.

7 Extensions

In this paper, we have measured the complexity of min-max LRTA* in action executions. Therefore, every action has a cost of one associated with it. However, our analysis can easily be generalized to arbitrary strictly positive cost structures, including ones with non-uniform costs (in a way analogous to [Koenig and Simmons, 1992] where we assume deterministic state spaces). Furthermore, various methods have been proposed that improve the performance of LRTA*, see for example [Matsubara and Ishida, 1994; Ishida, 1993; Knight, 1993; Hamidzadeh, 1992; Ishida, 1992; Russell and Wefald, 1991; Shekhar and Dutta, 1989] – we have applied and analyzed these methods in the context of min-max LRTA*.

If one could make assumptions about nature's strategy (for example, if one knew that nature is a neutral coin flipper) or the state space were not safely explorable, one would use a more sophisticated search strategy than a minimax approach. Consequently, our future publications will report real-time search results for these cases.

8 Related Work

[Korf, 1988] considered deterministic, strongly connected domains and showed that LRTA* reaches a goal state eventually. He also showed that LRTA* eventually finds a shortest path from the start state to a goal state if it is repeatedly reset into the start state when it reaches a goal state. [Ishida, 1993] performed additional systematic experiments to understand how the performance of LRTA* can be improved by utilizing initial knowledge in form of heuristic functions. [Barto *et al.*, 1995] showed how LRTA* can be generalized to finding paths of minimal average lengths in probabilistic domains. [Heger, 1994] used an on-line minimax algorithm based on Q-learning [Watkins, 1989] to learn paths of minimal worst-case length. Since his algorithm, Q-learning, is similar to min-max LRTA*, it benefits from our complexity analysis. (For the relationship between Q-learning and LRTA* see [Koenig and Simmons, 1993].) Neither of the above researchers have analyzed the complexity of their algorithms, but most of them report empirical results.

[Ishida and Korf, 1991] proposed the moving target search algorithm MTS, showed how to utilize initial knowledge in form of heuristic functions for MTS, and analyzed its complexity in deterministic, strongly connected state spaces that do not contain identity actions. MTS reduces to LRTA* with lookahead one if the target does not move. [Littman, 1994] pointed out that in antagonistic two-agent situations such as moving target search it can be advantageous to use probabilistic strategies over minimax strategies if both agents can move simultaneously.

9 Conclusion

In this paper, we have relaxed the standard assumption that search domains are deterministic and studied suboptimal real-time search methods in non-deterministic domains. We viewed real-time search as a game where the search algorithm selects the actions and nature, a fictitious opponent, chooses their outcomes. In particular, we introduced the min-max LRTA* algorithm, a simple extension of Korf's LRTA* algorithm to non-deterministic domains.

We analyzed the worst-case performance of min-max LRTA* theoretically. In particular, we introduced the notion of a safely explorable state space and showed that the complexity of uninformed min-max LRTA* in safely explorable state spaces is proportional to the product of the size of the state space $|S|$ and the average goal distance over *all* states. We proved that the complexity of uninformed min-max LRTA* can get as large as $|S|^2-|S|$ action executions, but not larger $(1/2|S|^2 - 1/2|S|$ action executions if the state space does not have identity actions that can leave the state unchanged). We also showed how min-max LRTA* can take advantage of initial knowledge in form of heuristic functions for the goal distances.

Our complexity results hold for deterministic domains as well. In particular, deterministic state spaces are not easier to solve with min-max LRTA* than non-deterministic ones. Since min-max LRTA* reduces to

LRTA* in deterministic domains, the complexity results also apply to the original LRTA* algorithm. It follows, for example, that uninformed LRTA* can search large state spaces with small average goal distances (such as sliding tile puzzles) much faster than equally large state spaces with large average goal distances.

Acknowledgements

Thanks to Alan Christiansen, Matt Mason, Andrew Moore, Sebastian Thrun, and especially Lonnie Chrisman and Matthias Heger for helpful discussions. Thanks also to Toru Ishida for providing an English translation of one of his papers that was written in Japanese.

References

[Barto *et al.*, 1995] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singh. Learning to act using real-time dynamic programming. *Artificial Intelligence*, pages 81–138, 1 1995.

[Christiansen, 1992] Alan D. Christiansen. *Automatic Acquisition of Task Theories for Robotic Manipulation.* PhD thesis, School of Computer Science, Carnegie Mellon University, 1992.

[Erdman and Mason, 1988] M.A. Erdman and M.T. Mason. An exploration of sensorless manipulation. *IEEE Journal of Robotics and Automation*, 4(4):369–379, 8 1988.

[Hamidzadeh, 1992] B. Hamidzadeh. Can real-time search algorithms meet deadlines? In *Proceedings of the AAAI*, pages 486–491, 1992.

[Heger, 1994] Matthias Heger. Consideration of risk in reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 105–111, 1994.

[Ishida and Korf, 1991] Toru Ishida and Richard E. Korf. Moving target search. In *Proceedings of the IJCAI*, pages 204–210, 1991.

[Ishida, 1992] Toru Ishida. Moving target search with intelligence. In *Proceedings of the AAAI*, pages 525–532, 1992.

[Ishida, 1993] Toru Ishida. Real-time bidirectional search. Technical report, NTT Communication Science Laboratory, Japan, 1993.

[Kadie, 1991] C.M. Kadie. Continuous conceptual set covering: Learning robot operators from example. In *Proceedings of the Eighth International Workshop on Machine Learning*, pages 615–619, 1991.

[Knight, 1993] Kevin Knight. Are many reactive agents better than a few deliberative ones? In *Proceedings of the IJCAI*, pages 432–437, 1993.

[Koenig and Simmons, 1992] Sven Koenig and Reid G. Simmons. Complexity analysis of real-time reinforcement learning applied to finding shortest paths in deterministic domains. Technical Report CMU–CS–93–106, School of Computer Science, Carnegie Mellon University, 1992.

[Koenig and Simmons, 1993] Sven Koenig and Reid G. Simmons. Complexity analysis of real-time reinforcement learning. In *Proceedings of the AAAI*, pages 99–105, 1993.

[Koenig, 1992] Sven Koenig. The complexity of real-time search. Technical Report CMU–CS–92–145, School of Computer Science, Carnegie Mellon University, 1992.

[Korf, 1987] Richard E. Korf. Real-time heuristic search: First results. In *Proceedings of the AAAI*, pages 133–138, 1987.

[Korf, 1988] Richard E. Korf. Real-time heuristic search: New results. In *Proceedings of the AAAI*, pages 139–144, 1988.

[Korf, 1990] Richard E. Korf. Real-time heuristic search. *Artificial Intelligence*, 42(2-3):189–211, 3 1990.

[Korf, 1993] Richard E. Korf. Linear-space best-first search. *Artificial Intelligence*, 62(1):41–78, 1 1993.

[Littman, 1994] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, pages 157–163, 1994.

[Matsubara and Ishida, 1994] Shigeo Matsubara and Toru Ishida. Real-time planning by interleaving real-time search with subgoaling. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, pages 122–127, 1994.

[Moore and Atkeson, 1993] Andrew W. Moore and Christopher G. Atkeson. The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In *Proceedings of the NIPS*, 1993.

[Pearl, 1985] Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley, Menlo Park, California, 1985.

[Pemberton and Korf, 1992] Joseph C. Pemberton and Richard E. Korf. Incremental path planning on graphs with cycles. In *Proceedings of the First Annual AI Planning Systems Conference*, pages 179–188, 1992.

[Reinefeld, 1993] Alexander Reinefeld. Complete solution of the eight-puzzle and the benefit of node ordering in IDA*. In *Proceedings of the IJCAI*, pages 248–253, 1993.

[Russell and Wefald, 1991] Stuart Russell and Eric Wefald. *Do the Right Thing – Studies in Limited Rationality.* The MIT Press, Cambridge, Massachusetts, 1991.

[Shekhar and Dutta, 1989] S. Shekhar and S. Dutta. Minimizing response times in real-time planning and search. In *Proceedings of the IJCAI*, pages 238–242, 1989.

[Watkins, 1989] Christopher J. Watkins. *Learning from Delayed Rewards.* PhD thesis, King's College, Cambridge University, 1989.