Real-Time Adaptive A*

Sven Koenig
Computer Science Department
University of Southern California
Los Angeles, CA 90089-0781
skoenig@usc.edu

Maxim Likhachev
Computer Science Department
Carnegie Mellon University
Pittsburgh, PA 15213-3891
maxim+@cs.cmu.edu

ABSTRACT

Characters in real-time computer games need to move smoothly and thus need to search in real time. In this paper, we describe a simple but powerful way of speeding up repeated A* searches with the same goal states, namely by updating the heuristics between A* searches. We then use this technique to develop a novel real-time heuristic search method, called Real-Time Adaptive A*, which is able to choose its local search spaces in a fine-grained way. It updates the values of all states in its local search spaces and can do so very quickly. Our experimental results for characters in real-time computer games that need to move to given goal coordinates in unknown terrain demonstrate that this property allows Real-Time Adaptive A* to follow trajectories of smaller cost for given time limits per search episode than a recently proposed real-time heuristic search method [5] that is more difficult to implement.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Problem Solving, Control Methods, and Search—*Graph and Tree Search Strategies*

General Terms

Algorithms

Keywords

A*; Agent Planning; D* Lite; Games; Heuristic Search; Incremental Search; Perception, Action and Planning in Agents; Planning with the Freespace Assumption; Real-Time Decision Making

1. INTRODUCTION

Agents need to use different search methods than the off-line search methods often studied in artificial intelligence. Characters in real-time computer games, for example, need to move smoothly and thus need to search in real time. Real-time heuristic search methods find only the beginning of a trajectory from the current state of an agent to a goal state [9, 4]. They restrict the search to a small part of the state space that can be reached from the current state with a small number of action executions (local search space). The agent determines the local search space, searches it, decides how to move within it, and executes one or more actions along the resulting trajectory. The agent then repeats this process until it reaches a goal state. Real-time heuristic search methods thus do not plan all the way to a goal state which often results in smaller total search times but larger trajectory costs. Most importantly, real-time heuristic search methods can satisfy hard real-time requirements in large state spaces since the sizes of their local search spaces (= their lookaheads) are independent of the sizes of the state spaces and can thus remain small. To focus the search and prevent cycling, they associate heuristics with the states and update them between searches, which accounts for a large chunk of the search time per search episode. In this paper, we describe Real-Time Adaptive A*, a novel real-time heuristic search method. It is a contract anytime method [12] that is able to choose its local search spaces in a very fine-grained way, updates the heuristics of all states in the local search space and is able to do so very quickly. Our experimental results for goal-directed navigation tasks in unknown terrain demonstrate that Real-Time Adaptive A* follows trajectories of smaller cost for given time limits per search episode than a recently proposed real-time heuristic search method [5] because it updates the heuristics more quickly, which allows it to use larger local search spaces and overcompensate for its slightly less informed heuristics. At the same time, Real-Time Adaptive A* is easier to implement.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

2. MAIN IDEA

Our main idea is simple but powerful. Assume that one has to perform several A* searches with consistent heuristics in the same state space and with the same goal states but possibly different start states. Adaptive A* makes the heuristics more informed after each A* search in order to speed up future A* searches. We now explain the main idea behind Adaptive A*.

A* [3] is an algorithm for finding cost-minimal paths in state spaces (graphs). For every state s, the user supplies a heuristic $h[s]$ that estimates the goal distance of the state (= the cost of a cost-minimal path from the state to a goal state). The heuristics need to be consistent [10]. For every state s encountered during the search, A* maintains two values: the smallest cost $g[s]$ of any discovered path from the start state s_{curr} to state s (which is initially infinity), and an estimate $f[s] := g[s] + h[s]$ of the distance from the start state s_{curr} via state s to a goal state. A* then operates as follows: It maintains a priority queue, called open list, which initially contains only the start state s_{curr}. A* removes a state s with a smallest f-value from the priority queue. If state s is a goal state, it terminates. Otherwise, it expands the state, meaning that it updates the g-value of each successor state of the state and then inserts those successor states into the open list whose g-value decreased. It then repeats

the process. After termination, the g-value of every expanded state s is equal to the distance from the start state s_{curr} to state s.

We now explain how one can make the heuristics more informed after each A* search in order to speed up future A* searches. Assume that s is a state that was expanded during such an A* search. We can obtain an admissible (= non-overestimating) estimate of its goal distance $gd[s]$ as follows: The distance from the start state s_{curr} to any goal state via state s is equal to the distance from the start state s_{curr} to state s plus the goal distance $gd[s]$ of state s. It clearly can not be smaller than the goal distance $gd[s_{\text{curr}}]$ of the start state s_{curr}. Thus, the goal distance $gd[s]$ of state s is no smaller than the goal distance $gd[s_{\text{curr}}]$ of the start state s_{curr} (= the f-value $f[\bar{s}]$ of the goal state $\bar{s}$ that was about to be expanded when the A* search terminates) minus the distance from the start state s_{curr} to state s (= the g-value $g[s]$ of state s when the A* search terminates).

$$\begin{aligned} g[s] + gd[s] &\geq gd[s_{\text{curr}}] \\ gd[s] &\geq gd[s_{\text{curr}}] - g[s] \\ gd[s] &\geq f[\bar{s}] - g[s] \end{aligned}$$

Consequently, $f[\bar{s}] - g[s]$ provides an admissible estimate of the goal distance $gd[s]$ of state s and can be calculated quickly. More informed heuristics can be obtained by calculating and assigning this difference to every state that was expanded during the A* search and thus is in the closed list when the A* search terminates. (The states in the open list are not updated since the distance from the start state to these states can be smaller than their g-values when the A* search terminates.) We evaluated this idea experimentally in [7]. We now use it to develop a novel real-time heuristic search method, called Real-Time Adaptive A* (RTAA*), which reduces to the case discussed above if its lookahead is infinity.

3. NOTATION

We use the following notation to describe search tasks: S denotes the finite set of states. $s_{\text{curr}} \in S$ denotes the start state of the search, and *GOAL* $\subset S$ denotes the set of goal states. $A(s)$ denotes the finite set of actions that can be executed in state $s \in S$. $c[s,a] > \epsilon$ (for a given constant $\epsilon > 0$) denotes the cost of executing action $a \in A(s)$ in state $s \in S$, whereas *succ*$(s,a) \in S$ denotes the resulting successor state. The action costs c can increase during the search but we require the goal distances of all states to remain bounded from above by some constant.

4. REAL-TIME ADAPTIVE A*

Figure 1 shows the pseudo code of RTAA*. The legend explains the constants, variables, and functions that we will refer to in the following. The variables annotated with [USER] have to be initialized before RTAA* is called. s_{curr} needs to be set to the start state of the agent, c to the initial action costs, and h to the initial heuristics, which need to be consistent for the initial action costs, that is, need to satisfy $h[s] = 0$ for all goal states s and $h[s] \leq h[\textit{succ}(s,a)] + c[s,a]$ for all non-goal states s and actions a that can be executed in them [10]. Variables annotated with [A*] are updated during the call to astar() {03} (= line 3 in the pseudo code), which performs a (forward) A* search guided by the current heuristics from the current state of the agent toward the goal states until a goal state is about to be expanded or *lookahead* > 0 states have been expanded. After the A* search, we require $\bar{s}$ to be the state that was about to be expanded when the A* search terminated. We denote this state with $\bar{s}$ consistently throughout this paper. We require that $\bar{s} =$ *FAILURE* if the A* search terminated due to an empty open list, in which case there is no finite-cost trajectory from the current state to any goal state and RTAA* thus returns failure {05}. We require *CLOSED* to contain the states expanded during the A* search and the g-value $g[s]$ to be defined for all generated states s, including all expanded states. We define the f-values $f[s] := g[s] + h[s]$ for these states s. The expanded states s form the local search space, and RTAA* updates their heuristics by setting $h[s] := f[\bar{s}] - g[s] = g[\bar{s}] + h[\bar{s}] - g[s]$ {06-07}. (The heuristics of the other states are not changed.) RTAA* then executes actions along the trajectory found by the A* search until state $\bar{s}$ is reached (or, equivalently, a state is reached that was not expanded or, also equivalently, the local search space is left), *movements* > 0 actions have been executed, or the cost of an action on the trajectory increases {09-16}. It then repeats the process until it reaches a goal state, in which case it returns success {17}.

constants and functions

S	set of states of the search task, a set of states
GOAL	set of goal states, a set of states
$A(\,)$	sets of actions, a set of actions for every state
succ$(\,)$	successor function, a state for every state-action pair

variables

lookahead	number of states to expand at most, an integer larger than zero
movements	number of actions to execute at most, an integer larger than zero
s_{curr}	current state of the agent, a state [USER]
c	current action costs, a float for every state-action pair [USER]
h	current (consistent) heuristics, a float for every state [USER]
g	g-values, a float for every state [A*]
CLOSED	closed list of A* (= all expanded states), a set of states [A*]
$\bar{s}$	state that A* was about to expand when it terminated, a state [A*]

```
procedure realtime_adaptive_astar():
{01} while (s_curr ∉ GOAL) do
{02}   lookahead := any desired integer greater than zero;
{03}   astar();
{04}   if s̄ = FAILURE then
{05}     return FAILURE;
{06}   for all s ∈ CLOSED do
{07}     h[s] := g[s̄] + h[s̄] − g[s];
{08}   movements := any desired integer greater than zero;
{09}   while (s_curr ≠ s̄ AND movements > 0) do
{10}     a := the action in A(s_curr) on the cost-minimal trajectory from s_curr to s̄;
{11}     s_curr := succ(s_curr, a);
{12}     movements := movements − 1;
{13}     for any desired number of times (including zero) do
{14}       increase any desired c[s, a] where s ∈ S and a ∈ A(s);
{15}     if any increased c[s, a] is on the cost-minimal trajectory from s_curr to s̄ then
{16}       break;
{17} return SUCCESS;
```

Figure 1: Real-Time Adaptive A*

The values of *lookahead* and *movements* determine the behavior of RTAA*. For example, RTAA* performs a single A* search from the start state to a goal state and then moves the agent along the trajectory found by the A* search to the goal state if it always chooses infinity for *lookahead* and *movements* and no action costs increase.

We now prove several important properties of RTAA* that hold no matter how it chooses its values of *lookahead* and *movements*. We make use of the following known properties of A* searches with consistent heuristics: First, they expand every state at most once. Second, the g-values of every expanded state and state $\bar{s}$ are equal to the distance from the start state to state s and state $\bar{s}$, respectively. Thus, one knows cost-minimal trajectory from the start state to all expanded states and state $\bar{s}$. Third, the f-values of the series of expanded states over time are monotonically nondecreasing. Thus, $f[s] \leq f[\bar{s}]$ for all expanded states s and $f[\bar{s}] \leq f[s]$ for all generated states s that remained unexpanded.

THEOREM 1. *The heuristics of the same state are monotonically nondecreasing over time and thus indeed become more informed over time.*

Proof: Assume that the heuristic of state s is updated on line

{07}. Then, state s was expanded and it thus holds that $f[s] \leq f[\bar{s}]$. Consequently, $h[s] = f[s] - g[s] \leq f[\bar{s}] - g[s] = g[\bar{s}] + h[\bar{s}] - g[s]$ and the update cannot decrease the heuristic of state s since it changes the heuristic from $h[s]$ to $g[\bar{s}] + h[\bar{s}] - g[s]$. ■

THEOREM 2. *The heuristics remain consistent.*

Proof: We prove this property by induction on the number of A* searches. The initial heuristics are provided by the user and consistent. It thus holds that $h[s] = 0$ for all goal states s. This continues to hold since goal states are not expanded and their heuristics thus not updated. (Even if RTAA* updated the heuristic of state $\bar{s}$, it would leave the h-value of that state unchanged since $f[\bar{s}] - g[\bar{s}] = g[\bar{s}] + h[\bar{s}] - g[\bar{s}] = h[\bar{s}]$. Thus, the heuristics of goal states would remain zero even in that case.) It also holds that $h[s] \leq h[succ(s, a)] + c[s, a]$ for all non-goal states s and actions a that can be executed in them. Assume that some action costs increase on lines {13-14}. Let c denote the action costs before all increases and c' denote the action costs after all increases. Then, $h[s] \leq h[succ(s, a)] + c[s, a] \leq h[succ(s, a)] + c'[s, a]$ and the heuristics thus remain consistent. Now assume that the heuristics are updated on lines {06-07}. Let h denote the heuristics before all updates and h' denote the heuristics after all updates. We distinguish three cases:

- First, both s and $succ(s, a)$ were expanded, which implies that $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s]$ and $h'[succ(s, a)] = g[\bar{s}] + h[\bar{s}] - g[succ(s, a)]$. Also, $g[succ(s, a)] \leq g[s] + c[s, a]$ since the A* search discovers a trajectory from the current state via state s to state $succ(s, a)$ of cost $g[s]+c[s, a]$ during the expansion of state s. Thus, $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s] \leq g[\bar{s}]+h[\bar{s}]-g[succ(s, a)]+c[s, a] = h'[succ(s, a)]+c[s, a]$.
- Second, s was expanded but $succ(s, a)$ was not, which implies that $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s]$ and $h'[succ(s, a)] = h[succ(s, a)]$. Also, $g[succ(s, a)] \leq g[s] + c[s, a]$ for the same reason as in the first case, and $f[\bar{s}] \leq f[succ(s, a)]$ since state $succ(s, a)$ was generated but not expanded. Thus, $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s] = f[\bar{s}] - g[s] \leq f[succ(s, a)] - g[s] = g[succ(s, a)]+h[succ(s, a)]-g[s] = g[succ(s, a)]+h'[succ(s, a)] - g[s] \leq g[succ(s, a)] + h'[succ(s, a)] - g[succ(s, a)] + c[s, a] = h'[succ(s, a)] + c[s, a]$.
- Third, s was not expanded, which implies that $h'[s] = h[s]$. Also, $h[succ(s, a)] \leq h'[succ(s, a)]$ since the heuristics of the same state are monotonically nondecreasing over time. Thus, $h'[s] = h[s] \leq h[succ(s, a)] + c[s, a] \leq h'[succ(s, a)] + c[s, a]$.

Thus, $h'[s] \leq h'[succ(s, a)] + c[s, a]$ in all three cases and the heuristics thus remain consistent. ■

THEOREM 3. *The agent reaches a goal state.*

Proof: Assume that the heuristics are updated on lines {06-07}. Let h denote the heuristics of RTAA* before all updates and h' denote the heuristics after all updates. The heuristics of the same state are monotonically nondecreasing over time, according to Theorem 1. Assume that the agent moves from its current state s to some state s' (with $s \neq s'$) along a cost-minimal trajectory from state s to state $\bar{s}$. It holds that $h'[s] = f[\bar{s}] - g[s] = f[\bar{s}]$ since state s is the start of the search and its g-value is thus zero and since it was expanded and its heuristic was thus updated. Furthermore, it holds that $h'[s'] = f[\bar{s}] - g[s']$ since either state s' was expanded and its heuristic was thus updated or $s' = \bar{s}$ and then $h'[s'] = h'[\bar{s}] = h[\bar{s}] = f[\bar{s}] - g[\bar{s}] = f[\bar{s}] - g[s']$. Thus, after the agent moved from state s to state s' and its current state thus changed from state s to state s', the heuristic of the current state decreased by $h'[s] - h'[s'] = f[\bar{s}] - (f[\bar{s}] - g[s']) = g[s']$ and the sum of the heuristics of all states but its current state thus increased by $g[s']$, which is bounded from below by a positive constant since $s \neq s'$ and we assumed that all action costs are bounded from below by a positive constant. Thus, the sum of the heuristics of all states but the current state of the agent increases over time beyond any bound if the agent does not reach a goal state. At the same time, the heuristics remain consistent, according to Theorem 2 (since consistent heuristics are admissible), and are thus no larger than the goal distances which we assumed to be bounded from above, which is a contradiction. Thus, the agent is guaranteed to reach a goal state. ■

THEOREM 4. *If the agent is reset into the start state whenever it reaches a goal state then the number of times that it does not follow a cost-minimal trajectory from the start state to a goal state is bounded from above by a constant if the cost increases are bounded from below by a positive constant.*

Proof: Assume for now that the cost increases leave the goal distances of all states unchanged. Under this assumption, it is easy to see that the agent follows a cost-minimal trajectory from the start state to a goal state if it follows a trajectory from the start state to a goal state where the h-value of every state is equal to its goal distance. If the agent does not follow such a trajectory, then it transitions at least once from a state s whose h-value is not equal to its goal distance to a state s' whose h-value is equal to its goal distance since it reaches a goal state according to Theorem 3 and the h-value of the goal state is zero since the heuristics remain consistent according to Theorem 2. We now prove that the h-value of state s is then set to its goal distance. When the agent executes some action $a \in A(s)$ in state s and transitions to state s', then state s is a parent of state s' in the A* search tree produced during the last call of astar() and it thus holds that (1) state s was expanded during the last call of astar(), (2) either state s' was also expanded during the last call of astar() or $s' = \bar{s}$, (3) $g[s'] = g[s] + c[s, a]$. Let h denote the heuristics before all updates and h' denote the heuristics after all updates. Then, $h'[s] = f[\bar{s}] - g[s]$ and $h'[s'] = f[\bar{s}] - g[s'] = gd[s']$. The last equality holds because we assumed that the h-value of state s' was equal to its goal distance and thus can no longer change since it could only increase according to Theorem 1 but would then make the heuristics inadmissible and thus inconsistent, which is impossible according to Theorem 2. Consequently, $h'[s] = f[\bar{s}] - g[s] = gd[s'] + g[s'] - g[s] = gd[s'] + c[s, a] \geq gd[s]$, proving that $h'[s] = gd[s]$ since a larger h-value would make the heuristics inadmissible and thus inconsistent, which is impossible according to Theorem 2. Thus, the h-value of state s is indeed set to its goal distance. After the h-value of state s is set to its goal distance, the h-value can no longer change since it could only increase according to Theorem 1 but would then make the heuristics inadmissible and thus inconsistent, which is impossible according to Theorem 2. Since the number of states is finite, it can happen only a bounded number of times that the h-value of a state is set to its goal distance. Thus, the number of times that the agent does not follow a cost-minimal trajectory from the start state to a goal state is bounded. The theorem then follows since the number of times that a cost increase does not leave the goal distances of all states unchanged is bounded since we assumed that the cost increases are

bounded from below by a positive constant but the goal distances are bounded from above. After each such change, the number of times that the agent does not follow a cost-minimal trajectory from the start state to a goal state is bounded. ■

5. RELATIONSHIP TO LRTA*

RTAA* is similar to a version of Learning Real-Time A* recently proposed in [5], an extension of the original Learning Real-Time A* algorithm [9] to larger lookaheads. For simplicity, we refer to this particular version of Learning Real-Time A* as LRTA*. RTAA* and LRTA* differ only in how they update the heuristics after an A* search. LRTA* replaces the heuristic of each expanded state with the sum of the distance from the state to a generated but unexpanded state s and the heuristic of state s, minimized over all generated but unexpanded states s. (The heuristics of the other states are not changed.) Let $\bar{h}'$ denote the heuristics after all updates. Then, the heuristics of LRTA* after the updates satisfy the following system of equations for all expanded states s:

$$\bar{h}'[s] = \min_{a \in A(s)} (c[s,a] + \bar{h}'[succ(s,a)])$$

The properties of LRTA* are similar to the ones of RTAA*. For example, its heuristics for the same state are monotonically nondecreasing over time and remain consistent, and the agent reaches a goal state. We now prove that LRTA* and RTAA* behave exactly the same if their lookahead is one and they break ties in the same way. They can behave differently for larger lookaheads, and we give an informal argument why the heuristics of LRTA* tend to be more informed than the ones of RTAA* with the same lookaheads. On the other hand, it takes LRTA* more time to update the heuristics and it is more difficult to implement, for the following reason: LRTA* performs one search to determine the local search space and a second search to determine how to update the heuristics of the states in the local search space since it is unable to use the results of the first search for this purpose, as explained in [5]. Thus, there is a trade-off between the total search time and the cost of the resulting trajectory, and we need to compare both search methods experimentally to understand this trade-off better.

THEOREM 5. *RTAA* with lookahead one behaves exactly like LRTA* with the same lookahead if they break ties in the same way.*

Proof: We show the property by induction on the number of A* searches. The heuristics of both search methods are initialized with the heuristics provided by the user and are thus identical before the first A* search. Now consider any A* search. The A* searches of both search methods are identical if they break ties in the same way. Let $\bar{s}$ be the state that was about to be expanded when their A* searches terminated. Let h denote the heuristics of RTAA* before all updates and h' denote the heuristics after all updates. Similarly, let $\bar{h}$ denote the heuristics of LRTA* before all updates and $\bar{h}'$ denote the heuristics after all updates. Assume that $h[s] = \bar{h}[s]$ for all states s. We show that $h'[s] = \bar{h}'[s]$ for all states s. Both search methods expand only the current state s of the agent and thus update only the heuristic of this one state. Since $s \neq \bar{s}$, it holds that $h'[s] = g[\bar{s}] + h[\bar{s}] - g[s] = g[\bar{s}] + h[\bar{s}]$ and $\bar{h}'[s] = \min_{a \in A(s)}(c[s,a] + \bar{h}'[succ(s,a)]) = \min_{a \in A(s)}(g[succ(s,a)] + \bar{h}'[succ(s,a)]) = \min_{a \in A(s)}(g[succ(s,a)] + \bar{h}[succ(s,a)]) = g[\bar{s}] + \bar{h}[\bar{s}] = g[\bar{s}] + h[\bar{s}]$. Thus, both search methods set the heuristic of the current state to the same value and then move to state $\bar{s}$. Notice that *lookahead* $= 1$ implies without loss of generality that *movements* $= 1$. Consequently, they behave exactly the same. ■

We now give an informal argument why the heuristics of LRTA* with lookaheads larger than one tend to be more informed than the ones of RTAA* with the same lookahead (if both real-time heuristic search methods use the same value of *movements*). This is not a proof but gives some insight into the behavior of the two search methods. Assume that both search methods are in the same state and break ties in the same way. Let h denote the heuristics of RTAA* before all updates and h' denote the heuristics after all updates. Similarly, let $\bar{h}$ denote the heuristics of LRTA* before all updates and $\bar{h}'$ denote the heuristics after all updates. Assume that $h[s] = \bar{h}[s]$ for all states s. We now prove that $h'[s] \leq \bar{h}'[s]$ for all states s. The A* searches of both search methods are identical if they break ties in the same way. Thus, they expand the same states and thus also update the heuristics of the same states. We now show that the heuristics h' cannot be consistent if $h'[s] > \bar{h}'[s]$ for at least one state s. Assume that $h'[s] > \bar{h}'[s]$ for at least one state s. Pick a state s with the smallest $\bar{h}'[s]$ for which $h'[s] > \bar{h}'[s]$ and pick an action a with $a = \arg\min_{a \in A(s)}(c[s,a] + \bar{h}'[succ(s,a)]$. State s must have been expanded since $h[s] = \bar{h}[s]$ but $h'[s] > \bar{h}'[s]$. Then, it holds that $\bar{h}'[s] = c[s,a] + \bar{h}'[succ(s,a)]$. Since $\bar{h}'[s] = c[s,a] + \bar{h}'[succ(s,a)] > \bar{h}'[succ(s,a)]$ and state s is a state with the smallest $\bar{h}'[s]$ for which $h'[s] > \bar{h}'[s]$, it must be the case that $h'[succ(s,a)] \leq \bar{h}'[succ(s,a)]$. Put together, it holds that $h'[s] > \bar{h}'[s] = c[s,a] + \bar{h}'[succ(s,a)] \geq c[s,a] + h'[succ(s,a)]$. This means that the heuristics h' are inconsistent but we have earlier proved already that they remain consistent, which is a contradiction. Consequently, it holds that $h'[s] \leq \bar{h}'[s]$ for all states s. Notice that this proof does not imply that the heuristics of LRTA* always dominate the ones of RTAA* since the search methods can move the agent to different states and then update the heuristics of different states, but it suggests that the heuristics of LRTA* with lookaheads larger than one tend to be more informed than the ones of RTAA* with the same lookaheads and thus that the trajectories of LRTA* tend to be of smaller cost than the trajectories of RTAA* with the same lookaheads (if both real-time heuristic search methods use the same value of *movements*).

In the remainder of the paper, we assume that the agents always choose the same constant for *lookahead*, which is an external parameter, and always use infinity for *movements*, both for LRTA* and RTAA*.

6. APPLICATION

Real-time heuristic search methods are often used as alternative to traditional search methods for solving offline search tasks [9]. We, however, apply RTAA* to goal-directed navigation in unknown terrain, a search task that requires agents to execute actions in real time. Characters in real-time computer games, for example, often do not know the terrain in advance but automatically observe it within a certain range around them and then remember it for future use. Figure 2 shows an example. To make these agents easy to control, the users can click on some position in known or unknown terrain and the agents then move autonomously to this position. If the agents observe during execution that their current trajectory is blocked, then they need to search for another plan. The searches need to be fast since the agents need to move smoothly even if the processor is slow, the other game components use most of the available processor cycles and there are a large number of agents that all have to search repeatedly. Thus, there is a time limit per A* search, which suggests that real-time heuristic search methods are a good fit for our navigation tasks. To apply them, we discretize the terrain into cells that are either blocked or unblocked, a common practice in the context of real-time computer games [1]. The agents initially do not know which cells are blocked but use a navigation strat-

Figure 2: Total Annihilation

egy from robotics [8]: They assume that cells are unblocked unless they have the cells already observed to be blocked (freespace assumption). They always know which (unblocked) cells they are in, observe the blockage status of their four neighboring cells, raise the action costs of actions that enter the newly observed blocked cells, if any, from one to infinity, and then move to any one of the unblocked neighboring cells with cost one. We therefore use the Manhattan distances as consistent heuristic estimates of the goal distances. The task of the agents is to move to the given goal cell, which we assume to be possible.

7. ILLUSTRATION

Figure 3 shows a simple goal-directed navigation task in unknown terrain that we use to illustrate the behavior of RTAA*. Black squares are blocked. All cells have their initial heuristic in the lower left corner. We first compare RTAA* with lookahead infinity to LRTA* with the same lookahead and forward A* searches. All search methods start a new search episode (= run another search) when the cost of an action on their current trajectory increases and break ties between cells with the same f-values in favor of cells with larger g-values and remaining ties in the following order, from highest to lowest priority: right, down, left and up. We do this since we believe that systematic rather than random tie-breaking helps the readers to understand the behavior of the different search methods better since all search methods then follow the same trajectories. Figures 4, 5, and 6 show the agent as a small black circle. The arrows show the planned trajectories from the current cell of the agent to the goal cell, which is in the lower right corner. Cells that the agent has already observed to be blocked are black. All other cells have their heuristic in the lower left corner. Generated cells have their g-value in the upper left corner and their f-value in the upper right corner. Expanded cells are grey and, for RTAA* and LRTA*, have their updated heuristics in the lower right corner, which makes it easy for the readers to compare them to the heuristics before the update in the lower left corner. Notice that forward A* searches, RTAA* with lookahead infinity and LRTA* with the same lookahead follow the same trajectories if they break ties in the same way. They differ only in the number of cell expansions, which is larger for forward A* searches (23) than for RTAA* (20) and larger for RTAA* (20) than for LRTA* (19). The first property is due to RTAA* and LRTA* updating the heuristics while forward A* searches do not. Thus, forward A*

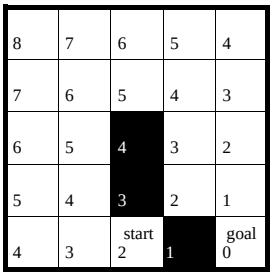

Figure 3: Example

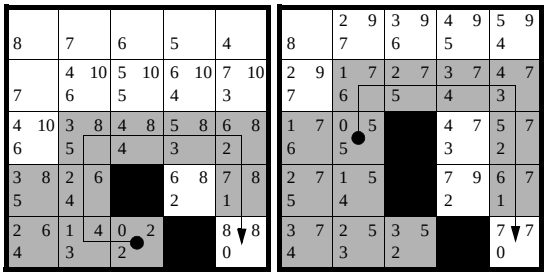

Figure 4: Forward A* Searches

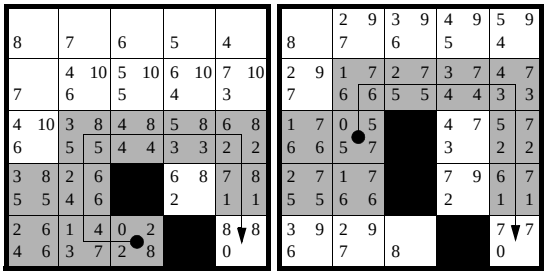

Figure 5: RTAA* with *lookahead* $= \infty$

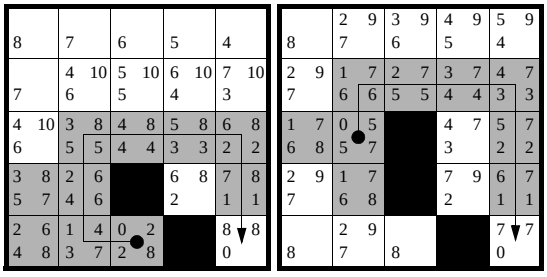

Figure 6: LRTA* with *lookahead* $= \infty$

searches fall prey to the local minimum in the heuristic value surface and thus expand the three leftmost cells in the lowest row a second time, while RTAA* and LRTA* avoid these cell expansions. The second property is due to some updated heuristics of LRTA* being larger than the ones of RTAA*. Notice, however, that most updated heuristics are identical, although this is not guaranteed in general. We also compare RTAA* with lookahead four to RTAA* with lookahead infinity. Figures 5 and 7 show that decreasing the lookahead of RTAA* increases the trajectory cost (from 10 to 12) but decreases the number of cell expansions (from 20 to 17) because smaller lookaheads imply that less information is used dur-

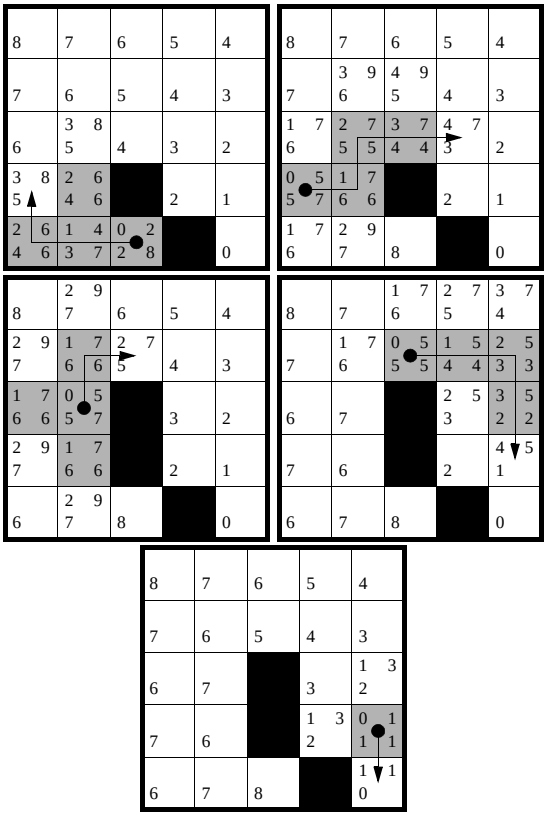

Figure 7: RTAA* with *lookahead* $= 4$

ing each search episode. (Notice that the last search episode of RTAA* with lookahead four expands only one cell since the goal cell is about to be expanded next.) We now perform systematic experiments to see whether they confirm the trends shown in our examples.

8. EXPERIMENTAL RESULTS

We now run experiments to compare RTAA* to forward A* searches and LRTA*, as in the previous section. All three search methods search forward. We also compare RTAA* to two search methods that search backward, namely backward A* searches and D* Lite [6] (an incremental version of backward A* searches that is similar to but simpler than D* [11]), which is possible because there is only one goal cell.[1] All search methods use binary heaps as priority queues and now break ties between cells with the same f-values in favor of cells with larger g-values (which is known to be a good tie-breaking strategy) and remaining ties randomly.

[1]Notice that the total number of cell expansions and total search time of forward A* searches are significantly smaller than the ones of backward A* searches. The big impact of the search direction can be explained as follows: The agent observes blocked cells close to itself. Thus, the blocked cells are close to the start of the search in the early phases of forward A* searches, but close to the goal of the search in the early phases of backward A* searches. The closer the blocked cells are to the start of the search, the more cells there are for which the Manhattan distances are perfectly informed and thus the faster A* searches are that break ties between cells with the same f-values in favor of cells with larger g-values since they expand only states along a cost-minimal trajectory from cells for which the Manhattan distances are perfectly informed to the goal of the search.

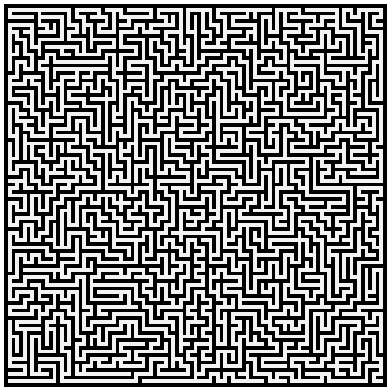

Figure 8: Test Terrain: Mazes

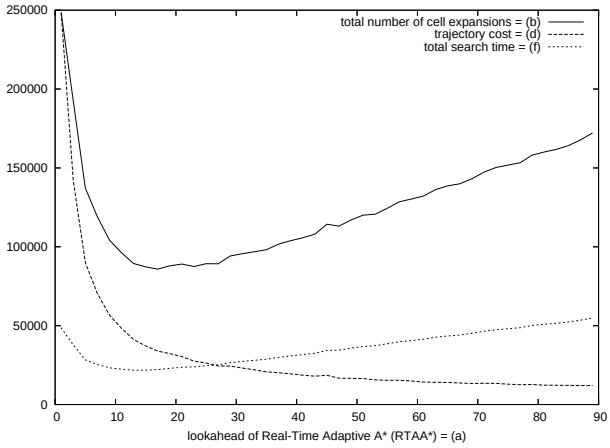

Figure 9: Performance of Real-Time Adaptive A*

We perform experiments on a SUN PC with a 2.4 GHz AMD Opteron Processor 150 in randomly generated four-connected mazes of size 151×151 that are solvable. Their corridor structure is generated with depth-first search. The start and goal cells are chosen randomly. Figure 8 shows an example (of smaller size than used in the experiments). We use the Manhattan distances as heuristics. The performance measures in Table 1 are averaged over the same 2500 grids. We show the standard deviation of the mean for three key performance measures in square brackets to demonstrate the statistical significance of our results, namely the total number of cell expansions and the total search time on one hand and the resulting trajectory cost on the other hand. We first verify the properties suggested in the previous sections:

- RTAA* with lookahead infinity, LRTA* with the same lookahead, D* Lite and forward and backward A* searches follow the same trajectories if they break ties in the same way and their trajectory costs are indeed approximately equal. (The slight differences are due to remaining ties being broken randomly.) The total number of cell expansions and the total search time are indeed larger for forward A* searches than RTAA*, and larger for RTAA* than LRTA*, as suggested in "Illustration."

- Decreasing the lookahead of RTAA* indeed increases the trajectory cost but initially decreases the total number of cell expansions and the total search time, as suggested in "Il-

Table 1: Experiments in Mazes

(a) = lookahead (= bound on the number of cell expansions per search episode), (b) = total number of cell expansions (until the agent reaches the goal cell) [in square brackets: standard deviation of the mean], (c) = total number of search episodes (until the agent reaches the goal cell), (d) = trajectory cost (= total number of action executions until the agent reaches the goal cell) [in square brackets: standard deviation of the mean], (e) = number of action executions per search episode, (f) = total search time (until the agent reaches the goal cell) in microseconds [in square brackets: standard deviation of the mean], (g) = search time per search episode in microseconds, (h) = search time per action execution in microseconds, (i) = increase of heuristics per search episode and expanded cell (= per update) = $h'[s] - h[s]$ averaged over all search episodes and expanded cells s

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)	(i)
	Real-Time Adaptive A* (RTAA*)							
1	248537.79 [5454.11]	248537.79	248537.79 [5454.11]	1.00	48692.17 [947.08]	0.20	0.20	1.00
9	104228.97 [2196.69]	11583.50	56707.84 [1248.70]	4.90	23290.36 [360.77]	2.01	0.41	2.40
17	85866.45 [1701.83]	5061.92	33852.77 [774.35]	6.69	22100.20 [301.99]	4.37	0.65	3.10
25	89257.66 [1593.02]	3594.51	26338.21 [590.00]	7.33	24662.64 [310.42]	6.86	0.94	3.30
33	96839.84 [1675.46]	2975.65	22021.81 [521.77]	7.40	27994.45 [350.79]	9.41	1.27	3.28
41	105702.99 [1699.20]	2639.59	18628.66 [435.06]	7.06	31647.50 [382.30]	11.99	1.70	3.08
49	117035.65 [1806.59]	2473.55	16638.27 [390.09]	6.73	35757.69 [428.17]	14.46	2.15	2.90
57	128560.04 [1939.38]	2365.94	15366.63 [361.95]	6.49	39809.02 [477.72]	16.83	2.59	2.79
65	138640.02 [2019.98]	2270.38	14003.74 [314.38]	6.17	43472.99 [517.36]	19.15	3.10	2.63
73	150254.51 [2176.68]	2224.29	13399.01 [309.72]	6.02	47410.44 [567.88]	21.31	3.54	2.57
81	160087.23 [2269.20]	2172.94	12283.65 [270.03]	5.65	50932.60 [608.94]	23.44	4.15	2.39
89	172166.56 [2436.73]	2162.75	12078.40 [261.16]	5.58	54874.88 [663.96]	25.37	4.54	2.37
∞	642823.02 [20021.00]	1815.92	5731.20 [81.72]	3.16	226351.59 [7310.53]	124.65	39.49	4.22
	Learning Real-Time A* (LRTA*)							
1	248537.79 [5454.11]	248537.79	248537.79 [5454.11]	1.00	67252.19 [1354.67]	0.27	0.27	1.00
9	87613.37 [1865.31]	9737.38	47290.61 [1065.07]	4.86	27286.14 [437.09]	2.80	0.58	2.93
17	79312.59 [1540.44]	4676.76	30470.32 [698.08]	6.52	29230.15 [409.96]	6.25	0.96	3.61
25	82850.86 [1495.61]	3338.86	23270.38 [551.75]	6.97	34159.80 [450.49]	10.23	1.47	3.74
33	92907.75 [1548.37]	2858.19	20015.55 [472.86]	7.00	40900.68 [516.47]	14.31	2.04	3.71
41	102787.86 [1619.33]	2570.83	17274.12 [403.65]	6.72	47559.60 [587.96]	18.50	2.75	3.54
49	113139.63 [1716.88]	2396.66	15398.47 [360.45]	6.42	54324.06 [665.02]	22.67	3.53	3.38
57	125013.41 [1829.10]	2307.68	14285.14 [328.39]	6.19	61590.97 [744.33]	26.69	4.31	3.25
65	133863.67 [1956.49]	2201.60	13048.50 [300.69]	5.93	67482.95 [829.44]	30.65	5.17	3.12
73	146549.69 [2080.76]	2181.76	12457.92 [277.60]	5.71	74868.92 [909.31]	34.32	6.01	3.02
81	157475.45 [2209.65]	2150.04	11924.96 [262.61]	5.55	81469.32 [989.84]	37.89	6.83	2.95
89	166040.29 [2355.33]	2102.91	11324.72 [246.94]	5.39	86883.98 [1077.54]	41.32	7.67	2.88
∞	348072.76 [7021.57]	1791.19	5611.09 [80.43]	3.13	203645.42 [3782.37]	113.69	36.29	8.20
	D* Lite							
–	47458.83 [581.03]	1776.24	5637.46 [77.03]	3.17	37291.83 [378.20]	20.99	6.62	–
	Forward A* Search							
–	1857468.48 [68324.90]	1732.07	5354.26 [76.91]	3.09	544065.45 [21565.61]	314.11	101.61	–
	Backward A* Search							
–	5245087.82 [93697.15]	1795.72	5535.05 [77.09]	3.08	1698163.04 [31051.10]	945.67	306.80	–

lustration." Increasing the trajectory cost increases the total number of search episodes. If the lookahead is already small and continues to decrease, then eventually the speed with which the total number of search episodes increases is larger than the speed with which the lookahead and the time per search episode decreases, so that the number of cell expansions and the total search time increase again, as the graphs in Figure 9 show. (The graph also shows that the total number of cell expansions and the total search time are proportional, as expected.)

- RTAA* with lookaheads larger than one and smaller than infinity indeed increases the heuristics less than LRTA* with the same lookaheads per update, as suggested in "Relationship to LRTA*." Consequently, its trajectory costs and total number of cell expansions are larger than the ones of LRTA* with the same lookaheads. However, it updates the heuristics much faster than LRTA* with the same lookaheads, resulting in smaller total search times.

- RTAA* with lookahead one and LRTA* with the same lookahead follow the same trajectories and update the same states if they break ties in the same way and their trajectory costs and total number of cell expansions are indeed approximately equal, as suggested in "Relationship to LRTA*."

One advantage of RTAA* is that its total planning time with a carefully chosen lookahead is smaller than that of all other search methods, although its trajectory costs then are not the smallest ones. This is important for applications where planning is slow but actions can be executed fast. However, the advantage of RTAA* over the other search methods for our particular application is a different one: Remember that there is a time limit per search episode so that the characters move smoothly. If this time limit is larger than 20.99 microseconds, then one should use D* Lite for the search because the resulting trajectory costs are smaller than the ones of all other search methods whose search time per search episode is no larger than 20.99. (The table shows that the trajectory costs of forward A* search are smaller but, as argued before, this difference is due to noise.) However, if the time limit is smaller than 20.99 microseconds, then one has to use a real-time heuristics search method. In this case, one should use RTAA* rather than LRTA*. Assume, for example, that the time limit is 20.00 microseconds. Then one can use either RTAA* with a lookahead of 67, resulting in a trajectory cost of 13657.31, or LRTA* with a lookahead of 43, resulting in a trajectory cost of 16814.49. Thus, the trajectory cost of LRTA* is about 23 percent higher than the one of RTAA*, which means that RTAA* improves the state of the art in real-time heuristic search. A similar argument holds if the search time is amortized over the number of action executions, in which case there is a time limit per action execution. If this time limit is larger than 6.62 microseconds, then one should use D* Lite for the search because the resulting trajectory costs are smaller than the ones of all other search methods whose search time per action execution is no larger than 6.62. However, if the time limit is smaller than 6.62 microseconds, then one has to use a real-time heuristics search method. In this case, one should use RTAA* rather than LRTA*. Assume, for example,

that the time limit is 4.00 microseconds. Then one can use either RTAA* with a lookahead of 79, resulting in a trajectory cost of 12699.99, or LRTA* with a lookahead of 53, resulting in a trajectory cost of 14427.80. Thus, the trajectory cost of LRTA* is about 13 percent higher than the one of RTAA*, which again means that RTAA* improves the state of the art in real-time heuristic search.

9. CONCLUSIONS

In this paper, we developed Real-Time Adaptive A* (RTAA*). This real-time heuristic search method is able to choose its local search spaces in a fine-grained way. It updates the values of all states in its local search spaces and can do so very quickly. Our experimental results for goal-directed navigation tasks in unknown terrain demonstrated that this property allows RTAA* to move to the goal with smaller total search times than a variety of tested alternative search methods. Furthermore, we showed that RTAA* follows trajectories of smaller cost for given time limits per search episode than a recently proposed real-time heuristic search method because it updates the heuristics more quickly, which allows it to use larger local search spaces and overcompensate for its slightly less informed heuristics. It is future work to extend RTAA* to inconsistent heuristics, compare it to real-time heuristic search methods besides LRTA* and combine the idea behind RTAA* with other ideas of enhancing real-time heuristic search methods, for example, the ones described in [2] and [4]. In particular, it would be interesting to study a hierarchical version of RTAA*.

Acknowledgments

This research has been partly supported by an NSF award to Sven Koenig under contract IIS-0350584. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the sponsoring organizations, agencies, companies or the U.S. government.

10. REFERENCES

[1] M. Bjornsson, M. Enzenberger, R. Holte, J. Schaeffer, and P. Yap. Comparison of different abstractions for pathfinding on maps. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1511–1512, 2003.

[2] V. Bulitko and G. Lee. Learning in real-time search: A unifying framework. *Journal of Artificial Intelligence Research*, page (in press), 2005.

[3] P. Hart, N. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 2:100–107, 1968.

[4] T. Ishida. *Real-Time Search for Learning Autonomous Agents*. Kluwer Academic Publishers, 1997.

[5] S. Koenig. A comparison of fast search methods for real-time situated agents. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, pages 864–871, 2004.

[6] S. Koenig and M. Likhachev. D* Lite. In *Proceedings of the National Conference on Artificial Intelligence*, pages 476–483, 2002.

[7] S. Koenig and M. Likhachev. Adaptive A* [poster abstract]. In *Proceedings of the International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1311–1312, 2005.

[8] S. Koenig, C. Tovey, and Y. Smirnov. Performance bounds for planning in unknown terrain. *Artificial Intelligence*, 147:253–279, 2003.

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

[10] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1985.

[11] A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 1652–1659, 1995.

[12] S. Zilberstein. *Operational Rationality through Compilation of Anytime Algorithms*. PhD thesis, Computer Science Department, University of California at Berkeley, Berkeley (California), 1993.