Speeding up the Convergence of Real-Time Search

David Furcy and **Sven Koenig**
Georgia Institute of Technology
College of Computing
Atlanta, GA 30332-0280
{dfurcy,skoenig}@cc.gatech.edu

Abstract

Learning Real-Time A* (LRTA*) is a real-time search method that makes decisions fast and still converges to a shortest path when it solves the same planning task repeatedly. In this paper, we propose new methods to speed up its convergence. We show that LRTA* often converges significantly faster when it breaks ties towards successors with smallest f-values (*a la* A*) and even faster when it moves to successors with smallest f-values instead of only breaking ties in favor of them. FALCONS, our novel real-time search method, uses a sophisticated implementation of this successor-selection rule and thus selects successors very differently from LRTA*, which always minimizes the estimated cost to go. We first prove that FALCONS terminates and converges to a shortest path, and then present experiments in which FALCONS finds a shortest path up to sixty percent faster than LRTA* in terms of action executions and up to seventy percent faster in terms of trials. This paper opens up new avenues of research for the design of novel successor-selection rules that speed up the convergence of both real-time search methods and reinforcement-learning methods.

Introduction

Real-time (heuristic) search methods interleave planning (via local searches) and plan execution, and allow for fine-grained control over how much planning to perform between plan executions. They have successfully been applied to a variety of planning problems, including traditional search problems (Korf 1990), moving-target search problems (Ishida & Korf 1991), STRIPS-type planning problems (Bonet, Loerincs, & Geffner 1997), robot navigation and localization problems with initial pose uncertainty (Koenig & Simmons 1998), robot exploration problems (Koenig 1999), totally observable Markov decision process problems (Barto, Bradtke, & Singh 1995), and partially observable Markov decision process problems (Geffner & Bonet 1998). Learning-Real Time A* (LRTA*) is probably the most popular real-time search method (Korf 1990). It converges to a shortest path when it solves the same planning task repeatedly. Unlike traditional search methods, such as A* (Nilsson 1971), it can not only act in real time (which is important, for example, for real-time control) but also amortize learning over several planning episodes. This allows it to find a suboptimal path fast and then improve the path until it follows a shortest path. Thus, the sum of planning and plan-execution time is always small, yet LRTA* follows a shortest path in the long run.

Recently, researchers have attempted to speed up the convergence of LRTA* while maintaining its advantages over traditional search methods, that is, without increasing its lookahead. Ishida, for example, achieved a significant speedup by sacrificing the optimality of the resulting path (Ishida & Shimbo 1996; Ishida 1997). We, on the other hand, show how to achieve a significant speedup without sacrificing the optimality of the resulting path. FALCONS (FAst Learning and CONverging Search), our novel real-time search method, looks similar to LRTA* but selects successors very differently. LRTA* always greedily minimizes the estimated cost to go (in A* terminology: the sum of the cost of moving to a successor and its h-value). FALCONS, on the other hand, always greedily minimizes the estimated cost of a shortest path from the start to a goal via the successor it moves to (in A* terminology: the f-value of the successor). This allows FALCONS to focus the search more sharply on the neighborhood of an optimal path. Our experiments on standard search domains from the artificial intelligence literature show that FALCONS indeed converges typically about twenty percent faster and in some cases even sixty percent faster than LRTA* in terms of travel cost. It also converges typically about forty percent faster and in some cases even seventy percent faster than LRTA* in terms of trials, even though it looks at the same states as LRTA* when it selects successors and even though it is not more knowledge-intensive to implement.

This paper, in addition to its relevance to the real-time search community, also sends an important message to reinforcement-learning researchers. Indeed, they are typically interested in fast convergence to an optimal behavior and use methods that, just like LRTA*, interleave planning (via local searches) and plan execution and converge to optimal behaviors when they solve the same planning task repeatedly (Barto, Bradtke, & Singh 1995). Furthermore, during exploitation, all commonly-used reinforcement-learning methods, again just like LRTA*, always greedily move to minimize the expected estimated cost to go (Thrun 1992). The results of this paper therefore suggest that it might be possible to design reinforcement-learning methods that converge substantially faster to optimal behaviors than state-of-the-art reinforcement-learning methods, by using information to guide exploration and exploitation that is more directly related to the learning objective.

Definitions

Throughout this paper, we use the following notation and definitions. S denotes the finite state space; $s_{start} \in S$ denotes the start state; and $s_{goal} \in S$ denotes the goal state.[1] $succ(s) \subseteq S$ denotes the set of successors of state s, and $pred(s) \subseteq S$ denotes the set of its predecessors. $c(s,s') > 0$ denotes the cost of moving from state s to successor $s' \in succ(s)$. The goal distance $gd(s)$ of state s is the cost of a shortest path from state s to the goal, and the start distance $sd(s)$ of state s is the cost of a shortest path from the start to state s. Each state s has a g-value and an h-value associated with it, two concepts known from A* search (Nilsson 1971). We use the notation $g(s)/h(s)$ to denote these values. The h-value of state s denotes an estimate of its true goal distance $h^*(s) := gd(s)$. Similarly, the g-value of state s denotes an estimate of its true start distance $g^*(s) := sd(s)$. Finally, the f-value of state s denotes an estimate of the cost $f^*(s) := g^*(s) + h^*(s)$ of a shortest path from the start to the goal through state s. H-values are called admissible iff $0 \le h(s) \le gd(s)$ for all states s, that is, if they do not overestimate the goal distances. They are called consistent iff $h(s_{goal}) = 0$ and $0 \le h(s) \le c(s,s') + h(s')$ for all states s with $s \neq s_{goal}$ and $s' \in succ(s)$, that is, if they satisfy the triangle inequality. It is known that zero-initialized h-values are consistent, and that consistent h-values are admissible (Pearl 1985). The definition of admissibility can be extended in a straightforward way to the g- and f-values, and the definition of consistency can be extended to the g-values (Furcy & Koenig 2000).

Assumptions

In this paper, we assume that the given heuristic values are admissible. Almost all commonly-used heuristic values have this property. If $h(s,s')$ denotes $h(s)$ with respect to goal s', then we initialize the g- and h-values as follows: $h(s) := h(s, s_{goal})$ and $g(s) := h(s_{start}, s)$ for all states s. We also assume that the domain is safely explorable, i.e., the goal distances of all states are finite, which guarantees that the task remains solvable by real-time search methods since they cannot accidentally reach a state with infinite goal distance.

Learning Real-Time A*

In this section, we describe Learning Real-Time A* (LRTA*) (Korf 1990), probably the most popular real-time search method. LRTA* (with lookahead one) is shown in Figure 1. Each state s has an h-value associated with it.

[1]We assume that there is only one goal throughout this paper (with the exception of Figure 5) to keep the notation simple. All of our results continue to hold in domains with multiple goals.

1. $s := s_{start}$.
2. $s' := \arg\min_{s'' \in succ(s)}(c(s,s'') + h(s''))$.
   Break ties arbitrarily.
3. $h(s) :=$ if $s = s_{goal}$ then $h(s)^\dagger$
   else $\max(h(s), \min_{s'' \in succ(s)}(c(s,s'') + h(s'')))$.
4. If $s = s_{goal}$, then stop successfully.
5. $s := s'$.
6. Go to 2.

Figure 1: LRTA*

LRTA* first decides which successor to move to (successor-selection rule, Step 2). It looks at the successors of the current state and always greedily minimizes the estimated cost to go, that is, the sum of the cost of moving to a successor and the estimated goal distance of that successor (i.e., its h-value). Then, LRTA* updates the h-value of its current state to better approximate its goal distance (value-update rule, Step 3). Finally, it moves to the selected successor (Step 5) and iterates the procedure (Step 6). LRTA* terminates successfully when it reaches the goal (Step 4). A more comprehensive introduction to LRTA* and other real-time search methods can be found in (Ishida 1997).

The following properties of LRTA* are known: First, its h-values never decrease and remain admissible. Second, LRTA* terminates (Korf 1990). We call a *trial* any execution of LRTA* that begins at the start and ends in the goal. Third, if LRTA* is reset to the start whenever it reaches the goal and maintains its h-values from one trial to the next, then it eventually follows a shortest path from the start to the goal (Korf 1990). We call a *run* any sequence of trials from the first one until convergence is detected. We say that LRTA* breaks ties systematically if it breaks ties for each state according to an arbitrary ordering on its successors that is selected at the beginning of each run. If LRTA* breaks ties systematically, then it must have converged when it did not change any h-value during a trial. We use this property to detect convergence. To represent the state of the art, we use LRTA* that "breaks ties randomly," meaning that ties are broken systematically according to orderings on the successors that are randomized before each run.

Tie-Breaking

LRTA* terminates and eventually follows a shortest path no matter how its successor-selection rule breaks ties among successors. In this section, we demonstrate, for the first time, that the tie-breaking criterion crucially influences the convergence speed of LRTA*. We present an experimental study that shows that LRTA* converges significantly faster to a shortest path when it breaks ties towards successors with smallest f-values rather than, say, randomly or towards successors with largest f-values. Breaking ties towards successors with smallest f-values is inspired by the A* search method, that efficiently finds a shortest path by always ex-

†This test could be eliminated by moving Step 4 before Step 2 so that the h-value of s_{goal} is never modified. However, we prefer the current (equivalent) formulation since it makes the value-update rule for the h-values completely symmetrical with the value-update rule for the g-values to be introduced in FALCONS.

1. $s := s_{start}$.
2. $s' := \arg\min_{s'' \in succ(s)}(c(s, s'') + h(s''))$. Break ties in favor of a successor s'' with a smallest f-value, where $f(s'') := g(s'') + h(s'')$. Break remaining ties arbitrarily (but systematically).
3. $g(s) :=$ if $s = s_{start}$ then $g(s)$ else $\max(g(s), \min_{s'' \in pred(s)}(g(s'') + c(s'', s)))$.
   $h(s) :=$ if $s = s_{goal}$ then $h(s)$ else $\max(h(s), \min_{s'' \in succ(s)}(c(s, s'') + h(s'')))$.
4. If $s = s_{goal}$, then stop successfully.
5. $s := s'$.
6. Go to 2.

Figure 2: TB-LRTA*

panding a leaf node of the search tree with the smallest f-value, where $f(s) := g^*(s) + h(s)$ for all states s (Pearl 1985). If the g- and h-values are perfectly informed (that is, the g-value of each state is equal to its start distance and its h-value is equal to its goal distance), then the states with smallest f-values are exactly those on shortest paths from the start to the goal. Thus, if LRTA* breaks ties towards successors with smallest f-values, it breaks ties towards a shortest path. If the g- and h-values are not perfectly informed (the more common case), then LRTA* breaks ties towards what currently looks like a shortest path and may thus converge faster. To implement this tie-breaking criterion, LRTA* does not have the g*-values available but can approximate them with g-values. It can update the g-values in a way similar to how it updates the h-values, except that it uses the predecessors instead of the successors. Figure 2 shows TB-LRTA* (Tie-Breaking LRTA*), our real-time search method that maintains g- and h-values and breaks ties towards successors with smallest f-values, where $f(s) := g(s) + h(s)$ for all states s. Remaining ties can be broken arbitrarily (but systematically). We compared TB-LRTA* against versions of LRTA* that break ties randomly or towards successors with largest f-values. We performed experiments in thirteen combinations of standard search domains from the artificial intelligence literature and heuristic values, averaged over at least one thousand runs each. The section on "Experimental Results" contains additional information on the domains, heuristic values, and experimental setup. Table 1 shows that in all cases but one (Permute-7 with the zero (Z) heuristic[2]) breaking ties towards successors with smallest f-values (statistically) significantly sped up the convergence of LRTA* in terms of travel cost (action executions).

domain and heuristic values		LRTA* that breaks ties . . . towards a largest f-value	randomly	towards a smallest f-value (TB-LRTA*)
8-Puzzle	M	64,746.47	45,979.19	**18,332.39**
	T	911,934.40	881,315.71	**848,814.91**
	Z	2,200,071.25	2,167,621.63	**2,141,219.97**
Gridworld	N	116.50	97.32	**82.08**
	Z	1,817.57	1,675.87	**1,562.46**
Permute-7	A	302.58	298.42	**288.62**
	Z	**16,346.56**	16,853.69	16,996.51
Arrow	F	1,755.42	1,621.26	**1,518.27**
	Z	7,136.93	7,161.71	**7,024.11**
Tower of Hanoi	D	145,246.55	130,113.43	**116,257.30**
	Z	156,349.86	140,361.39	**125,332.52**
Words	L	988.15	813.66	**652.95**
	Z	16,207.19	16,137.67	**15,929.81**

Table 1: Travel Cost to Convergence

1. $s := s_{start}$.
2. $s' := \arg\min_{s'' \in succ(s)} f(s'')$, where $f(s'') := g(s'') + h(s'')$. Break ties in favor of a successor s'' with the smallest value of $c(s, s'') + h(s'')$. Break remaining ties arbitrarily (but systematically).
3. $g(s) :=$ if $s = s_{start}$ then $g(s)$ else $\max(g(s), \min_{s'' \in pred(s)}(g(s'') + c(s'', s)))$.
   $h(s) :=$ if $s = s_{goal}$ then $h(s)$ else $\max(h(s), \min_{s'' \in succ(s)}(c(s, s'') + h(s'')))$.
4. If $s = s_{goal}$, then stop successfully.
5. $s := s'$.
6. Go to 2.

Figure 3: Naive FALCONS (initial, non-functional version)

FALCONS: A Naive Approach

We just showed that TB-LRTA* converges significantly faster than LRTA* because it breaks ties towards successors with smallest f-values. We thus expect real-time search methods that implement this principle more consequently and always move to successors with smallest f-values to converge even faster. Figure 3 shows Naive FALCONS (FAst Learning and CONverging Search), our real-time search method that maintains g- and h-values, always moves to successors with smallest f-values, and breaks ties to minimize the estimated cost to go. Remaining ties can be broken arbitrarily (but systematically). To understand why ties are broken to minimize the estimated cost to go, consider g- and h-values that are perfectly informed. In this case, all states on a shortest path have the same (smallest) f-values and breaking ties to minimize the estimated cost to go ensures that Naive FALCONS moves towards the goal. (All real-time search methods discussed in this paper have the property that they follow a shortest path right away if the g- and h-values are perfectly informed.) To summarize, Naive FALCONS is identical to TB-LRTA* but switches the primary and secondary successor-selection criteria. Unfortunately, we show in the remainder of this section that Naive FALCONS does not necessarily terminate nor converge to a shortest path. In both cases, this is due to Naive FALCONS being unable to increase misleading f-values of states that it visits, because they depend on misleading g- or h-values of states that it does not visit and thus cannot increase.

Naive FALCONS can cycle forever: Figure 4 shows an example of a domain where Naive FALCONS does not terminate for g- and h-values that are admissible but inconsistent. Naive FALCONS follows the cyclic path $s_0, s_1, s_2, s_3, s_2, s_3, \ldots$ without modifying the g- or h-values of any state. For example during the first trial, Naive FALCONS updates $g(s_2)$ to one (based on $g(s_7)$) and $h(s_2)$ to one (based on $h(s_6)$), and thus does not modify them. $g(s_7)$ and $h(s_6)$ are both zero and thus strictly underestimate

[2]This exception will disappear in our results with FALCONS.

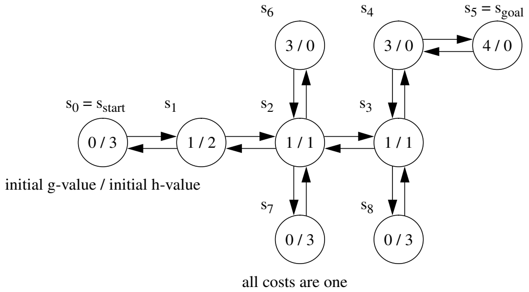

Figure 4: Naive FALCONS Cycles Forever

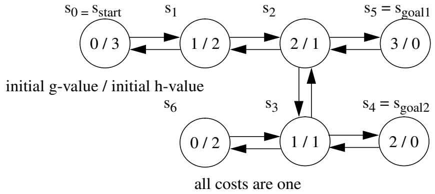

Figure 5: Naive FALCONS Converges to a Suboptimal Path

the true start and goal distances of their respective states. Unfortunately, the successor of state s_2 with the smallest f-value is state s_3. Thus, Naive FALCONS moves to state s_3 and never increases the misleading $g(s_7)$ and $h(s_6)$ values. Similarly, when Naive FALCONS is in state s_3 it moves back to state s_2, and thus cycles forever.

Naive FALCONS can converge to suboptimal paths: Figure 5 shows an example of a domain where Naive FALCONS terminates but converges to a suboptimal path even though the g- and h-values are consistent. Naive FALCONS converges to the suboptimal path s_0, s_1, s_2, s_3, and s_4. The successor of state s_2 with the smallest f-value is state s_3. $f(s_3)$ is two and thus clearly underestimates $f^*(s_3)$. Even though Naive FALCONS moves to state s_3, it never increases its f-value because it updates its g-value to one (based on $g(s_6)$) and $h(s_3)$ to one (based on $h(s_4)$), and thus does not modify them. Naive FALCONS then moves to state s_4. Thus, the trial ends and Naive FALCONS has followed a suboptimal path. Since no g- or h-values changed during the trial, Naive FALCONS has converged to a suboptimal path.

FALCONS: The Final Version

In the previous section, we showed that Naive FALCONS does not necessarily terminate nor converge to a shortest path. Figure 6 shows the final (improved) version of FALCONS that solves both problems. We can prove the following theorems provided that the assumptions described earlier are satisfied.

Theorem 1 *Each trial of FALCONS terminates.*

Theorem 2 *FALCONS eventually converges to a path from the start to the goal if it is reset to the start whenever it reaches the goal and maintains its g- and h-values from one trial to the next one.*

Theorem 3 *The path from the start to the goal that FALCONS eventually converges to is a shortest path.*

1. $s := s_{start}$.
2. $s' := \arg\min_{s'' \in succ(s)} f(s'')$, where $f(s'') := \max(g(s'') + h(s''), h(s_{start}))$. Break ties in favor of a successor s'' with the smallest value of $c(s, s'') + h(s'')$. Break remaining ties arbitrarily (but systematically).
3. $g(s) :=$ if $s = s_{start}$ then $g(s)$
   else $\max(g(s),$
   $\min_{s'' \in pred(s)}(g(s'') + c(s'', s)),$
   $\max_{s'' \in succ(s)}(g(s'') - c(s, s''))).$
   $h(s) :=$ if $s = s_{goal}$ then $h(s)$
   else $\max(h(s),$
   $\min_{s'' \in succ(s)}(c(s, s'') + h(s'')),$
   $\max_{s'' \in pred(s)}(h(s'') - c(s'', s))).$
4. If $s = s_{goal}$, then stop successfully.
5. $s := s'$.
6. Go to 2.

Figure 6: FALCONS (final version)

The proofs of these theorems are nontrivial and much more complex than their counterparts for LRTA*. They can be found in (Furcy & Koenig 2000). In the following, we give some intuitions behind the new value-update and successor-selection rules and show that they solve the problems of Naive FALCONS for the examples introduced in the previous section.

FALCONS terminates: The new value-update rules of FALCONS cause it to terminate. We first derive the new value-update rule for the h-values. It provides more informed but still admissible estimates of the h-values than the old value-update rule, by making better use of information in the neighborhood of the current state. The new value-update rule makes the h-values locally consistent and is similar to the path-max equation used in conjunction with A*. If the h-values are consistent, then there is no difference between the old and new value-update rules. To motivate the new value-update rule, assume that the h-values are admissible and FALCONS is currently in some state s with $s \neq s_{goal}$. The old value-update rule used two lower bounds on the goal distance of state s, namely $h(s)$ and $\min_{s'' \in succ(s)}(c(s, s'') + h(s''))$. The new value-update rule adds a third lower bound, namely $\max_{s'' \in pred(s)}(h(s'') - c(s'', s))$. To understand the third lower bound, note that the goal distance of any predecessor s'' of state s is at least $h(s'')$ since the h-values are admissible. This implies that the goal distance of state s is at least $h(s'') - c(s'', s)$. Since this is true for all predecessors of state s, the goal distance of state s is at least $\max_{s'' \in pred(s)}(h(s'') - c(s'', s))$. The maximum of the three lower bounds then is an admissible estimate of the goal distance of state s and thus becomes its new h-value. This explains the new value-update rule for the h-values. The new value-update rule for the g-values can be derived in a similar way.

As an example, we show that Naive FALCONS with the new value-update rules now terminates in the domain from Figure 4. When Naive FALCONS is in state s_2 during the first trial, it increases both $g(s_2)$ and $h(s_2)$ to two and then moves to state s_3. The successor of state s_3 with the smallest f-value is state s_4, and no longer state s_2, because $f(s_2)$

domain and heuristic values		LRTA* that breaks tie randomly	TB-LRTA*	FALCONS
8-Puzzle	M	45,979.19 (100%)	**18,332.39**	**18,332.39** (39.87%)
	T	881,315.71 (100%)	848,814.91	**709,416.75** (80.50%)
	Z	2,167,621.63 (100%)	2,141,219.97	**1,955,762.18** (90.23%)
Gridworld	N	97.32 (100%)	82.08	**57.40** (58.98%)
	Z	1,675.87 (100%)	1,562.46	**1,440.02** (85.93%)
Permute-7	A	298.42 (100%)	288.62	**284.95** (95.49%)
	Z	16,853.69 (100%)	16,996.51	**16,334.67** (96.92%)
Arrow	F	1,621.26 (100%)	1,518.27	**1,372.62** (84.66%)
	Z	7,161.71 (100%)	7,024.11	**6,763.49** (94.44%)
Tower of Hanoi	D	130,113.43 (100%)	116,257.30	**107,058.94** (82.28%)
	Z	140,361.39 (100%)	125,332.52	**116,389.79** (82.92%)
Words	L	813.66 (100%)	652.95	**569.71** (70.02%)
	Z	16,137.67 (100%)	15,929.81	**15,530.42** (96.24%)

Table 2: Travel Cost to Convergence

domain and heuristic values		LRTA* that breaks tie randomly	TB-LRTA*	FALCONS
8-Puzzle	M	214.37 (100%)	**58.30**	**58.30** (27.20%)
	T	1,428.57 (100%)	1,214.63	**797.26** (55.81%)
	Z	1,428.59 (100%)	1,227.74	**756.47** (52.95%)
Gridworld	N	6.06 (100%)	5.01	**2.90** (47.85%)
	Z	32.02 (100%)	26.30	**19.77** (61.74%)
Permute-7	A	26.91 (100%)	25.55	**22.10** (82.13%)
	Z	117.82 (100%)	92.63	**75.22** (63.84%)
Arrow	F	114.94 (100%)	110.60	**89.01** (77.44%)
	Z	171.50 (100%)	135.13	**105.92** (61.76%)
Tower of Hanoi	D	214.47 (100%)	177.96	**109.13** (50.88%)
	Z	216.77 (100%)	166.55	**101.44** (46.80%)
Words	L	32.82 (100%)	22.72	**18.40** (56.06%)
	Z	71.86 (100%)	55.77	**50.10** (69.72%)

Table 3: Trials to Convergence

domain and heuristic values		LRTA* that breaks tie randomly	TB-LRTA*	FALCONS
8-Puzzle	M	**311.18** (100%)	452.84	452.84 (145.52%)
	T	1,342.75 (100%)	**970.87**	1,057.86 (78.78%)
	Z	81,570.22 (100%)	81,585.44	**81,526.34** (99.95%)
Gridworld	N	**12.15** (100%)	12.70	20.92 (172.18%)
	Z	**182.37** (100%)	182.55	183.13 (100.42%)
Permute-7	A	8.14 (100%)	**7.75**	8.13 (99.88%)
	Z	**2,637.86** (100%)	2,639.13	2,639.13 (100.05%)
Arrow	F	**15.85** (100%)	16.62	33.61 (212.05%)
	Z	**1,016.33** (100%)	1,016.83	1,016.83 (100.05%)
Tower of Hanoi	D	4,457.86 (100%)	**3,654.80**	3,910.46 (87.72%)
	Z	4,839.49 (100%)	4,803.81	**4,801.84** (99.22%)
Words	L	**24.27** (100%)	27.79	37.80 (155.75%)
	Z	**2,899.73** (100%)	2,900.36	2,900.68 (100.03%)

Table 4: Travel Cost of the First Trial

was increased to four. Thus, Naive FALCONS now moves to state s_4 and breaks the cycle. Unfortunately, the new value-update rules are not sufficient to guarantee that Naive FALCONS converges to a shortest path. The domain from Figure 5 still provides a counterexample.

FALCONS converges to a shortest path: The new successor-selection rule of FALCONS causes it to converge to a shortest path by using more informed but still admissible estimates of the f*-values. In the following, we assume that the g- and h-values are admissible and we present two lower bounds on $f^*(s)$. First, $f^*(s)$ is at least $g(s) + h(s)$, since the g- and h-values are admissible. Second, $f^*(s)$ is at least as large as the cost of a shortest path from the start to the goal, a lower bound of which is $h(s_{start})$, since the h-values are admissible. The maximum of the two lower bounds is an admissible estimate of $f^*(s)$ and thus becomes the new f-value of s. This explains the new calculation of the f-values performed by the successor-selection rule. The other parts of the successor-selection rule remain unchanged. The new f-value of state s, unfortunately, cannot be used to update its g- or h-values, because it is unknown by how much to update the g-value and by how much to update the h-value.

As an example, we show that FALCONS now converges to a shortest path in the domain from Figure 5. When FALCONS reaches state s_2 in the first trial, $f(s_3)$ is now three. All three successors of state s_2 have the same f-value and FALCONS breaks ties in favor of the one with the smallest h-value, namely state s_5. Thus, the trial ends and FALCONS has followed a shortest path. Since no g- or h-values changed, FALCONS has converged to a shortest path.

Experimental Results

In this section, we describe our evaluation of FALCONS, that we tested against LRTA* that breaks ties randomly and TB-LRTA*. We used the following domains from the artificial intelligence literature in conjunction with consistent heuristic values: 8-Puzzle (Korf 1990), Gridworld (Ishida 1997), Permute-7 (Holte *et al.* 1994), Arrow (Korf 1980), Tower of Hanoi(Holte *et al.* 1994), and Words (Holte *et al.* 1996). All of these domains satisfy our assumptions. The domains and heuristic values are described in (Furcy & Koenig 2000). Tables 2, 3, and 4 report the travel cost (action executions) until convergence, the number of trials until convergence, and the travel cost of the first trial, respectively. The data in all three cases were averaged over at least one-thousand runs, each with different start states and ways how the remaining ties were broken in case the successor-selection rule and tie-breaking criterion did not uniquely determine the successor. The remaining ties were broken systematically, that is, for each state according to an ordering on its successors that was selected at the beginning of each run. The orderings were randomized between runs. Convergence was detected when no g- or h-values changed during a trial. Additional details on the experimental setup can be found in (Furcy & Koenig 2000).

Table 2 shows that, in all cases, FALCONS converged to a shortest path with a smaller travel cost (action executions) than LRTA* that breaks ties randomly and, in all cases but one, faster than TB-LRTA*. The percentages in the last column compare the travel cost of FALCONS with that of LRTA*. FALCONS converged 18.57 percent faster over all thirteen cases and in one case even 60.13 percent faster. We tested these results for pairwise significance using the (nonparametric) sign test. All the comparisons stated above are significant at the one-percent confidence level with only one exception: The comparison of FALCONS and TB-LRTA* in the Permute-7 domain with the zero (Z) heuristic is significant only at the five-percent confidence level. The heuristic values for each domain are listed in order of their decreasing informedness (sum of the heuristic values over all states). For example, the (completely uninformed) zero (Z) heuristic is listed last. Table 2 then, shows that the speedup of FALCONS over LRTA* was positively correlated with the informedness of the heuristic values. This suggests that FALCONS makes better use of the given heuristic values. Notice that it cannot be the case that FALCONS converges more quickly than LRTA* because it looks at different (or more) states than LRTA* when selecting successor states. FALCONS looks at both the predecessors and successors of the current state while LRTA* looks only at the successors, but all of our domains are undirected and thus every predecessor is also a successor. This implies that FALCONS and LRTA* look at exactly the same states.

Table 3 shows that, in all cases, FALCONS converged to a shortest path with a smaller number of trials than LRTA* that breaks ties randomly and, in all cases but one, faster than TB-LRTA*. FALCONS converged 41.94 percent faster over all thirteen cases and in some cases even 72.80 percent faster.

To summarize, Table 2 and Table 3 show that FALCONS converges faster than LRTA* and even TB-LRTA*, both in terms of travel cost and trials. The first measure determines the total time during the trials. The second measure determines the total time between the trials, for example, to set up the next trial. This is important in case the set-up time is large.

We originally expected that FALCONS would increase the travel cost during the first trial, since the successor-selection rule of LRTA* (minimize the cost to go) has experimentally been shown to result in a small travel cost during the first trial under various conditions. Table 4 shows that, in four of the thirteen cases, the travel cost of FALCONS during the first trial was larger than that of LRTA*; in seven cases it was approximately the same (99 percent to 101 percent); and in two cases it was lower. The travel cost of FALCONS during the first trial was 19.35 percent larger than that of LRTA* over the thirteen cases. Overall, there is no systematic relationship between the travel cost of FALCONS and LRTA* during the first trial, and the sum of planning and plan-execution times is always small for FALCONS, just like for LRTA*.

Discussion of Results and Future Work

In future theoretical work, we intend to derive analytical upper bounds on the travel cost to convergence of FALCONS, similar to those given in (Koenig & Simmons 1995) for LRTA*.

On the experimental side, all of our domains have uniform costs (i.e., all actions have cost one). Although the theory behind FALCONS guarantees that it will terminate and converge to an optimal path even in domains with non-uniform costs, FALCONS may not converge with smaller travel cost than LRTA* in such domains because the successor-selection rule of FALCONS chooses a successor state with lowest f-value even if the cost of moving to it is very large.

domain and heuristic values		LRTA*	FALCONS	FALCONS without g updates
8-Puzzle	M	45,979.19 (100%)	**18,332.39** (39.87%)	19,222.08 (41.81%)
	T	881,315.71 (100%)	**709,416.75** (80.50%)	817,078.12 (92.71%)
Gridworld	N	97.32 (100%)	**57.40** (58.98%)	58.82 (60.44%)
Permute-7	A	298.42 (100%)	284.95 (95.49%)	**263.00** (88.13%)
Arrow	F	1,621.26 (100%)	**1,372.62** (84.66%)	1,533.11 (94.56%)
T. Hanoi	D	130,113.43 (100%)	**107,058.94** (82.28%)	128,987.97 (99.14%)
Words	L	813.66 (100%)	569.71 (70.02%)	**547.35** (67.27%)

Table 5: Travel Cost to Convergence

This property of FALCONS is inherited from A*, which always expands a leaf node of the search tree with the smallest f-value, regardless of the cost of reaching the corresponding state from the current state, since A* only simulates these actions in memory. In future work, we intend to modify the successor-selection rule of FALCONS so that it takes into account the immediate action cost.

So far, one of the main evaluation criteria has been the travel cost to convergence. One may complain that the speedup exhibited by FALCONS over LRTA* comes at an extra computational cost, namely an extra value update per action execution. To decrease the total computational cost (value updates), FALCONS would have to cut the travel cost to convergence at least in half. However, it reduces the travel cost by only 18.57 percent. We also compared FALCONS with a variant of LRTA* that performs two value updates per action execution. This can be done in various ways. Among the ones we tried, our best results were obtained with a variant of LRTA* that first updates $h(s')$ (where s' is the successor of the current state s with the smallest $c(s,s') + h(s')$), then updates $h(s)$, and finally selects the successor s'' of s with the smallest $c(s,s'') + h(s'')$, which may be different from s'. Empirically, this algorithm had a smaller travel cost to convergence than FALCONS. However, we can modify FALCONS so that it never updates the g-values, resulting in one value-update per action execution, just like LRTA*. In (Furcy & Koenig 2000), we prove that our Theorems 1 through 3 can be extended to FALCONS without g updates, provided that the g-values are consistent. Table 5 reports experimental results that clearly show that FALCONS without g updates had a smaller travel cost to convergence than LRTA* (with lookahead one). The speedup was 22.28 percent on average, and up to 58.19 percent. Additional results show that the number of trials to convergence for FALCONS without g updates was 25.97 percent less than for LRTA* on average (and up to 68.71 percent less), and that FALCONS executed an average of 57.51 percent more actions than LRTA* in the first trial.[3] These results are important for two reasons. First, they support the claim that the successor-selection rule of FALCONS speeds up convergence by making better use of the available heuristic knowledge and is able to decrease both the travel cost and computational cost to convergence. Second, they suggest that FALCONS may

[3]In domains with uniform costs, with consistent h-values, and with zero-initialized g-values, FALCONS without g updates reduces to LRTA*. Thus, Table 5 does not show results for completely uninformed heuristic values and our averages do not include them.

benefit from an enhanced successor-selection rule that focuses the search even more sharply around an optimal path by speeding up the learning of more accurate g-values, while still making efficient use of the initial heuristic knowledge.

Finally, we intend to apply FALCONS to domains from real-time control. These domains require real-time action-selection and convergence to optimal behaviors but, at the same time, the setup for each trial is expensive and thus it is important to keep the number of trials small. For learning how to balance poles or juggle devil-sticks (Schaal & Atkeson 1994), for example, the pole needs to be picked up and brought into the initial position before every trial. Domains from real-time control are typically directed and sometimes probabilistic, and we have not yet applied FALCONS to domains with these properties. The main difficulty of applying FALCONS to probabilistic domains is to adapt the notion of f-values to probabilistic domains. In contrast, FALCONS can be applied without modification to directed domains since all of our theoretical results continue to hold.

Conclusions

The research presented in this paper is a first step towards real-time search methods that converge to a shortest path faster than existing real-time search methods. We presented FALCONS, a real-time search method that looks similar to LRTA* but selects successors very differently, proved that it terminates and converges to a shortest path, and demonstrated experimentally, using standard search domains from the artificial intelligence literature, that it converges typically about twenty percent faster to a shortest path and in some cases even sixty percent faster than LRTA* in terms of travel cost (action executions). It also converges typically about forty percent faster and in some cases even seventy percent faster than LRTA* in terms of trials. The key idea behind FALCONS is to maintain f-values, that can be used to focus the search more sharply on the neighborhood of optimal paths. First, we demonstrated that breaking ties in favor of successors with smallest f-values speeds up the convergence of LRTA*, resulting in our TB-LRTA*. Then, we demonstrated that selecting successors with smallest f-values (instead of only breaking ties in favor of them) speeds up the convergence of LRTA* even further, resulting in our FALCONS. Our approach differs from that of Ishida who had to sacrifice the optimality of the resulting paths to speed up the convergence of LRTA* (Ishida & Shimbo 1996; Ishida 1997). It opens up new avenues of research for the design of real-time search methods and reinforcement-learning methods that converge substantially faster to a shortest path, by guiding exploration and exploitation with information that is more directly related to the overall learning objective.

Acknowledgments

We thank Eric Hansen, Maxim Likhachev, Yaxin Liu, Bill Murdock, Joseph Pemberton, and Patrawadee Prasangsit for interesting discussions about the convergence behavior of real-time search methods. The Intelligent Decision-Making Group is supported by an NSF Career Award under contract IIS-9984827. 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 the sponsoring organizations and agencies or the U.S. Government.

References

Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 73(1):81–138.

Bonet, B.; Loerincs, G.; and Geffner, H. 1997. A robust and fast action selection mechanism. In *Proceedings of the National Conference on Artificial Intelligence*, 714–719.

Furcy, D., and Koenig, S. 2000. Speeding up the convergence of real-time search: Empirical setup and proofs. Technical Report GIT-COGSCI-2000/01, College of Computing, Georgia Institute of Technology, Atlanta (Georgia).

Geffner, H., and Bonet, B. 1998. Solving large POMDPs by real-time dynamic programming. Technical report, Departamento de Computación, Universidad Simón Bolivar, Caracas (Venezuela).

Holte, R.; Drummond, C.; Perez, M.; Zimmer, R.; and MacDonald, A. 1994. Searching with abstractions: A unifying framework and new high-performance algorithm. In *Proceedings of the Canadian Conference on Artificial Intelligence*, 263–270.

Holte, R.; Perez, M.; Zimmer, R.; and MacDonald, A. 1996. Hierarchical A*: Searching abstraction hierarchies efficiently. In *Proceedings of the National Conference on Artificial Intelligence*, 530–535.

Ishida, T., and Korf, R. 1991. Moving target search. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 204–210.

Ishida, T., and Shimbo, M. 1996. Improving the learning efficiencies of real-time search. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 305–310.

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

Koenig, S., and Simmons, R. 1995. Real-time search in non-deterministic domains. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1660–1667.

Koenig, S., and Simmons, R. 1998. Solving robot navigation problems with initial pose uncertainty using real-time heuristic search. In *Proceedings of the International Conference on Artificial Intelligence Planning Systems*, 154–153.

Koenig, S. 1999. Exploring unknown environments with real-time search or reinforcement learning. In *Proceedings of the Neural Information Processing Systems*, 1003–1009.

Korf, R. 1980. Towards a model of representation changes. *Artificial Intelligence* 14:41–78.

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

Nilsson, N. 1971. *Problem-Solving Methods in Artificial Intelligence*. McGraw-Hill.

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

Schaal, S., and Atkeson, C. 1994. Robot juggling: An implementation of memory-based learning. *Control Systems Magazine* 14.

Thrun, S. 1992. The role of exploration in learning control with neural networks. In White, D., and Sofge, D., eds., *Handbook of Intelligent Control: Neural, Fuzzy and Adaptive Approaches*. Van Nostrand Reinhold. 527–559.