Real-Time Heuristic Search with LTLf Goals (Appendix)

Jaime Middleton[1,4] , **Rodrigo Toro Icarte**[1,2] , **Jorge A. Baier**[1,2,3]
[1]Department of Computer Science, Pontificia Universidad Católica de Chile, Santiago, Chile
[2]Centro Nacional de Inteligencia Artificial CENIA, Santiago, Chile
[3]Instituto Milenio Fundamentos de los Datos, Santiago, Chile
[4]Tucar SpA, Santiago, Chile
jamiddleton@uc.cl, rodrigo.toro@ing.puc.cl, jabaier@ing.puc.cl

A Theoretical Results From Section 3

In this section, we prove all the theorems from Section 3.

A.1 Proof of Theorem 1

Straightforward from the correctness of the DFA construction and the fact that we define the costs of the cLSP from the costs of the LSP.

A.2 Proof of Theorem 2

Theorem 2 states two properties about the cross-product heuristic h_φ. First, if $\hat{h}$ is admissible for $\mathcal{G}$ then h_φ is admissible for P_φ. And second, if $\hat{h}$ is consistent for $\mathcal{G}$ then h_φ is consistent for P_φ. Below, we prove each of these properties independently.

Admissibility

Let π^* be any minimal cost path for the cLSP P_φ when starting from the cross-product state (s, q). Let us further assume that π^* transitions from one DFA state $q \in Q$ to a different DFA state $q' \in Q$ at most n times.

We will prove by induction in n that if $\hat{h}$ is admissible for $\mathcal{G}$ then h_φ is admissible for P_φ. Let us assume that $\hat{h}(s,t)$ is admissible. We will show that $h_\varphi(s,q) \leq h^*(s,q) = c(\pi^*)$ for any initial $s \in S$ and $q \in Q$. If $n = 0$, then q must be an accepting state. In that case, $h_\varphi(s,q) = h^*(s,q) = 0$.

Now, let us assume that $h_\varphi(s,q) \leq c(\pi^*)$ when π^* changes the DFA state at most n times. Let us now consider the case where π^* changes $n+1$ times its DFA state. Let us say that the first time π^* changes to a different DFA state is in the cross-product state (u, q'). This implies that $q' = \delta(q, \mathcal{L}(u))$ and also that $\Delta(q') < \infty$ (since π^* eventually reaches an accepting state). Then,

$$h_\varphi(s,q) = \min_{t \in S_q} \tilde{c}(s,t) + h_\varphi(t, \delta(q, \mathcal{L}(t))) \tag{1}$$

$$\leq \tilde{c}(s,u) + h_\varphi(u, q') \tag{2}$$

$$\leq \tilde{c}(s,u) + h^*(u, q') \tag{3}$$

$$\leq h^*(s,u) + h^*(u, q') = c(\pi^*) \tag{4}$$

Equation (1) is the direct definition of h_φ. Note that $h_\varphi(s,q)$ contains a minimization across all states $t \in S_q$. Equation (2) follows from the fact that $u \in S_q$. Then, we used the fact that $h_\varphi(u,q') \leq h^*(u,q')$ by the inductive hypothesis in equation (3). Finally, equation (4) follows from the fact that $\tilde{c}(s,u) \leq h^*(s,u)$ because $\tilde{c}(s,u)$ is either equal to ϵ, which is the smallest possible cost of performing any action, or equal to $\hat{h}(s,u)$, which is lower or equal to $h^*(s,u)$ because we assumed that $\hat{h}$ is admissible.

Consistency

Let us assume that $\hat{h}$ is consistent for $\mathcal{G}$. That means that

$$\hat{h}(s,v) \leq c(s,u) + \hat{h}(u,v) \tag{5}$$

for all $(s,u) \in E$ and $v \in S$. Then, we will show that the cross-product h_φ is consistent for P_φ. That is, we will show that

$$h_\varphi(s,q) \leq c(s,u) + h_\varphi(u, \delta(q, \mathcal{L}(u))) \tag{6}$$

for all $(s,u) \in E$ and $q \in Q$. We divide this proof into 3 cases, depending on whether moving from state s to state u causes a transition in the DFA. Let $q' = \delta(q, \mathcal{L}(u))$. Then, there are the following three possibilities (besides the trivial cases where $q \in F$ or $\Delta(q) = \infty$).

Case 1. If $q \neq q'$ and $\Delta(q') = \infty$, then moving from s to u (while the DFA state is q) causes the DFA to move to a DFA state q' from where it is not possible to reach an accepting state. Thus, according to definition 3, $h_\varphi(u,q') = \infty$. Finally, since $h_\varphi(u, \delta(q, \mathcal{L}(u)))$ is infinity, equation (6) holds trivially for this case.

Case 2. If $q \neq q'$ but $\Delta(q') < \infty$, then moving from s to u (while the DFA state is q) causes the DFA to move to a different DFA state q' from where it is possible to reach an accepting state. Then,

$$h_\varphi(s,q) = \min_{t \in S_q} \tilde{c}(s,t) + h_\varphi(t, \delta(q, \mathcal{L}(t))) \tag{7}$$

$$\leq \tilde{c}(s,u) + h_\varphi(u, \delta(q, \mathcal{L}(u))) \tag{8}$$

$$\leq c(s,u) + h_\varphi(u, \delta(q, \mathcal{L}(u))) \tag{9}$$

Equation (7) is the definition of $h_\varphi(s,q)$. Since $q \neq q'$ but $\Delta(q') < \infty$, we know that $u \in S_q$. Equation (8) then follows from the fact that $h_\varphi(s,q)$ is the minimum among the different elements in S_q, including u. Finally, we know that $\tilde{c}(s,u) = \max(\hat{h}(s,u), \epsilon) \leq c(s,u)$ because:

1. $\epsilon \leq c(s,u)$ by definition, and
2. $\hat{h}(s,u) \leq c(s,u)$ because $\hat{h}$ is also admissible.

Replacing $\tilde{c}(s,u) \leq c(s,u)$ in equation (8), we get to equation (9), showing that equation (6) holds in this case too.

Case 3. If $q = q'$, then moving from s to u (while the DFA state is q) does not cause a transition to a different DFA state. By definition, we know that:

$$h_\varphi(u, q) = \min_{t \in S_q} \tilde{c}(u, t) + h_\varphi(t, \delta(q, \mathcal{L}(t))) \tag{10}$$

$$h_\varphi(u, q) = \tilde{c}(u, v) + h_\varphi(v, \delta(q, \mathcal{L}(v))) \tag{11}$$

for some $v \in S_q$. Then,

$$h_\varphi(s, q) = \min_{t \in S_q} \tilde{c}(s, t) + h_\varphi(t, \delta(q, \mathcal{L}(t))) \tag{12}$$

$$\leq \tilde{c}(s, v) + h_\varphi(v, \delta(q, \mathcal{L}(v))) \tag{13}$$

Equation (13) results from replacing t by $v \in S_q$, where v is the same state from equation (11). We also know that:

$$\tilde{c}(s, v) = \max(\hat{h}(s, v), \epsilon) \leq c(s, u) + \hat{h}(u, v) \tag{14}$$

because if $\tilde{c}(s, v) = \epsilon$ then

$$\tilde{c}(s, v) = \epsilon \leq c(s, u) \leq c(s, u) + \hat{h}(u, v).$$

And if $\tilde{c}(s, v) = \hat{h}(s, v)$, then

$$\tilde{c}(s, v) = \hat{h}(s, v) \leq c(s, u) + \hat{h}(u, v)$$

because $\hat{h}$ is consistent. Finally, if we replace equation (14) in (13), we get the following relation:

$$h_\varphi(s, q) \leq c(s, u) + \hat{h}(u, v) + h_\varphi(u, \delta(q, \mathcal{L}(u))) \tag{15}$$

$$\leq c(s, u) + \tilde{c}(u, v) + h_\varphi(u, \delta(q, \mathcal{L}(u))) \tag{16}$$

$$= c(s, u) + h_\varphi(u, \delta(q, \mathcal{L}(u))) \tag{17}$$

where equation (16) uses that $\hat{h}(u, v) \leq \max(\hat{h}(u, v), \epsilon) = \tilde{c}(u, v)$ by definition and equation (17) results from replacing equation (11) in (16). Thus, equation (6) also holds in this case.

A.3 Formal Analysis of Myopic Heuristics

Formally, a myopic heuristic is defined as follows:

Definition 4 (The myopic heuristic)**.** *Given an LSP $P = (S, E, c, s_{start}, \mathcal{P}, \varphi, \mathcal{L})$ and a goal-independent heuristic $\hat{h} : S \times S \mapsto [0, \infty)$ for $\mathcal{G} = (S, E, c)$, we define the myopic heuristic $\tilde{h}_\varphi : S \times Q \mapsto [0, \infty)$ as follows. Let $A_\varphi = (Q, 2^{\mathcal{P}}, \delta, q_0, F)$ be the DFA representation of the LTL$_f$ formula φ. And let $S_q \subseteq S$ be the set of all states that cause a transition from $q \in Q$ to $q' \in Q$ such that $q \neq q'$ while an accepting state can be reached from q' in A_φ. That is, $S_q = \{s \in S \mid q' = \delta(q, \mathcal{L}(s)), q \neq q', \Delta(q') < \infty\}$. Then,*

- $\tilde{h}_\varphi(s, q) = 0$ *if* $q \in F$.
- $\tilde{h}_\varphi(s, q) = \infty$ *if* $\Delta(q) = \infty$.
- $\tilde{h}_\varphi(s, q) = \min_{t \in S_q} \tilde{c}(s, t)$*, otherwise.*

where $\tilde{c}(s, t) = \max(\hat{h}(s, t), \epsilon)$ and ϵ is the minimum cost of any transitions in $\mathcal{G}$. That is, $\epsilon = \min\{c(s, t) \mid (s, t) \in E\}$.

The main advantage of myopic heuristics over cross-product heuristics is that they can be computed faster. Indeed, the complexity of computing $\tilde{h}_\varphi(s, q)$ is $O(|S_q|)$ whereas the complexity of computing $h_\varphi(s, q)$ is $O(v^2)$ where $v = 1 + \sum_{q \in Q} |S_q|$. However, myopic heuristics are weaker than cross-product heuristics, as stated in the following theorem.

Theorem 5 (Myopic heuristics are weaker than cross-product heuristics)**.** *Let $P = (S, E, c, s_0, \mathcal{P}, \varphi, \mathcal{L})$ be an LSP and $\hat{h} : S \times S \mapsto [0, \infty)$ be a heuristic for $\mathcal{G} = (S, E, c)$. Let h_φ be the cross-product heuristic constructed from P and $\hat{h}$ and $\tilde{h}_\varphi$ be the myopic heuristic constructed from P and $\hat{h}$. Then, for all $(s, q) \in S \times Q$, $\tilde{h}_\varphi(s, q) \leq h_\varphi(s, q)$.*

Proof. h_φ and $\tilde{h}_\varphi$ are equivalent in all cases except when $q \notin F$ and $\Delta(q) < \infty$. In that case,

$$\tilde{h}_\varphi(s, q) = \min_{t \in S_q} \tilde{c}(s, t) \tag{18}$$

$$\leq \min_{t \in S_q} \tilde{c}(s, t) + h_\varphi(t, \delta(q, \mathcal{L}(t))) \tag{19}$$

$$= h_\varphi(s, q) \tag{20}$$

□

That said, myopic heuristics have two quite positive theoretical results. First, if the based heuristic of a myopic heuristic is admissible, then the myopic heuristic is also admissible.

Theorem 6 (Admissibility of the myopic heuristic)**.** *Let $P = (S, E, c, s_0, \mathcal{P}, \varphi, \mathcal{L})$ be an LSP, P_φ be the cLSP for P, and $\hat{h} : S \times S \mapsto [0, \infty)$ be a heuristic for $\mathcal{G} = (S, E, c)$. Let $\tilde{h}_\varphi$ be the myopic heuristic constructed from P and $\hat{h}$. If $\hat{h}$ is admissible for $\mathcal{G}$ then $\tilde{h}_\varphi$ is admissible for P_φ.*

Proof. From Theorem 1, we know that if $\hat{h}$ is admissible, then the cross-product heuristic h_φ is also admissible. That means that $h_\varphi(s, q) \leq h^*(s, q)$ for all $(s, q) \in S \times Q$. Because of Theorem 5, we know that the myopic heuristic is defined such that, for all $(s, q) \in S \times Q$, $\tilde{h}_\varphi(s, q) \leq h_\varphi(s, q)$. Thus, $\tilde{h}_\varphi(s, q) \leq h^*(s, q)$, which implies that $\tilde{h}_\varphi$ is admissible. □

The second theoretical property of myopic heuristics is the following. If the base heuristic of a myopic heuristic is consistent, then the myopic heuristic is consistent. This property is formally stated in the following theorem.

Theorem 7 (Consistency of the myopic heuristic)**.** *Let $P = (S, E, c, s_0, \mathcal{P}, \varphi, \mathcal{L})$ be an LSP, P_φ be the cLSP for P, and $\hat{h} : S \times S \mapsto [0, \infty)$ be a heuristic for $\mathcal{G} = (S, E, c)$. Let $\tilde{h}_\varphi$ be the myopic heuristic constructed from P and $\hat{h}$. If $\hat{h}$ is consistent for $\mathcal{G}$ then $\tilde{h}_\varphi$ is consistent for P_φ.*

Proof. Let us assume that $\hat{h}$ is consistent for $\mathcal{G}$. Thus,

$$\hat{h}(s, v) \leq c(s, u) + \hat{h}(u, v) \tag{21}$$

for all $(s, u) \in E$ and $v \in S$. Then, we will show that the myopic heuristic $\tilde{h}_\varphi$ is consistent for P_φ. That is,

$$\tilde{h}_\varphi(s, q) \leq c(s, u) + \tilde{h}_\varphi(u, \delta(q, \mathcal{L}(u))) \tag{22}$$

for all $(s,u) \in E$ and $q \in Q$. We split this proof into 3 cases, depending on whether moving from state s to state u causes a transition in the DFA. Let $q' = \delta(q, \mathcal{L}(u))$. Then, there are the following three possibilities (besides the trivial cases where $q \in F$ or $\Delta(q) = \infty$).

Case 1. If $q \neq q'$ and $\Delta(q') = \infty$, then moving from s to u (while the DFA state is q) causes the DFA to move to a DFA state q' from where it is not possible to reach an accepting state. Thus, according to definition 4, $\tilde{h}_\varphi(u, q') = \infty$. Finally, since $\tilde{h}_\varphi(u, \delta(q, \mathcal{L}(u)))$ is infinity, equation (22) holds.

Case 2. If $q \neq q'$ but $\Delta(q') < \infty$, then moving from s to u (while the DFA state is q) causes the DFA to move to a different DFA state q' from where it is possible to reach an accepting state. Then,

$$\tilde{h}_\varphi(s,q) = \min_{t \in S_q} \tilde{c}(s,t) \tag{23}$$
$$\leq \tilde{c}(s,u) \tag{24}$$
$$\leq c(s,u) + \tilde{h}_\varphi(u, \delta(q, \mathcal{L}(u))) \tag{25}$$

Equation (23) is the definition of $\tilde{h}_\varphi(s,q)$. Since $q \neq q'$ but $\Delta(q') < \infty$, we know that $u \in S_q$. Equation (24) then follows from the fact that $\tilde{h}_\varphi(s,q)$ is the minimum among the different elements in S_q, including u. Finally, we know that $\tilde{c}(s,u) = \max(\hat{h}(s,u), \epsilon) \leq c(s,u)$ because:

1. $\epsilon \leq c(s,u)$ by definition, and
2. $\hat{h}(s,u) \leq c(s,u)$ because $\hat{h}$ is also admissible.

Replacing $\tilde{c}(s,u) \leq c(s,u)$ in equation (24), we get to equation (25), showing that equation (22) holds in this case too.

Case 3. If $q = q'$, then moving from s to u (while the DFA state is q) does not cause a transition to a different DFA state. By definition, we know that:

$$\tilde{h}_\varphi(u,q) = \min_{t \in S_q} \tilde{c}(u,t) = \tilde{c}(u,v) \tag{26}$$

for some $v \in S_q$. Then,

$$\tilde{h}_\varphi(s,q) = \min_{t \in S_q} \tilde{c}(s,t) \leq \tilde{c}(s,v) \tag{27}$$

Equation (27) results from replacing t by $v \in S_q$, where v is the same state from equation (26). We also know that:

$$\tilde{c}(s,v) = \max(\hat{h}(s,v), \epsilon) \leq c(s,u) + \hat{h}(u,v) \tag{28}$$

because if $\tilde{c}(s,v) = \epsilon$ then

$$\tilde{c}(s,v) = \epsilon \leq c(s,u) \leq c(s,u) + \hat{h}(u,v).$$

And if $\tilde{c}(s,v) = \hat{h}(s,v)$, then

$$\tilde{c}(s,v) = \hat{h}(s,v) \leq c(s,u) + \hat{h}(u,v)$$

because $\hat{h}$ is consistent. Finally, if we replace equation (28) in (27), we get the following relation:

$$\tilde{h}_\varphi(s,q) \leq c(s,u) + \hat{h}(u,v) \tag{29}$$
$$\leq c(s,u) + \tilde{c}(u,v) \tag{30}$$
$$= c(s,u) + \tilde{h}_\varphi(u, \delta(q, \mathcal{L}(u))) \tag{31}$$

where equation (30) uses that $\hat{h}(u,v) \leq \max(\hat{h}(u,v), \epsilon) = \tilde{c}(u,v)$ by definition and equation (31) results from replacing equation (26) in (30). Thus, equation (22) also holds in this case. □

B Theoretical Results From Section 4

In this section, we prove all the theorems from Section 4.

B.1 Proof of Theorem 3

To prove this theorem, we first need the following lemma and theorem:

Lemma 1 ([Koenig and Sun, 2009; Rivera *et al.*, 2015])**.** *If h is a consistent heuristic for final-state search problem $P = (S, E, c, s_{start}, G)$, and $T \subseteq S \setminus G$, then after running* `Dijkstra-Update(`$T, \partial T$`)`*, h remains consistent.*

Theorem 8. *Let P be an LSP and h be a consistent heuristic for P_φ. When LTL-LRTA* is run using P and h as input, h remains consistent after executing Line 8.*

Proof. Follows from Lemma 1, the fact that LTL-LRTA* by construction searches over P_φ, and that `Bounded-A`* leaves goal states in its *Open* queue. □

Then, the proof of Theorem 3 follows directly from Theorem 1, Theorem 8, and the fact that LTL-LRTA* run over P behaves exactly as LSS-LRTA* when run over the cLSP for P.

B.2 Proof of Theorem 4

To prove this theorem, we use of the following result.

Theorem 9. *Let P be an LSP, P_φ its corresponding cLSP, and h be a consistent heuristic for P_φ. When LTL-LRTA*$_A$ is run using P and h as input, h remains consistent after executing Line 8*

Proof. It follows directly from Lemma 1, since the way we modify `Bounded-A`* does not change the fact that *Open* $= \partial Closed$. □

We now prove Theorem 4 by contradiction. Assume execution enters an infinite loop. Without loss of generality assume that $(s_1, q_1)(s_2, q_2)\ldots(s_k, q_k)(s_1, q_1)$ defines the loop visited by the algorithm and that we have reached a point during the execution where (s_1, q_1) is a state at which a search episode is run. Assume further that at this point in execution the heuristic function is not updated anymore. This last assumption can indeed be made since the the heuristic remains consistent (cf. Theorem 9) and hence admissible and bounded by h^*. Moreover, the increments on the heuristic are finite and given by the costs of the graph thus convergence is reached in a finite number of steps.

We first prove the following property (**P1**), which is that $q_1 = q_2 = \cdots = q_k$. This follows from the fact that any path $\pi_\varphi = ((s_1, q_1), \ldots, (s_n, q_n))$ from $s_{start\,\varphi}$ is such that $\Delta(q_i) < \Delta(q_j)$ for every i, j such that $1 \leq i < j \leq n$ when $q_i \neq q_j$. Thus if $q_1 = q_2 = \cdots = q_k$ did not hold we would reach a contradiction such as $\Delta(q_1) \neq \Delta(q_1)$. This means that every call to our modified `Bounded-A`* call only explores states which refer to the same automaton state, or, more precisely, that the *Open* list only contains states which refer to the same state q_1.

The important consequence of **P1** is that the decision made by the algorithm is to move towards the state with minimum f-value in *Open* (just like LTL-LRTA* would). Let us now assume that we run a search at state (s_1, q_1) and that such a search ends with (s_v, q_v) at the top of $Open$. Then, because of the way the heuristic would be updated: $h(s_w, q_w) = c(s_w, s_{w+1}) + h(s_{w+1}, q_{w+1})$, for every $w \in \{1, 2, \ldots, v-1\}$. Applying the same reasoning to each decision, and using the fact that costs are positive, we obtain that $h(s_1, q_1) > h(s_2, q_2) > \cdots > h(s_k, q_k) > h(s_1, q_1)$, which is a contradiction.

C Pseudo Codes

In this section we explain and show the pseudo-code for Bounded A* and `Dijkstra-Update`.

Algorithm 1: Bounded A*, a bounded version of A*.

Input : A tuple (s_{root}, q_{root}) in $S \times Q$ indicating the root of the search
Output: A tuple $Open, Closed$ with the priority queue, $Open$, and the set of expanded states, $Closed$.

0 **function** `Stop()`
1     $(s, q) \leftarrow$ element at the top of $Open$
2   **return** $q \in F \vee expansions \geq k$
3 **for each** $(s, q) \in S \times Q$
4     $g(s, q) \leftarrow \infty$
5 $g(s_{root}, q_{root}) \leftarrow 0$
6 $f(s_{root}, q_{root}) \leftarrow h(s_{root}, q_{root})$
7 $expansions \leftarrow 0$
8 **if** *Automata Subgoaling* **then**
9     $Open \leftarrow$ Empty priority queue whose priority is given by (Δ, f)
10 **else**
11     $Open \leftarrow$ Empty priority queue whose priority is given by f
12 $Closed \leftarrow \emptyset$
13 Insert (s_{root}, q_{root}) into $Open$
14 **while** *not* `Stop()` *and Open is non-empty* **do**
15     $(s, q) \leftarrow$ extract element at the top of $Open$
16   Insert (s, q) into $Closed$
17   **for each** $t \in N(s)$
18         $r \leftarrow \delta(q, \mathcal{L}(t))$
19   **if** $g(t, r) > g(s, q) + c(s, t)$ **then**
20             $g(t, r) \leftarrow g(s, q) + c(s, t)$
21             $f(t, r) \leftarrow g(t, r) + h(t, r)$
22             $(t, r).back = (s, t)$
23   **if** $(t, r) \in Closed$ **then**
24   Remove (t, r) from $Closed$
25   Insert (t, r) in $Open$
26     $expansions \leftarrow expansions + 1$
27 **return** $Open, Closed$

The pseudo code for Bounded A* is shown in Algorithm 1. Bounded A* takes as inpur the current state of the agent, namely (s, q), and performs at most k expansions of A*. Note that the key difference between Bounded A* and standard A* is the termination condition. Bounded A* terminates as soon as k expansions have been made or when a goal state appears at the top of $Open$ (Line 2).

Our method, Automata subgoaling, modifies standard Bounded A* by greedily preferring states that make progress

Algorithm 2:

Procedure `Dijkstra-Update`(*Open, Closed*)
    `/* We assume` $Open$ `is a queue ordered by` h`-value */`
    for each $s \in Closed$
        $h(s) \leftarrow \infty$
    while $Closed \neq \emptyset$ **do**
        Extract s with minimum h-$value$ from $Open$
        if $s \in Closed$ **then**
            delete s from $Closed$
        for each s' *such that* $(s', s) \in E$
            if $s' \in Closed$ *and* $h(s') > h(s) + c(s', s)$ **then**
                $h(s') \leftarrow c(s', s) + h(s)$
                Insert s' into $Open$

over the DFA. Concretely, rather than ordering the $Open$ priority queue by the f-value of each state, it first orders the states by Δ and then breaks ties by their f-values. Recall that $\Delta(q)$ is the minimum distance between q and an accepting state in the DFA. For instance, in the DFA from Figure 1c, $\Delta(q_i)$=3, $\Delta(q_1)$=2, $\Delta(q_2)$=1, $\Delta(q_3)$=0, and $\Delta(q_4)$=∞.

Finally, the pseudo code of `Dijkstra-Update` is shown in Algorithm 2. Our method uses `Dijkstra-Update` to update the heuristic values after Bounded A* is performed. `Dijkstra-Update` takes as input the $Open$ and $Closed$ sets returned by Bounded A* and updates the heuristic of all the states in $Closed$ w.r.t. the best path to the states in $Open$.

D Extra Experiments

This section provides further details about our experimental evaluation.

D.1 LTL$_f$ Goals

We ran experiments using the following LTL$_f$ goals:

1. $\Diamond(a \wedge \bigcirc\Diamond(b \wedge \bigcirc\Diamond(c \wedge \bigcirc\Diamond(d \wedge \bigcirc\Diamond a))))$
2. $(\neg b) \mathsf{U} (a \wedge \bigcirc\Diamond b) \wedge ((\neg c) \mathsf{U} (b \wedge \bigcirc\Diamond c))$
3. $\Diamond(a \wedge \bigcirc\Diamond(b \wedge \bigcirc\Diamond(c \vee (d \wedge \bigcirc\Diamond e))))$
4. $\Box\neg d \wedge (\neg(b \vee c) \mathsf{U} a) \wedge (\neg c \mathsf{U} b) \wedge \Diamond(a \wedge \bigcirc\Diamond(b \wedge \bigcirc\Diamond c))$
5. $\Diamond(a \wedge \bigcirc\Diamond(b \wedge \bigcirc\Diamond(c \wedge \bigcirc\Diamond(d \wedge \bigcirc\Diamond(e \wedge \bigcirc\Diamond(f \wedge \bigcirc\Diamond(g \wedge \bigcirc\Diamond(h \wedge \bigcirc\Diamond(i \wedge \bigcirc\Diamond(a) \ldots)$

These goals allow us to empirical test how different features of LTL$_f$ affect the performance of our methods. In particular, goals 1 and 5 ask the agent to complete a sequence of tasks. Goals 2 and 4 define partial order tasks, which adds strict restrictions on the path of the agent, such as to not solve b before solving a (goal 2). Goal 4 includes the safety constraints of never reaching location d. Finally, goal 3 contains a disjunctive goal, where the agent can decide to either complete task a-b-c or task a-b-d-e.

D.2 Results using Different Lookaheads and Goals

In addition to the results from the paper, here we present an empirical comparison between LTL-LRTA* and LTL-LRTA*$_A$ with respect to the different goals and lookahead

Table 4: Result for different $\mathrm{LTL_f}$ goals

$\mathrm{LTL_f}$ Goal	LTL-LRTA*	Ties	LTL-LRTA*$_A$	Total
#1	1140	739	3521	5400
#2	1200	407	3793	5400
#3	1105	1043	3252	5400
#4	1082	1089	3229	5400
#5	1084	1082	3234	5400
Total	5611	4360	17029	27000

values. Specifically, Table 4 shows the number of instances that each approach solves faster w.r.t. the different $\mathrm{LTL_f}$ goals. And Tables 5–14 shows the average number of steps taken by each algorithm over 50 instances as we vary the lookahead.

References

[Koenig and Sun, 2009] Sven Koenig and Xiaoxun Sun. Comparing real-time and incremental heuristic search for real-time situated agents. *Autonomous Agents and Multi-Agent Systems*, 18(3):313–341, 2009.

[Rivera *et al.*, 2015] Nicolas Rivera, Jorge A. Baier, and Carlos Hernández. Incorporating weights into real-time heuristic search. *Artificial Intelligence*, 225:1–23, 2015.

Table 5: Result with 3 instances of each letter, and goal #1

Configuration			**Lookahead**					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h^1_φ	LTL-LRTA*	149967.2	102831.5	72362.6	57683.6	39083.1	29726.3
		LTL-LRTA*$_A$	**129710.8**	**86565.3**	**60293.5**	**44397.8**	**30719.2**	**21139.3**
	h^M_φ	LTL-LRTA*	**2036.6**	**1683.2**	1520.6	1395.6	1363.1	1304.6
		LTL-LRTA*$_A$	2048.1	1702.7	**1502.8**	**1384.4**	**1315.7**	**1267.0**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	1287117.3	674678.9	365327.5	202841.8	109234.2	59717.4
		LTL-LRTA*$_A$	**2241.0**	**1861.8**	**1631.5**	**1520.8**	**1445.6**	**1396.1**
StarCraft	h^1_φ	LTL-LRTA*	**249221.0**	206633.4	145682.1	101859.2	77413.7	54276.3
		LTL-LRTA*$_A$	254482.0	**172381.9**	**131076.4**	**89502.7**	**65955.3**	**46105.3**
	h^M_φ	LTL-LRTA*	189377.8	**120656.2**	**64807.5**	**37609.0**	**22915.3**	12109.6
		LTL-LRTA*$_A$	**188795.8**	121302.3	67785.0	38697.6	24027.0	**11238.5**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	2533327.5	1409876.8	783593.6	417667.2	216393.2	114796.4
		LTL-LRTA*$_A$	**195511.9**	**110990.4**	**61420.4**	**35611.8**	**20448.6**	**11850.7**
Maze	h^1_φ	LTL-LRTA*	272971.0	**189122.4**	152342.1	105676.4	69395.9	46592.4
		LTL-LRTA*$_A$	**271993.7**	191897.0	**142695.7**	**96858.2**	**60612.0**	**43292.4**
	h^M_φ	LTL-LRTA*	881672.5	**465973.3**	**267284.6**	**143151.3**	79642.1	**42808.6**
		LTL-LRTA*$_A$	**874816.9**	471560.2	271412.0	145185.1	**77045.0**	43802.9
	$\tilde{h}^M_\varphi$	LTL-LRTA*	1600877.5	937225.1	519547.1	274825.3	144887.8	79293.4
		LTL-LRTA*$_A$	**783841.8**	**453399.1**	**244846.8**	**137768.0**	**73354.9**	**43817.9**

Table 6: Result with 25 instances of each letter, and goal #1

Configuration			**Lookahead**					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h^1_φ	LTL-LRTA*	16715.6	11078.7	10071.9	7097.2	5612.4	3941.7
		LTL-LRTA*$_A$	**15453.4**	**10800.2**	**6712.1**	**5719.6**	**3754.2**	**1927.6**
	h^M_φ	LTL-LRTA*	597.5	503.0	466.3	452.9	437.0	423.5
		LTL-LRTA*$_A$	**582.6**	**496.3**	**463.0**	**427.6**	**413.6**	**394.9**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	44059.3	24023.3	12801.6	8642.1	4426.3	3101.7
		LTL-LRTA*$_A$	**695.8**	**581.7**	**541.5**	**480.1**	**464.5**	**440.9**
StarCraft	h^1_φ	LTL-LRTA*	**22662.4**	19232.8	12866.5	10589.1	7628.8	4229.5
		LTL-LRTA*$_A$	23525.9	**15369.7**	**9398.6**	**9047.3**	**5529.7**	**3125.2**
	h^M_φ	LTL-LRTA*	**10935.1**	**6348.8**	**4071.8**	**2551.6**	1912.2	**1120.3**
		LTL-LRTA*$_A$	12357.5	6645.2	4425.7	2603.8	**1812.6**	1167.6
	$\tilde{h}^M_\varphi$	LTL-LRTA*	91995.1	53578.7	29734.2	16549.8	8944.3	5187.2
		LTL-LRTA*$_A$	**10758.6**	**5059.9**	**3059.6**	**2093.1**	**1592.9**	**1131.7**
Maze	h^1_φ	LTL-LRTA*	20450.2	14281.0	11791.3	7690.6	5908.4	4265.8
		LTL-LRTA*$_A$	**18076.3**	**12453.5**	**9214.3**	**5402.0**	**4258.9**	**3141.9**
	h^M_φ	LTL-LRTA*	**23105.5**	15322.6	8115.6	**4535.3**	**2822.4**	**1587.3**
		LTL-LRTA*$_A$	23108.0	**15098.2**	**7871.3**	4759.6	3048.3	1745.3
	$\tilde{h}^M_\varphi$	LTL-LRTA*	69459.9	38610.8	21498.3	11902.1	6276.2	3550.6
		LTL-LRTA*$_A$	**22511.5**	**12655.8**	**8320.8**	**4725.9**	**2444.7**	**1692.5**

Table 7: Result with 3 instances of each letter, and goal #2

Configuration			Lookahead					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h^1_φ	LTL-LRTA*	267385.4	177080.3	157434.8	112976.4	83817.6	58890.3
		LTL-LRTA*$_A$	**257744.8**	**164640.8**	**125875.4**	**91228.5**	**55785.1**	**40787.4**
	h^M_φ	LTL-LRTA*	4050.0	3365.8	3029.1	2846.9	2714.7	2622.9
		LTL-LRTA*$_A$	**4048.7**	**3355.2**	**3009.1**	**2783.2**	**2669.3**	**2578.7**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	2790316.9	1466474.7	808174.7	446854.2	233911.1	127183.4
		LTL-LRTA*$_A$	**4568.0**	**3801.8**	**3503.3**	**3259.0**	**3142.8**	**3014.8**
StarCraft	h^1_φ	LTL-LRTA*	519610.9	365776.6	264405.1	198012.8	136279.2	96253.6
		LTL-LRTA*$_A$	**470331.2**	**351251.5**	**257256.5**	**180411.6**	**129796.3**	**85895.8**
	h^M_φ	LTL-LRTA*	**378254.2**	**213222.7**	120417.0	70214.9	**39984.5**	24387.3
		LTL-LRTA*$_A$	378982.1	216594.9	**118683.8**	**69000.6**	40667.4	**23196.6**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	4805839.4	2692205.2	1478040.4	800434.7	404783.5	218677.7
		LTL-LRTA*$_A$	**313086.6**	**178443.1**	**97095.7**	**54185.3**	**34034.8**	**21661.7**
Maze	h^1_φ	LTL-LRTA*	551845.7	396005.6	318250.0	205227.4	157227.4	**91536.6**
		LTL-LRTA*$_A$	**532569.0**	**362188.6**	**305820.6**	**195933.9**	**133005.5**	93665.0
	h^M_φ	LTL-LRTA*	1616873.7	**925686.8**	**492924.1**	273301.4	**150672.9**	**80625.3**
		LTL-LRTA*$_A$	**1595544.5**	927482.7	499627.1	**272127.1**	155482.0	83286.3
	$\tilde{h}^M_\varphi$	LTL-LRTA*	3923808.4	2303958.8	1211880.9	664265.3	337188.5	180663.3
		LTL-LRTA*$_A$	**1331384.6**	**759205.9**	**424897.1**	**230200.9**	**132408.1**	**67618.2**

Table 8: Result with 25 instances of each letter, and goal #2

Configuration			Lookahead					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h^1_φ	LTL-LRTA*	31367.7	23510.6	19400.3	15345.9	11566.9	8734.9
		LTL-LRTA*$_A$	**29977.8**	**17302.3**	**13214.6**	**10008.7**	**6065.6**	**4521.5**
	h^M_φ	LTL-LRTA*	1176.0	959.2	891.1	816.5	812.4	789.2
		LTL-LRTA*$_A$	**1163.8**	**945.3**	**847.8**	**768.2**	**741.1**	**711.2**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	75409.6	42577.2	25012.6	14924.9	7985.9	4949.4
		LTL-LRTA*$_A$	**1436.1**	**1133.3**	**1032.7**	**944.2**	**888.5**	**859.0**
StarCraft	h^1_φ	LTL-LRTA*	52157.3	36462.6	28987.9	20181.5	15939.6	10450.4
		LTL-LRTA*$_A$	**45332.2**	**31803.0**	**24565.9**	**16850.2**	**11698.5**	**8474.8**
	h^M_φ	LTL-LRTA*	31787.8	**18894.6**	11159.4	7109.0	**4532.9**	**2679.5**
		LTL-LRTA*$_A$	**31338.1**	19295.7	**10802.2**	**6584.6**	4923.1	2831.5
	$\tilde{h}^M_\varphi$	LTL-LRTA*	232472.3	133887.1	71283.3	40635.4	21995.2	10773.9
		LTL-LRTA*$_A$	**23609.1**	**14085.3**	**8308.1**	**5196.0**	**3597.1**	**2556.4**
Maze	h^1_φ	LTL-LRTA*	42789.7	35759.8	27067.3	18647.4	12898.3	9233.3
		LTL-LRTA*$_A$	**35203.7**	**28703.2**	**21770.8**	**15587.0**	**11208.2**	**6978.7**
	h^M_φ	LTL-LRTA*	68386.6	**37259.8**	22330.6	13399.4	7986.2	4646.2
		LTL-LRTA*$_A$	**67145.2**	38121.8	**22022.6**	**12401.7**	**7733.6**	**4534.4**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	161142.1	97342.9	51486.1	29859.4	16243.4	8491.0
		LTL-LRTA*$_A$	**60300.7**	**34725.1**	**19210.9**	**11168.0**	**6971.6**	**3633.5**

Table 9: Result with 3 instances of each letter, and goal #3

Configuration			Lookahead					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h^1_φ	LTL-LRTA*	95003.1	59870.6	49194.6	35722.1	24400.3	16571.9
		LTL-LRTA*$_A$	**84166.2**	**53758.3**	**41251.3**	**30048.2**	**20174.1**	**12820.3**
	h^M_φ	LTL-LRTA*	**1509.3**	**1242.6**	**1106.8**	1060.9	**995.8**	968.0
		LTL-LRTA*$_A$	1510.9	1251.6	1109.1	**1049.5**	1007.5	**959.4**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	432657.1	221969.2	125550.7	69445.4	37240.7	20237.8
		LTL-LRTA*$_A$	**1583.4**	**1336.3**	**1203.6**	**1140.8**	**1088.3**	**1059.9**
StarCraft	h^1_φ	LTL-LRTA*	148702.2	105977.1	78206.9	68256.7	35483.1	30191.6
		LTL-LRTA*$_A$	**129168.7**	**92637.9**	**64783.6**	**51593.2**	**33836.2**	**27011.4**
	h^M_φ	LTL-LRTA*	**164211.9**	97062.6	**51473.8**	29224.5	**17498.7**	10245.5
		LTL-LRTA*$_A$	164334.2	**95815.0**	54951.5	**27907.0**	17534.6	**10224.6**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	1189222.9	670426.3	356032.2	198289.1	101822.2	52433.2
		LTL-LRTA*$_A$	**139554.5**	**79131.1**	**44525.5**	**25405.0**	**14431.7**	**7387.2**
Maze	h^1_φ	LTL-LRTA*	**177264.9**	143202.5	93037.3	**62552.1**	**39412.4**	28211.5
		LTL-LRTA*$_A$	192203.4	**139003.3**	**85203.9**	65113.3	41165.4	**27987.6**
	h^M_φ	LTL-LRTA*	496045.4	287798.1	160235.5	88918.2	45444.4	24924.4
		LTL-LRTA*$_A$	**482009.4**	**286132.6**	**155367.5**	**86100.2**	**44614.2**	**23832.6**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	1493232.0	817207.6	440827.4	240789.3	134350.2	69396.8
		LTL-LRTA*$_A$	**464096.3**	**261671.7**	**144924.8**	**80714.7**	**41366.1**	**22023.9**

Table 10: Result with 25 instances of each letter, and goal #3

Configuration			Lookahead					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h^1_φ	LTL-LRTA*	10592.0	7781.2	5467.0	3821.7	3948.3	2704.9
		LTL-LRTA*$_A$	**10351.0**	**5893.7**	**4698.0**	**2918.4**	**2601.3**	**1477.9**
	h^M_φ	LTL-LRTA*	**387.8**	**339.4**	337.6	285.7	284.6	281.7
		LTL-LRTA*$_A$	399.6	339.8	**316.9**	**274.9**	**267.3**	**263.5**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	23104.1	12028.9	7681.2	3804.9	2742.3	1544.8
		LTL-LRTA*$_A$	**472.0**	**375.3**	**356.1**	**317.6**	**299.7**	**288.8**
StarCraft	h^1_φ	LTL-LRTA*	13580.0	8998.9	7900.9	**5301.7**	4821.7	**2973.2**
		LTL-LRTA*$_A$	**11483.5**	**8184.0**	**7147.3**	5506.0	**3437.8**	2986.3
	h^M_φ	LTL-LRTA*	5098.9	2947.9	4317.9	**1230.9**	947.7	597.9
		LTL-LRTA*$_A$	**5041.7**	**2681.5**	**4276.0**	1263.4	**889.0**	**585.4**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	39845.1	23824.1	13818.5	7749.7	3840.8	2462.0
		LTL-LRTA*$_A$	**4508.4**	**2788.6**	**1923.7**	**1181.9**	**910.1**	**691.7**
Maze	h^1_φ	LTL-LRTA*	10788.9	11096.7	7220.2	4867.8	2902.0	**1971.5**
		LTL-LRTA*$_A$	**9433.8**	**8336.8**	**7030.8**	**4180.9**	**2899.2**	2044.8
	h^M_φ	LTL-LRTA*	**13072.0**	7705.6	4418.1	**2396.9**	**1459.7**	**802.6**
		LTL-LRTA*$_A$	13294.7	**7505.9**	**4130.8**	2473.7	1546.6	804.7
	$\tilde{h}^M_\varphi$	LTL-LRTA*	30067.2	15544.9	9748.2	5110.1	3178.8	1979.8
		LTL-LRTA*$_A$	**11650.1**	**6604.5**	**3611.1**	**2144.2**	**1376.1**	**891.1**

Table 11: Result with 3 instances of each letter, and goal #4

Configuration			Lookahead					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h_φ^1	LTL-LRTA*	**69945.2**	50583.6	40210.9	32338.5	**18250.0**	14929.2
		LTL-LRTA*$_A$	74833.6	**45047.0**	**38642.8**	**25385.7**	18353.5	**11146.4**
	h_φ^M	LTL-LRTA*	1189.1	**940.9**	871.9	811.8	785.5	758.0
		LTL-LRTA*$_A$	**1179.1**	960.3	**869.8**	**803.1**	**770.9**	**741.6**
	$\tilde{h}_\varphi^M$	LTL-LRTA*	606811.8	314147.6	172195.4	94910.5	49204.6	25341.5
		LTL-LRTA*$_A$	**41051.3**	**22280.1**	**13585.4**	**7121.2**	**4159.3**	**3206.3**
StarCraft	h_φ^1	LTL-LRTA*	**115810.9**	102790.3	67109.8	54826.0	39585.0	26494.2
		LTL-LRTA*$_A$	119037.3	**94405.9**	**64839.6**	**48872.7**	**35379.3**	**22605.0**
	h_φ^M	LTL-LRTA*	**100798.8**	**59577.0**	**36949.2**	**19625.7**	12255.5	**6748.3**
		LTL-LRTA*$_A$	100824.0	59676.1	36969.1	19935.6	**11690.6**	6838.4
	$\tilde{h}_\varphi^M$	LTL-LRTA*	774117.4	439254.2	254813.5	129631.9	67025.2	37296.3
		LTL-LRTA*$_A$	**176298.6**	**100431.6**	**53548.9**	**32809.7**	**17754.2**	**10233.4**
Maze	h_φ^1	LTL-LRTA*	146562.6	**123481.3**	93773.0	**53744.1**	44519.0	31583.1
		LTL-LRTA*$_A$	**133118.3**	124783.7	**87015.4**	56317.5	**41294.0**	**24171.3**
	h_φ^M	LTL-LRTA*	493926.7	259875.8	**152246.1**	84821.9	**45216.6**	25430.5
		LTL-LRTA*$_A$	**492961.4**	**258093.2**	152256.3	**83542.6**	45661.9	**24933.1**
	$\tilde{h}_\varphi^M$	LTL-LRTA*	878868.4	522764.9	275791.2	149711.2	81274.0	38023.1
		LTL-LRTA*$_A$	**519341.9**	**294669.5**	**167271.9**	**84662.0**	**45599.7**	**26264.8**

Table 12: Result with 25 instances of each letter, and goal #4

Configuration			Lookahead					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h_φ^1	LTL-LRTA*	10477.6	**5591.6**	4838.2	3910.7	2794.3	2103.3
		LTL-LRTA*$_A$	**9162.2**	5717.8	**4451.0**	**2638.3**	**2079.6**	**1267.7**
	h_φ^M	LTL-LRTA*	**353.6**	**297.5**	291.0	**270.1**	270.5	255.9
		LTL-LRTA*$_A$	355.9	300.4	**278.3**	273.9	**248.7**	**240.0**
	$\tilde{h}_\varphi^M$	LTL-LRTA*	16383.6	8515.4	5229.9	2714.1	1669.9	849.0
		LTL-LRTA*$_A$	**1510.7**	**741.7**	**549.7**	**463.7**	**356.6**	**284.7**
StarCraft	h_φ^1	LTL-LRTA*	**10884.6**	10263.2	8104.4	4949.6	3799.8	2673.6
		LTL-LRTA*$_A$	11229.7	**7891.5**	**5627.8**	**3850.5**	**3132.4**	**2110.6**
	h_φ^M	LTL-LRTA*	4076.3	2305.0	**1774.3**	**976.8**	723.2	545.8
		LTL-LRTA*$_A$	4076.3	**2304.8**	1775.7	1036.7	**714.1**	**528.2**
	$\tilde{h}_\varphi^M$	LTL-LRTA*	43019.3	22938.0	13444.4	7255.6	4224.3	2400.4
		LTL-LRTA*$_A$	**4845.2**	**3118.2**	**1827.1**	**1383.8**	**776.2**	**561.8**
Maze	h_φ^1	LTL-LRTA*	10792.5	7570.2	5727.9	4592.7	3009.5	1998.7
		LTL-LRTA*$_A$	**8464.9**	**7191.9**	**5051.6**	**3529.4**	**2676.1**	**1936.1**
	h_φ^M	LTL-LRTA*	14767.6	9012.4	**4344.9**	2785.3	**1704.1**	**1159.6**
		LTL-LRTA*$_A$	**14734.0**	**8868.6**	4368.5	**2588.8**	1721.3	1171.9
	$\tilde{h}_\varphi^M$	LTL-LRTA*	29261.5	17394.8	9175.9	5203.3	2814.9	1826.2
		LTL-LRTA*$_A$	**11407.4**	**7558.7**	**4330.7**	**2498.0**	**1402.5**	**825.5**

Table 13: Result with 3 instances of each letter, and goal #5

Configuration			Lookahead					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h^1_φ	LTL-LRTA*	**86526.3**	56834.3	44982.4	33981.7	24079.5	15364.3
		LTL-LRTA*$_A$	87268.2	**52820.8**	**38532.2**	**26194.0**	**18571.4**	**11617.3**
	h^M_φ	LTL-LRTA*	1317.9	**1087.6**	970.9	907.9	873.3	823.8
		LTL-LRTA*$_A$	**1309.1**	1102.4	**953.2**	**897.9**	**847.7**	**813.3**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	834933.6	435398.0	240934.4	131211.3	70923.3	38190.6
		LTL-LRTA*$_A$	**1312.5**	**1106.4**	**958.4**	**898.2**	**844.9**	**809.1**
StarCraft	h^1_φ	LTL-LRTA*	161634.5	97391.3	86769.2	58936.8	41606.4	27173.2
		LTL-LRTA*$_A$	**143611.3**	**88604.5**	**83672.5**	**55414.5**	**39251.7**	**26728.3**
	h^M_φ	LTL-LRTA*	154983.7	**81677.8**	45826.4	26746.3	**14199.0**	**7732.3**
		LTL-LRTA*$_A$	**153994.8**	81748.4	**45720.9**	**26735.3**	14308.1	8661.7
	$\tilde{h}^M_\varphi$	LTL-LRTA*	1446754.4	772730.1	431612.3	227322.8	125553.0	63897.5
		LTL-LRTA*$_A$	**119778.2**	**80925.1**	**45424.7**	**23822.0**	**14817.3**	**8055.4**
Maze	h^1_φ	LTL-LRTA*	**168732.6**	112834.4	**91820.2**	66571.6	**40468.7**	31466.7
		LTL-LRTA*$_A$	187808.7	**100055.9**	91833.2	**58946.4**	43833.3	**25115.7**
	h^M_φ	LTL-LRTA*	442282.6	**252033.7**	141194.1	**72518.2**	**39141.7**	22149.5
		LTL-LRTA*$_A$	**442120.4**	253010.4	**140731.8**	73289.3	39175.0	**21987.4**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	884234.6	498259.0	274502.3	153179.0	84003.2	38962.8
		LTL-LRTA*$_A$	**496378.1**	**274925.6**	**147534.0**	**86417.9**	**44594.5**	**24112.1**

Table 14: Result with 25 instances of each letter, and goal #5

Configuration			Lookahead					
Domain	Heuristic	Algorithm	32	64	128	256	512	1024
Room	h^1_φ	LTL-LRTA*	12236.9	6807.6	5884.5	3874.7	3088.4	2536.6
		LTL-LRTA*$_A$	**10532.3**	**6013.2**	**4590.9**	**2791.9**	**1888.6**	**1382.9**
	h^M_φ	LTL-LRTA*	381.3	**318.5**	**295.5**	276.8	268.8	264.9
		LTL-LRTA*$_A$	**377.0**	322.3	299.0	**270.9**	**257.2**	**244.8**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	23228.3	13833.4	7170.4	4345.2	2385.3	1691.3
		LTL-LRTA*$_A$	**410.0**	**355.0**	**328.6**	**299.9**	**284.5**	**265.6**
StarCraft	h^1_φ	LTL-LRTA*	12414.9	10887.7	7257.5	6063.1	3843.8	2636.4
		LTL-LRTA*$_A$	**12365.4**	**8820.0**	**6566.1**	**5188.6**	**3656.2**	**2082.9**
	h^M_φ	LTL-LRTA*	4531.1	**2798.6**	**1984.7**	**1322.0**	809.7	570.6
		LTL-LRTA*$_A$	**4526.5**	2823.7	2022.0	1322.3	**789.0**	**568.6**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	41233.4	23727.6	13823.8	7899.6	4830.6	2357.7
		LTL-LRTA*$_A$	**4945.4**	**2736.3**	**1908.5**	**1393.4**	**797.2**	**493.7**
Maze	h^1_φ	LTL-LRTA*	11980.2	7855.2	6428.9	4728.9	3345.7	2382.3
		LTL-LRTA*$_A$	**10586.6**	**7823.4**	**5497.5**	**3884.2**	**3066.4**	**1812.1**
	h^M_φ	LTL-LRTA*	15995.4	10208.1	**5669.8**	**3300.5**	1877.0	1191.7
		LTL-LRTA*$_A$	**15605.8**	**10150.9**	5797.0	3416.7	**1852.4**	**1051.5**
	$\tilde{h}^M_\varphi$	LTL-LRTA*	39321.0	21720.0	13228.7	7640.9	3460.2	2230.6
		LTL-LRTA*$_A$	**12452.9**	**7082.5**	**4468.5**	**2896.1**	**1479.5**	**1077.0**