A Monte-Carlo Policy Rollout Planner for Pathfinding in Real-Time Strategy (RTS) Games

Munir Naveed, Diane Kitchin and Andrew Crampton
Department of Informatics
University of Huddersfield, England, UK.
Emails: \{m.naveed, d.kitchin, a.crampton\}@hud.ac.uk

Abstract
In this paper, we present a novel Monte-Carlo policy rollout algorithm (called MCRT-GAP) that uses a hybrid of focussed exploration of new actions and exploitation of promising actions to expand the look-ahead search from the current state in a Monte-Carlo simulation model. The initial estimates of the action values are computed using a combination of the distance heuristic and a collision estimate. To balance the exploration of new actions and the exploitation of already explored promising actions, MCRT-GAP uses a greedy action selection approach to exploit the best action and a random selection approach to explore new actions within a small set of useful actions (i.e. smaller than the size of the action set in a domain world). This small set is called a corridor. In this paper, we describe the motivation and the algorithmic details of MCRT-GAP. The paper also presents the theoretical properties of the algorithm.

INTRODUCTION
Monte-Carlo Policy Rollout algorithms (Bertsekas, Tsitsiklis, and Wu 1997) (Tesauro and Galperin 1996) (Kocsis and Szepesvári 2006) approximate action values using Monte-Carlo simulations. Monte-Carlo simulations require a model that can simulate the effects of an action using the state transition function and assign a reward for the state-action pair. These algorithms run several rollouts (or iterations from the current state) to approximate the values of the action applicable at the current state and select the action that has the highest value compared with other actions applicable at that state. These algorithms have been successfully applied to solve planning problems in deterministic domains (e.g. in Go (Lee et al. 2009)) and non-deterministic domains (e.g. Solitaire (Bjarnason, Fern, and Tadepalli 2009)). These algorithms are suitable for online planning in domains with large state and action spaces, for example, real-time strategy (RTS) games. The domain world of a RTS game not only has a huge state space but it also imposes tight time constraints on the planning time. Furthermore, the state-space in a RTS game can be changed during the game play. Path planning is one of the most challenging tasks in a RTS game because it is required by all movable characters in a game. Many pathfinding approaches e.g. A* (Nilsson 1982), Navigational Mesh (Hamm 2008) and anytime algorithms (Likhachev, Gordon, and Thrun 2004) are not suitable due to the challenging issues (time constraints, dynamic changes and large state space) in a RTS game world.

In this paper, we describe a novel Monte-Carlo policy rollout algorithm that uses a simple and real-time Monte-Carlo simulation model. The work is an extension of MCRT (Naveed et al. 2011) - a Monte-Carlo simulation based path planner. MCRT is based on a sparse sampling scheme while MCRT-GAP is a pure policy rollout algorithm with one stage. The main contribution of MCRT-GAP is a heuristic based exploration scheme. The exploration of new actions in a Monte-Carlo simulation is kept focussed in a small space around the best action. The focussed region (also called a corridor) at a state is changed if the simulation model finds a better action than the current best action at that state. The new algorithm is evaluated using the benchmark problems of one map.

NOTATION
We are interested in solving path planning problems in a domain world that is partially visible, real-time and dynamic (like a typical RTS game). The planning agent knows about its current location and the goal location. The planning agent can see only a limited part of the domain world near its current position. The visible part is a circular region of radius of ten states with the current state of the planning agent at the centre of the visibility circle. It is assumed that the states that are beyond the visibility range are not occupied by static obstacles. In this section, we formalise the planning problem (that we are interested in).

Definition 1
The domain world has a finite set of states $S$ and a finite set of actions $A$.

Definition 2
For each state $s \in S$, we define $A(s)$ to be the set of applicable actions at $s$ where $A(s) \subseteq A$.

Definition 3
$Next(s, a)$ is a set of states that are possible to reach from $s$ by taking action $a \in A(s)$.

Definition 4
$Transition(s, a)$ is a stochastic transition function that selects a next state $s' \in Next(s, a)$ for a state-action pair $(s, a)$ where $s \in S$ and $a \in A(s)$.

Definition 5
$R(s, a)$ is a reward function ($R : S \times A \rightarrow \mathbb{R}$) that assigns a value to a state action pair $(s, a)$ in a simulation model.
Definition 6
A simulation model is a part of MCRT-GAP that uses the function *Transition* to randomly select the next state $s'$ of a state-action pair $(s, a)$ for all $s \in S$ and $a \in A(s)$ and then uses $R$ to assign a reward value for the pair. The simulation model uses a probability distribution $P$ to prioritise the selection of the most likely transition. $P : S \times A \times S \rightarrow [0, 1]$ is a function that gives the likelihood of reaching a next state $s' \in Next(s, a)$ if an action $a \in A(s)$ is applied at state $s$. For example, if $s_1, s_2$ and $s_3$ are the possible next states of the current state $s_c$ with action $a \in A(s_c)$ then $Next(s_c, a) = \{s_1, s_2, s_3\}$. Suppose the transition probabilities are $P(s_c, a, s_1) = 0.2$, $P(s_c, a, s_2) = 0.6$, $P(s_c, a, s_3) = 0.2$. This means it is most likely that the planning agent would reach $s_2$ if $a$ is applied at $s_c$. If $P$ is not available in a domain, MCRT-GAP updates the probabilities using the online interaction with the environment. The probability of moving to an occupied state $s_0$ from $s$ with any action $a \in A(s)$ is always zero i.e. $P(s, a, s_0) = 0$.

Definition 7
$V(s)$ is the state value and $Q(s, a)$ is the estimated value for action $a \in A(s)$ at state $s$. These values are computed using the simulation model of MCRT-GAP.

Definition 8
The best action $a_b \in A(s)$ at $s$ has the highest estimated value at $s$, it is computed using equation (1) (adapted from (Tesauro and Galperin 1996)).

$$a_b = \arg \max_a Q(s, a)$$

MCRT-GAP
MCRT-GAP expands the look-ahead search from the current state $s$ to a fixed depth $D$ in a simulation. At $s$, it chooses the best action $a_c \in A(s)$ as a sample. It explores the actions at the successor state $s' = T(s, a_c)$ of $s$ seen in the look-ahead search. At $s'$, it selects an action $a' \in A(s')$ from a small list of action $A_m(s') \subseteq A(s')$ such that $a' \in A_c(s')$. The small list of actions is called a corridor. $A_m(s')$ is constructed using the best action $a$ at $s'$. This sampling continues until the look-ahead search reaches depth $D$. For each state-action pair seen during the look-ahead search, MCRT-GAP keeps computing the reward values using $R$ function. The state seen at depth $D$ is evaluated using an admissible heuristic and this value is added to the sum of the rewards of all state-action pairs seen in the look-ahead search in a simulation. This accumulated value is used to update the expected long term reward of the action sampled at the current state i.e. $Q(s, a_s)$. If the returned value is bigger than the current $Q(s, a_s)$ then the estimated action value is modified otherwise it remains unchanged. If $Q(s, a_s)$ remains unchanged for $n_{lim} > 0$ consecutive simulations, then MCRT-GAP selects the best action at $s$ by excluding $a_s$ from the action list. If $s$ is converged or if the time to run the simulations expires, it selects the best action at $s$ as a plan and returns $a$ for execution.

OBJECTIVE
In Monte-Carlo simulations, the exploration of new actions is an important and essential part of the search. It is important because the action values at the current state of the planning agent are estimated using the local information. However, the exploration of new actions that are not useful to solve the current planning problem is computationally very expensive for the planning search algorithm. It is intuitive to limit the exploration of the search space - during the Monte-Carlo simulations - within the vicinity (or corridor) of the effect(s) of the current best action of a state. The exploration scheme also gives importance to the actions that are less explored by increasing their chance of selection. The chance of selection is computed using $1/n(s, a)$ where $n(s, a)$ is the number of times an action $a \in A(s)$ is sampled at state $s$. The corridors are kept overlapping i.e. two corridors (each one by a different action) can have one or more common actions, so the exploration of the new actions can move from one corridor to another one in two consecutive simulations. At current state $s$, the exploration of the new action is performed if the estimated value of the best action $a_b \in A(s)$ does not change for some iterations.

ALGORITHMIC DETAILS
MCRT-GAP makes two main changes to the original MCRT (Naveed et al. 2011). First it uses the estimated action values to draw a sample at the current state of the planning agent. Second, it uses a corridor based exploration scheme to draw the action samples at the successor states of the current state. To explore the new actions, MCRT-GAP uses a small set of the actions that are relevant to the best action in the current state. MCRT-GAP performs exploration of the new actions within that corridor. The construction of the corridor for an action is done automatically. This can be done by using the angle between the directional lines of two actions if the actions are directional. A corridor for an action is built before the start of planning and is stored in memory for online use. The construction of a corridor for an action is a trivial process. We use angle between the directions of two actions to build a corridor. For example, a corridor of an action "MOVE TO NORTH" is a set "MOVE TO NORTH WEST", "MOVE TO NORTH", "MOVE TO NORTH EAST" where each member of the set has an angle of 45° or less with action "MOVE TO NORTH". The general overview of MCRT-GAP is given in Figure 1. At the current state $s_c$, if $s_c$ is converged then MCRT-GAP returns the best action at $s_c$ (Line 2, Figure 1). The convergence of a state $s_c$ is discussed later in this paper. If $s_c$ is not converged yet, then MCRT-GAP runs several rollouts depending on the time limit to estimate the action values at $s_c$. In each rollout, MCRT-GAP chooses (Line 6, Figure 1) the best action $a \in A(s_c)$ at $s_c$ as a sample. If $s_c$ is seen for the first time, or any of its actions are not sampled yet, then the action is selected randomly (from the unseen actions). The next state $s_n$ of $s_c$ with sampled action $a$ is estimated by using the stochastic state transition function $Transition(s, a)$ (Line 7, Figure 2). The immediate reward $R(s_c, a)$ of the state-action pair is computed and stored in $r_n$ (Line 8, Figure 2). $s_n$ is expanded for a length of $depth - 1$
using a combination of exploration of new actions and exploitation of the previously seen actions. \( s_n \) also chooses the best action if each applicable action \( a \in A(s_n) \) has been sampled at least once. If the best action \( a' \) at \( s_n \) has been sampled in the previous searches then it selects an action \( a \) randomly from the corridor of \( a' \). The chances of selection of an action in the random selection in the corridor depends on the number of times the action is sampled in the previous searches. An action \( a \in A(Corridor(a')) \) in the corridor of \( a' \) has more chances of selection as a sample than other members of \( Corridor(a') \) if \( a \) is the least explored. The immediate reward of the state action pair \((s_n, a)\) is computed and added to \( r_n \) (Line 15).

end of the simulations, MCRT-GAP selects the best action at \( s_n \) and returns it for execution. The details of the immediate reward function \( R(s_n, a) \) are given in Figure 2. This reward function has been used in the previous work (Naveed et al. 2011) and (Naveed, Crampton, and Kitchin 2010).

Function \( R(s_n, a) \)

**Read access** \( MDP, g \):
1. \( s_{next} := Transition(s_n, a) \);
2. \( rw := \frac{\sum_{i:t(p(s_n,a,s_n))>0 \forall s_n \in Next(s_n,a))}{dist(s_{next}, g)} \);
3. RETURN \( rw \)

End \( R \)

**Figure 2**: Reward Function

\( R \) (Figure 2) computes the reward of a state-action pair seen during Monte-Carlo simulations. The reward is computed using the current goal \( g \) of the planning problem. \( R \) estimates the transition (i.e. next state \( s_{next} \)) of the given state-action pair (Line 1, 2) and estimates the distance between \( s_{next} \) and goal state \( g \) using a heuristic function \( dist \). \( R \) uses the list of all possible states (i.e. \( Next(s_n,a) \)) reachable directly from the state-action pair \((s_n, a)\). The number of states in \( Next(s_n,a) \) which are unoccupied divided by the distance heuristic \( dist(s_{next}, g) \) is the immediate reward of \((s_n, a)\) (Line 2). A state \( s_i \in Next(s_n,a) \) is unoccupied if the probability of reaching \( s_i \) from \( s_n \) with action \( a \in A(s_n) \) is greater than zero.

**COMPLEXITY ANALYSIS**

MCRT-GAP selects an action from current state \( s_c \) using \( ChooseAction \) (Lines 6, Figure 1). It takes \( O(1) \) to select an action randomly at \( s_c \) if \( s_c \) is seen first time. If all actions at \( s_c \) have been sampled in the previous searching efforts, then it takes \( O(|A(s_c)|) \) to select the best action at \( s_c \) (Line 1, Figure 1). The average time complexity of \( ChooseAction \) is \( O(1) \) and the worst case time complexity is \( O(|A|) \). The worst case time complexity of \( ChooseActionFromCorridor \) (Line 14, Figure 1) is \( O(C) \) where \( C \) is the size of the corridor. The space complexity of MCRT-GAP per simulation is \( O(D + |A|) \); it has the worst case time complexity of \( O(D|A|C) \) and the average time complexity per simulation is \( O(DC) \).

**CONVERGENCE AND OPTIMALITY**

From Figure 2 (Line 22.), it is obvious that \( Q(s,a) \) for any state \( s \in S \) and action \( a \in A(s) \) remains the same or increases with the increase in the number of rollouts. At any simulation time \( t > 1 \), the \( Q_t(s,a) \) \( \forall s \in S, a \in A(s) \) is monotonically non-decreasing as shown in Equation 2.

\[
0 < Q_{t-1}(s,a) \leq Q_t(s,a), \quad \forall t > 1
\]

MCRT-GAP also preserves the monotonicity of the state value \( V(s) \) for each state \( s \in S \). The optimal state value \( V^*(s) \) for state \( s \in S \) is the supremum of the monotonic sequence \( V(s) \) and the optimal action value \( Q^*(s,a) \) is \( sup(Q(s,a)) \) for \( a \in A(s) \). In other words, the optimal
action value of a state action pair \((s, a)\) is an upper bound on the monotonic sequence of \(Q_t(s, a)\) for all \(t > 1\) (Equation (2)).

**Lemma 1.** A monotonic function (of real numbers) converges if it is bounded. (Copson 1970)

**THEOREM 1.** MCRT-GAP eventually finds the optimal action value \(Q(s, a)\) for a given state \(s\) and action \(a \in A(s)\) if repeated for several iterations.

**Proof:** The proof follows Lemma 1. MCRT-GAP generates a monotonic sequence \(Q(s, a)\) of action values for the state action pair \((s, a)\) by running several simulations from \(s \in S\). It is given that the optimal value \(Q^*(s, a)\) is an upper bound of monotonic sequence \(Q(s, a)\). Therefore, MCRT-GAP eventually converges to the optimal action value for a given state action pair \((s, a)\).

It is not clear how many iterations MCRT-GAP requires to converge the action value function \(Q(s, a)\) to an optimal value for a given state \(s\) and action \(a\). For practical reasons, we use an error bound based on the algorithmic parameter \(n_{lim} > 0\) to define the convergence of the action value function \(Q\). If MCRT-GAP does not change the action value of a state-action pair \((s, a)\) for \(n_{lim}\) consecutive simulations, then it is assumed that \(Q(s, a)\) has converged relative to parameter \(n_{lim}\). If all applicable actions \(a \in A(s)\) at \(s\) are converged with respect to \(n_{lim}\) then \(s\) is declared a converged state relative to \(n_{lim}\).

**PATH PLANNER**

MCRT-GAP is embedded in a complete real-time planner. The real-time planner interleaves planning and plan execution. In each planning episode, MCRT-GAP plans an action at the current state of the planning agent and the action is returned for execution. After execution, the planning agents move to a new state. At the new state, the planner selects an action using MCRT-GAP, executes it and moves to another state. This process continues until the planning agent reaches the goal state. A high level design of the planner is given in Figure 3.

**Procedure Planner**

Read \((s_0, g)\):
1. initialises parameters of MCRT-GAP and state \(s := s_0\);
2. REPEAT
3. \(a := MCRT-GAP(s, g)\);
4. \(s := Execute(a, s)\)
5. \(UpdateP\)
6. UNTIL \(s\).pos = \(g\);

**End Planner**

Figure 3: A Real-Time Planner

The planner reads a planning problem with the initial state \(s_0\) of the planning agent and the goal state \(g\) it tends to move to. The planner initialises the MCRT-GAP parameters e.g. \(timelimit, n_{lim}\) and look-ahead depth \(depth\) (Line 1, Figure 3). The planner calls MCRT-GAP for the initial state \(s_0\) to select an action \(a\) to move towards \(g\) (Line 3). The action selected by MCRT-GAP is executed at \(s\) and the agent moves to a new state \(s\) (Line 4). The state transition probabilities are updated for the transition (Line 5). If \(s\) is a goal state then the planner stops otherwise it keeps planning and executing until the planning agent finds the goal state.

**RELATED WORK**

Tesauro (Tesauro and Galperin 1996) explored the policy rollout algorithm in a stochastic board game called Backgammon. The simulation model in Tesauro’s work takes several iterations per move to decide an action at the current move. This approach is expensive for a real-time application. Kearns et al. (1999) present a spare sampling based approach that generates a look-ahead tree of fixed depth \(H\) but each action applicable at state (seen during the look-ahead search) is sampled \(C\) times in a simulation. The number of samples in a simulation are exponential in \(H\). To avoid exploring all actions in a sparse sampling scheme, Auer et al. (2002) demonstrate an adaptive action sampling approach (called Upper Confidence Bounds or UCB) that selects only one action per state in the look-ahead search in a simulation. It selects the best action as a sample at a state.

To balance the exploration of new actions and exploitation of the best action, Auer et al. present upper bounds on the selection of an action as a sample at a state. However, Auer et al.’s sampling approach continues the exploration of new actions forever. Kocsis and Szepesvári (2006) present a variation of UCB, called Upper Confidence Bounds applied to Trees (UCT), that performs selective action sampling in a policy rollout fashion. The default rollout policy is random. UCT has been successful in Go (Lee et al. 2009) and Solitaire (Bjarnason, Fern, and Tadepalli 2009) games. However, UCT and UCB are applicable in a domain if the action values are in the range \([0,1]\). In RTS games, action values are beyond this range. Balla et al. (Balla and Fern 2009) present a variation of UCT in a RTS game to solve the tactical assault problem. The variation of UCT uses a reward function that can optimise one parameter (time or health factor) in a simulation model.

MCRT-GAP performs exploration of the new actions using the heuristic function. It avoids the exploration of the actions that are not useful according to the current estimates. It exploits the knowledge it discovers in the previous efforts to draw samples in the look-ahead search. It estimates the action values for a given state under the tight real-time constraints. MCRT-GAP also shares some characteristics with the recent real-time path planners like LSS-LRTS (Koenig and Sun 2009) and Real-time D* Lite (RTD) (Bond et al. 2010). RTD and LSS-LRTA interleave planning and plan execution like MCRT-GAP. LSS-LRTA only updates the cost values of the actions if they are increasing. It does not decrease the cost of an action if it is reduced due to a dynamic change in a domain world.

**EXPERIMENTAL DETAILS**

MCRT-GAP is empirically evaluated using a benchmark map called Arena2. Arena2 (Figure 4) has \(281 \times 209\) states. We use the MAI tool to implement MCRT-GAP and the
planner. The tool has been used in a previous study (Bond et al. 2010). It provides an easy to use programming environment for the exploration of real-time path planning in a partially visible and dynamic world. The tool also supports STRIPS planning. For the benchmark map, the tool reads the first 900 planning problems and passes them to the planner sequentially. The maximum path length in this problem set is 359.397. The planner reads a planning problem and solves it. The tool imposes a limit on the solution time. If the planner does not reach the goal state in the allowable time, then the problem is declared unsolvable by the planner.

IMPLEMENTATION DETAILS

To load the benchmark map in the MAI tool, we use HOG-GridWorld library. The MAI tool encodes actions as a navigational Compass. The action set is MOVE TO NORTH, MOVE TO EAST, MOVE TO SOUTH, MOVE TO WEST, MOVE TO NORTH EAST, MOVE TO SOUTH EAST, MOVE TO EAST, MOVE TO SOUTH, MOVE TO WEST, and MOVE TO NORTH WEST. A state is a \((x, y)\) location on the map. \(n(s, a)\) for each sampled action \(a\) at state \(s\), action value \(Q(s, a)\) and the transitions probabilities \(P\) are stored in the hash tables. \(n(s, a)\), \(P\) and \(Q\) are empty at the start of the planning and are populated during the planning episodes. \(P\) is updated after a state transition occurs (Line 5, Figure 3) or when a state occupied by an static obstacle is seen. if a state-transition has an entry in \(P\) then it is accessed through the key otherwise MCRT-GAP uses the default value. The default value for a state transition probability is 0.000001. If a state-action \((s, a)\) does not have an entry in \(Q\) table, then MCRT-GAP assumes that \(a\) is not sampled at \(s\). Every sampled action \(a\) at any state \(s \in S\) has an entry in \(Q\). The states that are not seen during the planning search, they are not added to \(Q\).

PERFORMANCE MEASUREMENT

The performance of MCRT-GAP is measured using two parameters: time to solve 900 planning problems and sub-optimality of the solution by the planner. The sub-optimality is measured using a ration of \(l_o\) to \(l_a\) where \(l_o\) is the length of the solution by the planner and \(l_a\) is the length of the optimal path given in the benchmark. The higher values of these two parameters represent poor performance of the planner.

RESULTS

MCRT-GAP Planner is run on seventeen different machines of same hardware and software configurations. Each machine has Intel(R) Core (TM) 2 Quad processors each of speed 2.6 GHz CPU speed and 8 GB ram. MCRT-GAP planner is run for different \(\text{timelimit}\) and \(\text{depth}\) values. The visibility of the planning agent is kept the same in all experiments. The experimental results on Arena2 are shown in Table 1. These results are the average of five runs. There are twenty five problems in first 900 planning problems that are not included in the set because MCRT could not solve them under the given time limit. The initial results show that the look-ahead depth plays an important role in reducing the sub-optimality of the planning algorithm, however, it also increases the time to solve the planning problems. Increasing \(\text{timelimit}\) can reduce the time to solve the planning problems. This is due to the convergence of the states with respect to \(n_{\text{limit}}\). \(n_{\text{limit}} = 1\) means if the action value of a state-action pair does not change between two consecutive simulations, then this action is not explored in the future search efforts. With small look-ahead depth e.g. 3, the states are converged quickly relative to \(n_{\text{limit}} = 1\) and at a converged state MCRT-GAP selects the best action using the previously computed estimates, therefore, it takes a small duration to solve a planning problem but at the cost of optimality. The sub-optimality is highest at the shallow look-ahead depth with a higher simulation time.

CONCLUSION AND FUTURE WORK

We present a work in progress in this paper. The paper presents a new policy rollout algorithm, called MCRT-GAP, that learns the action values online using a simulation model. MCRT-GAP explores new actions in a focussed part of the look-ahead search to avoid exploration of the actions that are not useful. The paper describes the algorithmic details and the theoretical details of the algorithm. The results of the initial experiments are described on a benchmark map Arena2. The initial results show that the performance of MCRT-GAP depends on look-ahead depth and convergence parameter \(n_{\text{limit}}\). The quality of solution by MCRT-GAP is poor in the initial results. However, the results also indicate that the higher values of \(n_{\text{limit}}\) and \(\text{timelimit}\) can be useful to improve performance of the algorithm.

<table>
<thead>
<tr>
<th>Timelimit</th>
<th>Depth</th>
<th>Avg Time (sec)</th>
<th>Avg Subopt</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>3</td>
<td>87990.20</td>
<td>10.66</td>
</tr>
<tr>
<td>10</td>
<td>9</td>
<td>137728.53</td>
<td>9.25</td>
</tr>
<tr>
<td>10</td>
<td>12</td>
<td>200145.47</td>
<td>8.72</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>208809.61</td>
<td>8.99</td>
</tr>
<tr>
<td>30</td>
<td>3</td>
<td>1136.68</td>
<td>15.64</td>
</tr>
<tr>
<td>30</td>
<td>12</td>
<td>64843.06</td>
<td>8.54</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>65091.56</td>
<td>9.0</td>
</tr>
</tbody>
</table>

Table 1: Time and Suboptimality (Subopt) of MCRT-GAP planner (with \(n_{\text{limit}} = 1\)) on Arena2.
In future work, we are aiming to extend the experimental work for a detailed analysis of the relationship between depth, timelimit and nlimit in MCRT-GAP. In future experiments, we plan to use six more maps from the benchmark problems. These maps are Den401d, Lake303d, Orz103d, Orz701d, Orz702d and Orz900d. These maps vary in size and the number of obstacles (or blocked states). The results of MCRT-GAP will be compared against RTD, LSS-LRTA and MCRT.

References


Likhachev, M.; Gordon, G.; and Thrun, S. 2004. Ara*: Anytime a* with provable bounds on sub-optimality. In *IN ADVANCES IN NEURAL INFORMATION PROCESSING SYS-