Exponential Deepening A* for Real-Time Agent-Centered Search
(Extended Abstract), full version has been accepted to AAAI-2014

Guni Sharon
ISE Department
Ben-Gurion University
Israel
gunisharon@gmail.com

Ariel Felner
ISE Department
Ben-Gurion University
Israel
felner@bgu.ac.il

Nathan R. Sturtevant
Department of Computer Science
University of Denver
USA
sturtevant@cs.du.edu

Abstract
This paper introduces Exponential Deepening A* (EDA*), an Iterative Deepening (ID) algorithm where the threshold between successive Depth-First calls is increased exponentially. EDA* can be viewed as a Real-Time Agent-Centered (RTACS) algorithm. Unlike most existing RTACS algorithms, EDA* is proven to hold a worst case bound that is linear in the state space. Experimental results demonstrate up to 5x reduction over existing RTACS solvers wrt distance traveled, states expanded and CPU runtime. Full version of this paper appears in AAAI-14.

Real-Time Agent-Centered Search
In the real-time agent-centered search (RTACS) problem, an agent is located in start and its task is to physically arrive at the goal. RTACS algorithms perform cycles that include a planning phase where search occurs and an acting phase where the agent physically moves. Several plan-act cycles are performed under the following restrictive assumptions:
Assumption 1: As a real-time problem, the agent can only perform a constant-bounded number of computations before it must act by following an edge from its current state. Then, a new plan-act cycle begins from its new position.
Assumption 2: The internal memory of the agent is limited. But, agents are allowed to write a small (constant) amount of information into each state (e.g., g- and h- values). In this way RTACS solvers are an example of ‘ant’ algorithms, with limited computation and memory (Shiloni et al. 2009).
Assumption 3: As an agent-centered problem, the agent is constrained to only manipulate (i.e., read and write information) states which are in close proximity to it; these are usually assumed to be contiguous around the agent.

Most existing RTACS solvers belong to the LRTA* family (Korf 1990; Koenig and Sun 2009; Hernández and Baier 2012). The core principles of this family is that when a state is visited by the agent, its heuristic value is updated through its neighbors. An RTACS agent has two types of state visits:
First visit - the current state was never visited previously by the agent. We denote the number of first visits by \( F \).
Revisit - the current state was visited previously by the agent. We denote the number of revisits by \( R \).

In areas where large heuristic errors exist, all LRTA* algorithms may revisit states many times, potentially linear in the state space per state (Koenig 1992). Thus, denoting the size of the state-space by \( N \), while \( F = O(N) \), \( R = O(N^2) \) in the worst case. Consequently, the total number of states visits \( (F + R) = O(N^2) \) is quadratic in \( N \).

We define an RTACS algorithm to be efficient if \( F = \theta(R) \). We break this into two conditions:
Condition 1 - \( R = O(F) \), meaning that the order of \( F \) is greater than or equal to \( R \).
Condition 2 - \( F = O(R) \), meaning that the order of \( R \) is greater than or equal to \( F \).

If \( R \gg F \) (condition 1 is violated) the agent spends most of the time revisiting previously seen (non-goal) states. Many existing RTACS algorithms (e.g., the LRTA* family) do not satisfy condition 1.

If \( F \gg R \) (condition 2 is violated) the agent might spend too much time in exploring new but irrelevant states.

Lemma 1 An algorithm has a worst case complexity linear in the size of the state space \( N \) if it satisfies condition 1.
Proof: Since in the worst case the entire state space will be visited, and each state can be visited for the first time only once, \( F = O(N) \). If condition 1 is satisfied then \( R = O(F) \). Now, since \( F = O(N) \) then \( R = O(N) \) too. Thus the complexity of the algorithm \( (F + R) \) is also \( O(N) \). On the other hand, if \( F + R = O(N) \) since \( F = O(N) \), \( R \) must also be \( O(N) \), so \( R = O(F) \) and condition 1 is satisfied. □

Exponential Deepening A*
To tackle the problem of extensive state revisiting we introduce Exponential Deepening A* (EDA*), a variant of IDA* (Korf 1985).

IDA* acts according to the high-level procedure presented in Algorithm 1. \( T \) denotes the threshold for a given Bounded DFS (BDFS) iteration where all states with \( f \leq T \) will be visited. In IDA*, \( T \) is initialized to \( h(\text{start}) \) (line 1). For the

Algorithm 1: IDA*/RIBS/EDA*

\[
\begin{align*}
\text{Input:} & \quad \text{Vertex} \ start, \ \text{Vertex} \ goal, \ \text{Int} \ C \\
1 & \quad T = start.h \\
2 & \quad \text{while } \text{BDFS}(\text{start}, \text{goal}, T) = \text{FALSE} \text{ do} \\
3 & \quad \quad \text{Case IDA*: } T = T + C \\
4 & \quad \quad \text{Case EDA*: } T = T \times C
\end{align*}
\]
next iteration, $T$ is incremented to the lowest $f$-value seen in the current iteration that is larger than $T$. For simplicity, we assume that $T$ is incremented by a constant $C$ (line 3). A lower bound for $C$ is the minimal edge cost.

Unlike IDA*, where the threshold for the next iteration grows linearly, in EDA* the threshold for the next iteration is multiplied by a constant factor ($C$) (line 4) and thus grows exponentially. To deal with the efficiency conditions for EDA*, we distinguish two types of domains.

1. **Exponential domains**: In exponential domains, the number of states at depth $d$ is $b^d$ (exponential in $d$) where $b$ is the branching factor. Assume that the depth of the goal is $d = C^i + 1$. In this case, the goal will not be found in iteration $i$, and will instead be found in iteration $i + 1$. All states with $f \leq C^{i+1}$ will be visited during the last iteration. There are $F = O(b(C^{i+1}))$ such states in total. In all previous iterations $R = \sum_{i=0}^{r} b(C^i) = O(b^{C+1})$ states will be visited. In exponential domains EDA* satisfies Condition 1, $R = O(F)$.

EDA*, however, violates Condition 2. Let $d$ be the optimal solution. We say that states with $f > d$ are surplus (Felner et al. 2012). Since the EDA* threshold may be increased beyond $d = C^i + 1$ up to $C^{i+1}$, the number of surplus nodes that EDA* will visit is $O(b^{C^i+1})$. This is exponentially more than the $b^{C^i+1}$ necessary nodes to verify the optimal solutions, i.e., those with $f \leq d$ (which are expanded by A*). Since $R = O(b^{C^i+1})$, $R < d$ and condition 2 is violated. Thus, EDA* is not efficient for exponential domains.

2. **Polynomial domains**: In polynomial domains the number of states at radius $r$ from the start state is $r^k$ where $k$ is the dimension of the domain. We assume that in a polynomial domain the number of unique states visited by EDA* within a threshold $T$ is $\theta(T^k)$. If the goal is found in iteration $i$, EDA* will visit $F = (C^i)^k = (C^k)^i = (\hat{C})^i$ states, where $\hat{C} = C^k$ is a constant. In all previous iterations the agent will visit $R = \sum_{i=0}^{r-1} (C^i)^k = \sum_{i=0}^{r-1} (\hat{C}^i) = \theta(\hat{C})$. Consequently, $F = \theta(R)$. EDA* satisfies both conditions 1 and 2. Since EDA* satisfies condition 1, its worst case complexity is linear in the state space, as proven in Lemma 1. Since it satisfies condition 2, the number of surplus nodes visited will not hurt the complexity. As a result, EDA* is considered fully efficient on polynomial domains.

**Experimental Results**

We experimented with the entire set of Dragon-Age: Origins (DAO) problems (all buckets, all instances) from (Sturtevant 2012). $h$ was set to octile distance. The algorithms used for this experiment were: IDA* (Korf 1985), LRITA*, RTA* (Korf 1990), daLRITA* and daRTA* (Hernández and Baier 2012), $f$-LRITA* (Sturtevant and Bulitko 2011), RIBS (Sturtevant et al. 2010) and EDA*. For EDA*, the number in parenthesis denotes the size of the constant factor $C$. $C$ was chosen from $\{1.1, 1.5, 2, 4, 8, 16\}$.

Table 1 reports the averages over all instances of three measures aspects: (1) The number of node expansions (expanded). (2) The total distance traveled during the solving process (Distance). (3) CPU runtime in ms. The CPU time spent in the planning phases (Time).

The best algorithm in each category is in bold. Different $C$ values for EDA* influence the performance. The value of $C = 8$ was best for all 4 measures. EDA* outperformed all other algorithms in all measures.

**Conclusions and Future Work**

This paper presents Exponential Deepening A*. EDA* is intuitive and very simple to implement. To the best of our knowledge EDA* is the only RTACS algorithm that is, in the worst case, linear in the state space. Experimental results on grids support our theoretical claims; EDA* outperforms other algorithms in all the measurements if standard lookahead of radius 1 is assumed. If deeper lookahead is allowed (not reported), EDA* is best in all measurements except Distance.

This research was supported by the Israel Science Foundation (ISF) under grant #417/13 to Ariel Felner.

**References**


