PARA*: Parallel Anytime Repairing A*

AAAI 2015 Submission X

Abstract

Recent advances in processor performance increasingly come in the form of additional processors as opposed to faster ones. In order for heuristic searches to take advantage of these architectures, we must adapt the algorithms for parallel processing. wPA*SE (weighted Parallel A* for Slow Expansions) is a recent parallel variant of A*, which guarantees a solution costing within a specified factor of optimal, while expanding each state at most once. wPA*SE can achieve a nearly linear speedup in the number of processor cores if expansions are sufficiently time-consuming to dominate the search runtime. Much of the overhead of wPA*SE is due to the careful selection of the next state to expand, which is needed to maintain the theoretical properties. In this work, we present Enhanced PA*SE (ePA*SE), which maintains speedups when expansions are faster than wPA*SE allows. ePA*SE reduces the overhead of wPA*SE in selecting states for expansion by maintaining tighter bounds on the suboptimality of each individual state. We show comparable performance to wPA*SE when expansions are slow, and better performance as the number of cores increases and expansions become faster. On the theoretical side, ePA*SE provides the same guarantees on completeness and solution quality as wPA*SE. We also show how it generalizes single-source shortest paths, providing performance bounds in the massively parallel limit. Finally, we present PARA*, an anytime variant of ePA*SE.

Introduction

Breadth-first and depth-first search are generalized by a class of frontier-based search algorithms, differing mainly in the means by which nodes are selected from the frontier for expansion. In the weighted A* algorithm, the choice combines a greedy goal-directed bias to reduce search time, with a breadth-first bias which guarantees suboptimality by a specified factor. With the advent of multi-core processors, making use of parallelism has become a priority for algorithm designers. Parallel A* for Slow Expansions (PA*SE) and its weighted variant wPA*SE offer nearly linear speedup in the number of cores, provided the search time is dominated by time-consuming expansions.

In this paper, we present Enhanced PA*SE (ePA*SE). Its performance at least rivals wPA*SE in general, and surpasses it when expansions times are faster or a lot of processor cores are available. These improvements are achieved by tightening the analysis of wPA*SE. This enables several theoretical results, which we also present.

Finally, we present Parallel Anytime Repairing A* (PARA*), a simultaneous improvement over both Anytime Repairing A* (ARA*) and wPA*SE. ARA* and PARA* are anytime search algorithms, gradually reducing the goal-directed bias to improve solution cost as much as planning time allows.

Alternative selling point: our enhancements are two-fold. On one hand, we make algorithmic improvements which generalize the expansion rule, increasing parallelism while reducing the overhead for making use of said parallelism. On the other, we present a more detailed theoretical analysis, discussing the algorithm's properties in further detail.

Problem Formulation

We wish to find approximate single-pair shortest-paths. That is, given a directed graph with non-negative edge costs $c(s, s') \geq 0$, we must identify a path from s_{start} to s_{goal} whose cost is at most a specified factor $\epsilon \geq 1$ of the true distance $c^*(s_{start}, s_{goal})$. Subject to this bounded optimality guarantee, we want to plan as quickly as possible.

We assume the distances can be estimated by a **consistent heuristic** h, meaning $h(s, s') \leq c(s, s')$ and $h(s, s') \leq h(s, s'') + h(s'', s')$ for all s, s', s''. Of course, consistency implies **admissibility**, meaning $h(s, s') \leq c^*(s, s')$.

An alternate view of of wA*

Many A* variants work by maintaining a set of estimates $g(s)$ bounding the optimal cost $g^*(s) = c^*(s_{start}, s)$ of reaching s from s_{start}. The estimates are constructive: every state s in the search tree has a back-pointers $bp(s)$, and these can be followed back from s to s_{start} to yield a path of costing at most $g(s)$.

In hopes of avoiding duplicate effort, the A* variants we consider are designed to expand each node at most once. Thus, before expanding s, it's important to verify that we already have a near-optimal path from s_{start} to s. Formally, we say a state s is **safe for expansion** once we have deduced that $g(s) \leq \epsilon g^*(s)$.

wA* sorts the frontier by the numeric keys $f(s) = g(s) + wh(s)$ where $w = \epsilon$. Let $bound(s)$ be defined by $g(s) + f(s') - f(s)$ where s' is the first open state of the frontier, i.e. one with minimal f-value. Then

$$\begin{aligned}
bound(s) &= g(s) + \min_{s'}(f(s')) - f(s) \\
&= \min_{s'}(g(s) + f(s') - f(s)) \\
&= \min_{s'}(g(s') + \epsilon(h(s', s_{goal}) - h(s, s_{goal}))) \\
&\leq \min_{s'}(g(s') + \epsilon h(s', s)) \\
&\leq \epsilon g^*(s')
\end{aligned}$$

Therefore, a state s can be considered safe for expansion if $g(s) \leq bound(s)$, which of course reduces to $0 \leq f(s') - f(s)$. In other words, s must have the minimal f-value of the frontier. This can be stated as a principle:

Expansion Rule 1 (wA* rule)**.** *A state $s \in OPEN$ is safe for expansion if its f-value is minimal among states in the frontier.*

Already, this grants a trivial degree of parallelism: if multiple states have the same minimal f-value, they can be expanded simultaneously. The main contribution of wPA*SE is to generalize this principle, allowing more states to be simultaneously safe for expansion.

Review of wPA*SE

In order to clarify the role of the enhancements we will make, we present wPA*SE in an equivalent but slightly different form from the original in (**?**). Algorithm 1 is a skeleton for wPA*SE. It begins by clearing the data structures and expanding out all edges coming from the start node.

Intuitively, $OPEN$ represents the frontier of states which are candidates for expansion, initially consisting of the direct successors of s_{start}. Once selected for expansion, a state moves from $OPEN$ to $CLOSED$. BE represents the freshly $CLOSED$ states, i.e. those which are still in the process of being expanded. Its cardinality $|BE|$ will never exceed the number of threads.

Algorithm 1 main()

```
OPEN := BE := CLOSED := FROZEN := ∅
g(s_start) := 0
expand(s_start)
run search() on multiple threads in parallel
```

Every thread of wPA*SE runs Algorithm 2 in parallel. $OPEN$ and BE are represented by balanced binary trees sorted by $f(s) = g(s) + wh(s, s_{goal})$ for some parameter $w \geq 0$. Usually we recommend setting $w = \epsilon$, but alternatives are discussed later.

Each thread begins by attempting to extract an element $s \in OPEN$ which is safe for expansion.Each time a thread finds a safe s, it performs an expansion as described in Algorithm 3. The search terminates once the goal is safe for expansion.

Algorithm 2 search()

```
while g(s_goal) > bound(s_goal) do
    among s ∈ OPEN such that g(s) ≤ bound(s), remove one with the smallest f(s) and LOCK s
    if such an s does not exist then
        wait until OPEN or BE change
        continue
    end if
    insert s into CLOSED
    insert s into BE with key f(s)
    v_expand := g(s)
    UNLOCK s
    expand(s)
    v(s) := v_expand
    remove s from BE
end while
```

Algorithm 3 expand(s)

```
for all s' ∈ successors(s) do
    LOCK s'
    if s' has not been generated yet then
        g(s') := v(s') := ∞
    end if
    if g(s') > g(s) + c(s, s') then
        g(s') = g(s) + c(s, s')
        bp(s') = s
        if s' ∈ CLOSED then
            insert s' in FROZEN
        else
            insert/update s' in OPEN with key f(s')
        end if
    end if
    UNLOCK s'
end for
```

The assignable variables $v(s)$ and the $FROZEN$ list are never used, and exist in the pseudocode only to aid the analysis. Intuitively, $v(s)$ is the distance label held by s during its most recent expansion. If $g(s) < v(s)$, s should be a candidate for future expansion. $FROZEN$ consists of nodes for which $g(s) < v(s)$, and hence would be candidates for expansion if not for the fact that s was already expanded. Thus, $OPEN \cup FROZEN$ is precisely the set of states s for which $g(s) < v(s)$. All other states have $g(s) = v(s)$.

There are many ways to define the auxiliary function $bound(s)$; as we will see, this contributes one of the major differences between wPA*SE and ePA*SE. In either case, it must satisfy $bound(s) \leq \epsilon g^*(s)$, so that the condition $g(s) \leq bound(s)$ suffices to ensure s is safe for expansion. We begin with a simple example

Weighted A* always expands an $OPEN$ state with minimal $f(s)$, effectively requiring that $f(s') - f(s) \leq 0$ where s' is the first element (i.e. the one with minimal f-value) of $OPEN \cup BE$. If we implement $bound(s)$ as $g(s) + f(s') - f(s)$, the rule can be rewritten as requiring $g(s) \leq bound(s)$. The wA* rule already offers a small degree of parallelism, as several states with the same minimum f-value can be expanded simultaneously. In order to generalize the safety criterion to increase parallelism, wPA*SE generalizes this rule.

Expansion Rule 2 (wPA*SE rule)**.** *A state $s \in OPEN$ is safe for expansion if its $g(s) \leq bound(s)$ using the implementation of bound listed in Algorithm 4.*

To explain the intuition behind $bound$, we argue that in order for $s \in OPEN$ to be unsafe for expansion, there must be an optimal path from s passing through some $s' \in OPEN \cup BE$. Thus, $g^*(s) = g^*(s') + c^*(s', s) \geq g^*(s') + h(s', s)$. It can be shown that, if some s' makes s unsafe, then there is such an s' whose g-value is ϵ-optimal. For s to be safe, it then suffices that $g(s) \leq g(s') + \epsilon h(s', s)$ since the latter quantity is at most $\epsilon(g(s') + h(s', s)) \leq \epsilon g^*(s)$.

Provided $w \leq \epsilon$, this inequality is guaranteed to hold when $f(s') \geq f(s)$. Hence, it suffices to check s' for which $f(s') < f(s)$. See () for a proof that wPA*SE is $\max(w, \epsilon)$-optimal. In fact, it can be made ϵ-optimal by checking all of $OPEN \cup BE$ instead of just the early elements. However, these checks can be expensive. Even in the former case, the cost of these checks for each state is at least proportional to the parallelism. The principal aim of our enhancements is to substantially reduce the number of checks needed while increasing parallelism.

Atomic locks are used for concurrency; for conceptual clarity, the mechanism presented here is considerably simpler than our C++11 implementation. We will not discuss details here, but it bears mentioning that every use of the main data structures is guarded by the same global lock.

Improvements toward ePA*SE

We introduce the variables $g_p(s)$. Their semantics are similar to $bound(s)$ but somewhat more intricate. While $bound(s)/\epsilon$ is a lower bound on $c^*(s_{start}, s)$, $g_p(s)/\epsilon$ is a lower bound on the cost from s_{start} to s, restricting ourselves to paths in which s is immediately preceded by an expanded node. That is, $g_p(s) \leq \epsilon(c^*(s_{start}, s') + c(s', s))$ for all $s' \in CLOSED$. To maintain this invariant, we initialize $g_p(s)$ to ∞ just as we did for $g(s)$ and $v(s)$, and then add the following line immediately before the second **if** statement in expand(s):

$$g_p(s') := \min(g_p(s'),\ g_{bound} + \epsilon c(s, s'))$$

Here, g_{bound} is the lower bound on $\epsilon c^*(s_{start}, s)$ computed when $bound(s)$ was called in the state expansion process, or 0 if $s = s_{start}$. The enhanced version of $bound(s)$, listed in Algorithm 5, makes use of a constant $c_l \geq 0$, denoting the best known lower bound on the graph's edge costs. c_l can be 0 if we are agnostic about costs, but ePA*SE can make use of larger bounds if available.

Expansion Rule 3 (ePA*SE rule)**.** *A state $s \in OPEN$ is safe for expansion if its $g(s) \leq bound(s)$ using the implementation of bound listed in Algorithm 5.*

TODO: explain changes to $bound(s)$. Then prove completeness if finite out-degree, $w < \infty$ and $c_l > 0$

Algorithm 4 Auxiliary Functions

FUNCTION $successors(s)$
return $\{s' \mid c(s, s') < \infty\}$
FUNCTION $f(s)$
return $g(s) + wh(s, s_{goal})$
FUNCTION $bound(s)$
$g_{front} := g(s)$
$s' :=$ first node in $OPEN \cup BE$
while $f(s') < f(s)$ **and** $g(s) \leq g_{front}$ **do**
  $g_{front} := \min(g_{front},\ g(s') + \epsilon h(s', s))$
  $s' :=$ node following s' in $OPEN \cup BE$
end while
return g_{front}

PARA*: Parallel Anytime Reparing A*

Finally, we note that by analogy with ARA*, ePA*SE can be made into an anytime algorithm, iteratively computing solutions with progressively smaller suboptimality bounds. In main(), instead of calling the parallel search() only once, it's called in a loop which terminates only when the agent decides it's no longer worthwhile to spend additional planning time to improve the solution. Between iterations, the thaw() procedure in Algorithm 5 must be called to place the $FROZEN$ states back into the $OPEN$ list.

The g_p inequality should also hold for every s' which was expanded (hence $CLOSED$) during prior anytime iterations. So the g_p values need to be reset. thaw() already does this for the $OPEN$ list. When another state is seen for the first time in the present iteration (or equivalently, expand() encounters an $s' \notin OPEN \cup CLOSED$), it performs the reset operation

$$g_p(s') := g(s') + 2(\epsilon - 1)c_l$$

This is a generalization of the $g_p(s') := \infty$ step from the non-anytime version.

The following lemma lists some easily checked invariants of wPA*SE, ePA*SE and PARA*.

Algorithm 5 Auxiliary Functions ePA*SE

```
FUNCTION g_back(s', s)
if s' = NULL then
   return ∞
else if w ≤ ε then
   return g(s) + f(s') − f(s) + (2ε − w − 1)c_l
else
   return (ε/w)(g(s) + f(s') − f(s)) + (ε − 1)c_l
end if
FUNCTION bound(s)
g_front := g_p(s)
s' := first node in OPEN ∪ BE
while g_back(s', s) < g(s) ≤ g_front do
   g_front := min(g_front, g_p(s') + εh(s', s))
   s' := node following s' in OPEN ∪ BE
end while
return min(g_front, g_back(s', s))
PROCEDURE thaw()
choose new ε ∈ [1, ∞] and w ∈ [0, ∞]
OPEN := OPEN ∪ FROZEN with keys f(s)
CLOSED := FROZEN := ∅
for all s ∈ OPEN do
   g_p(s) := g(s) + (ε − 1) min(g(s), 2c_l)
end for
```

Lemma 1. *At all times, the following invariants hold:*

- $OPEN \cap CLOSED = \emptyset$
- $BE \cup FROZEN \subseteq CLOSED$
- $s \in OPEN \cup BE \cup FROZEN \Leftrightarrow g(s) < v(s)$
- $s \notin OPEN \cup BE \cup FROZEN \Leftrightarrow g(s) = v(s)$
- $g(bp(s)) + c(bp(s), s) \leq g(s) \leq min_{s'}\{v(s') + c(s', s)\}$
- *Following $bp(\cdot)$ from s yields a path costing at most $g(s)$*
- $s \in OPEN \cup CLOSED \Rightarrow g(s) + (\epsilon - 1)c_l \leq g_p(s) \leq \epsilon g(s)$
- $s \in OPEN \cup CLOSED$ *iff we had* $g(s) < v(s)$ *sometime during the current main() loop iteration*

Proof. Induction on time. □

Analysis

In addition to making algorithmic enhancements, we present a deeper analysis than the wPA*SE paper (). We begin by looking at the properties of *bound* from Algorithm 5.

Lemma 2. *At all times, for all states s and $s' \notin \{s_{start}, s\}$:*

$$g_{back}(s', s) \leq g_p(s') + \epsilon h(s', s).$$

Proof. If $w \leq \epsilon$, then

$$\begin{aligned}
& g(s) + f(s') - f(s) + (2\epsilon - w - 1)c_l \\
= \;& g(s') + w(h(s', s_{goal}) - h(s, s_{goal})) + (2\epsilon - w - 1)c_l \\
\leq \;& g(s') + wh(s', s) + (2\epsilon - w - 1)c_l \\
\leq \;& g(s') + \epsilon h(s', s) + (w - \epsilon)c_l + (2\epsilon - w - 1)c_l \\
= \;& g(s') + (\epsilon - 1)c_l + \epsilon h(s', s) \\
\leq \;& g_p(s') + \epsilon h(s', s)
\end{aligned}$$

On the other hand, if $w > \epsilon$, then

$$\begin{aligned}
& \frac{\epsilon}{w}(g(s) + f(s') - f(s)) + (\epsilon - 1)c_l \\
= \;& \frac{\epsilon}{w}(g(s') + w(h(s', s_{goal}) - h(s, s_{goal}))) + (\epsilon - 1)c_l \\
\leq \;& g(s') + \epsilon(h(s', s_{goal}) - h(s, s_{goal})) + (\epsilon - 1)c_l \\
\leq \;& g(s') + (\epsilon - 1)c_l + \epsilon h(s', s) \\
\leq \;& g_p(s') + \epsilon h(s', s)
\end{aligned}$$

□

Lemma 3. *For all $s \in OPEN \cup BE$, $bound(s) \leq \min_{s' \in OPEN \cup BE} g_p(s') + \epsilon h(s', s)$. Furthermore, $g(s) \leq bound(s)$ iff $g(s) \leq \min_{s' \in OPEN \cup BE} g_p(s') + \epsilon h(s', s)$.*

Proof. By construction, $bound(s)$ is bounded above by $g_p(s') + \epsilon h(s', s)$ for $s' = s$ as well as for the other states s' which are checked in the loop. As for the remaining states $s' \in OPEN \cup BE$, the algorithm ensures that $bound(s) \leq g_{back}(s', s)$ for these by using a minimum representative. By Lemma 2, it follows that

$$bound(s) \leq \min_{s' \in OPEN \cup BE} g_p(s') + \epsilon h(s', s).$$

To prove the second claim, note that the loop in $bound(s)$ terminates under only two conditions. Either $g(s) > g_{front}$, in which case we have $g(s) > g_p(s') + \epsilon h(s', s) \geq bound(s)$ for the s' which began the final iteration; or $g(s) \leq g_{back}(s', s)$, in which case $g(s) \leq bound(s)$ iff $g(s) \leq g_{front}$ iff $g(s) \leq g_p(s') + \epsilon h(s', s)$ for all $s' \in OPEN \cup BE$. □

Theorem 1. *For all $s \in OPEN \cup BE$, $bound(s) \leq \epsilon g^*(s)$. Hence, for all $s \in CLOSED$, $g(s) \leq v(s) \leq \epsilon g^*(s)$.*

Proof. We proceed by induction on the order in which states are expanded.

Let $\pi = \langle s_0, s_1, \ldots, s_N \rangle$ be a minimum-cost path from $s_0 = s_{start}$ to $s_N = s \in OPEN \cup BE$. Choose the minimum i such that $s_i \in OPEN \cup BE$. If $i = 1$, then

$$g_p(s_i) \leq \epsilon g(s_i) = \epsilon g^*(s_i)$$

If $i \geq 2$, there are two cases to consider, depending on whether $s_{i-1} \in CLOSED$.

If so, then $expand(s_{i-1})$ has assigned to $g_p(s_i)$. Hence by the induction hypothesis,

$$\begin{aligned}
g_p(s_i) &\leq v(s_{i-1}) + \epsilon c(s_{i-1}, s_i) \\
&\leq \epsilon g^*(s_{i-1}) + \epsilon c(s_{i-1}, s_i) \\
&= \epsilon g^*(s_i)
\end{aligned}$$

On the other hand, suppose $s_{i-1} \notin CLOSED$. Choose the maximum $j < i$ such that $s_j \in CLOSED$, or $j = 0$ if there is no such j. Then $j \leq i - 2$ and, by the induction hypothesis, $g(s_j) \leq \epsilon g^*(s_j)$. Furthermore, $g(s_k) = v(s_k)$ for all $j < k < i$. Let $g_{old}(s_i)$ denote the value of $g(s_i)$ at the start of the current main() loop iteration. Then,

$$\begin{aligned}
g_p(s_i) &\leq g_{old}(s_i) + 2(\epsilon - 1)c_l \\
&\leq v(s_j) + c^*(s_j, s_i) + 2(\epsilon - 1)c_l \\
&\leq \epsilon g^*(s_j) + c^*(s_j, s_i) + 2(\epsilon - 1)c_l \\
&= \epsilon(g^*(s_j) + c^*(s_j, s_i)) + (\epsilon - 1)(2c_l - c^*(s_j, s_i)) \\
&\leq \epsilon g^*(s_i)
\end{aligned}$$

In all three cases, we found that

$$g_p(s_i) + \epsilon h(s_i, s) \leq \epsilon g^*(s_i) + \epsilon c^*(s_i, s) = \epsilon g^*(s).$$

Therefore, by Lemma 3,

$$bound(s) \leq \min_{s' \in OPEN \cup BE} g_p(s') + \epsilon h(s', s) \leq \epsilon g^*(s).$$

□

Corollary 1. *At the end of a main() loop iteration, the path obtained by following the back-pointers $bp(\cdot)$ from s_{goal} to s_{start} is ϵ-suboptimal.*

Proof. The termination condition of PARA* implies $g(s_{goal}) \leq bound(s_{goal})$. By construction, the path given by following back-pointers costs at most $g(s_{goal})$. The claim now follows from Theorem 1. □

Performance Guarantees - Blind PARA*

By deleting the while loop in bound(s), we arrive at a simplified version of the algorithm which we call Blind PARA*. g_p values are no longer used, so their computation can be ommitted. Blind PARA* can only expand states which would be proved safe in PARA* using zero iterations of the bound(s) loop. Thus, every performance guarantees that we prove for Blind PARA* also holds for PARA*.

Theorem 2. *If $w \leq 1$, the parallel depth of Blind PARA* is bounded above by*

$$\min\left(\frac{\epsilon g^*(s_{goal})}{(1-w)c_l}, \frac{(\epsilon g^*(s_{goal}))^2}{(4\epsilon - 2w - 2)c_l^2}\right).$$

Proof. We prove the two bounds separately. For the first, note that if the lowest f-value is f_{min}, every state with f-value up to $f_{min} + (2\epsilon - w - 1)c_l$ can simultaneously be expanded. Since h is consistent, the successors' f-values is at least $f_{min} + (1-w)c_l$. Therefore, the depth is at most

$$\frac{\epsilon g^*(s_{goal})}{(1-w)c_l}$$

For the other bound, write t for $2\epsilon - w - 1$. Notice that since f-values never decrease along paths, once the minimum f-value in $OPEN$ surpasses f_{min}, from then on all nodes with f-value up to $f_{min} + tc_l$ are always safe for expansion. And during each iteration of the simultaneous expansions, the g-value of all such nodes increases by at least c_l. Since g cannot exceed f, this continues for at most $(f_{min} + tc_l)/c_l = f_{min}/c_l + t$ iterations, after which every node in $OPEN$ has f-value $\geq f_{min} + tc_l$. Continuing this process until f_{min} exceeds $\epsilon g^*(s_{goal})$, a bound on the total iteration count is (TODO: fix this analysis)

$$\begin{aligned}
& t + 2t + 3t + ... + \epsilon g^*(s_{goal})/c_l \\
\leq\ & \epsilon g^*(s_{goal})/(tc_l)(2 + \epsilon g^*(s_{goal})/c_l + t)/2 \\
\leq\ & (\epsilon g^*(s_{goal})/c_l)^2 (tc_l/(\epsilon g^*(s_{goal})) + 1/(2t) + c_l/(2\epsilon g^*(s_{goal}))) \\
\leq\ & (\epsilon g^*(s_{goal})/c_l)^2 (t + 1/(2t) + 1/2)
\end{aligned}$$

□

Edgewise Supobtimality

Let $k(s)$ be the least number of edges used in a minimum-cost path to s and fix $\delta > 0$. If g_{front} and g_{back} are each increased by 2δ, then by similar arguments to the proofs earlier in the paper, we find that, upon expanding s, $g(s) \leq \epsilon g^*(s) + \delta k(s)$.

Here's an extension inspired by (Klein and Subramanian 1997): suppose the mean edge cost c_m along the optimal path is known to be much greater than the lower bound c_l. In such a case, the bound in Theorem 2 scales poorly. To remedy the situation, we "grow" the small edges, effectively running PARA* with $c'_l = c_l + \delta$ and $c'(s, s') = \max(c(s, s'), c'_l)$.

Theorem 3. *If the mean cost of the edges along the minimum-cost path to s is at least c_m, then upon expansion, $g(s) \leq \epsilon(1 + \delta/c_m)g^*(s)$. Therefore, to get the same optimality factor as ϵ, we can set $\delta = (\epsilon - 1)c_m$.*

Proof. We assumed $c_m \leq g^*(s)/k(s)$, so $k(s) \leq g^*(s)/c_m$. It follows from Lemma 1 that $g'(s) \leq \epsilon g'^*(s) \leq \epsilon(g^*(s) + \delta k(s)) \leq \epsilon(1 + \delta/c_m)g^*(s)$. □

Corollary 2. *If $w \leq 1$ and $c_m \leq g^*(s)/k(s)$, the parallel depth of Blind PARA* can be improved to*

$$\frac{\epsilon g^*(s_{goal})}{(1-w)(c_l + (\epsilon - 1)c_m)}.$$

If in addition $c_m \geq g^*(s)/(mk(s))$, the depth is at most

$$\frac{\epsilon m k(s)}{(1-w)(\epsilon - 1)}$$

In other words, if we know the mean edge cost up to a small constant factor, we can find approximately optimal paths in a depth which is within a small factor of the "omniscient" algorithm that expands only along the optimal path.

Experiments

Conclusion

References

Klein, P. N., and Subramanian, S. 1997. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms* 25(2):205–220.