PARA*: Parallel Anytime Repairing A*

AAAI 2015 Submission X

Abstract

In order for heuristic searches to take advantage of modern CPU architectures, the algorithms must be parellelized. wPA*SE is a recent parallel variant of A*, which expands each state at most once and guarantees a solution cost not exceeding a specified factor of the optimal. 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 careful selection of states to expand, as required to meet the theoretical guarantees.

In this work, we present Enhanced PA*SE (ePA*SE), which maintains speedups at faster expansion rates 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. Experimentally, 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 bounded by a specified factor. With the advent of multicore processors, making use of parallelism has become a priority for algorithm designers. Parallel A* for Slow Expansions (PA*SE) and its weighted generalization 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.

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 $h(s, s') \leq c^*(s, s')$.

Algorithm Design

A parallel view 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

Before describing how wPA*SE generalizes the wA* rule for safe expansion, let's outline the entire search algorithm in more detail. Our presentation of wPA*SE differs slightly from the original version in (**?**), but the algorithm we describe is essentially equivalent. 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 a safe state is identified and selected for expansion, it's removed from $OPEN$ and inserted into BE and $CLOSED$. BE represents the freshly $CLOSED$ states; they are still in the process of being expanded, but are about to leave the frontier. 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 the key values $f(s) = g(s) + wh(s, s_{goal})$ for some parameter $w \geq 0$. Usually we recommend setting $w = \epsilon$, but possible motivations to select alternatives are discussed later.

Each thread begins by attempting to identify and 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 $CLOSED$ 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, $OPEN \cup BE \cup FROZEN$ 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 implement the auxiliary function $bound(s)$; as discussed in the previous section, we obtain a trivially parallelized version of wA* by using $bound(s) = g(s) + f(s') - f(s)$ for the minimizing $s' \in OPEN \cup BE$. Provided our implementation satisfies $bound(s) \leq \epsilon g^*(s)$, the same argument applies to show that the condition $g(s) \leq bound(s)$ suffices to ensure s is safe for expansion.

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 sketch the intuition behind the wPA*SE implementation of $bound$, 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 $\epsilon(g^*(s') + h(s', s)) \leq \epsilon g^*(s)$.

Assuming $w \leq \epsilon$, this inequality is guaranteed to hold whenever $f(s') \geq f(s)$. 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, but when $w > \epsilon$ that requires checking all $s' \in OPEN \cup BE$ instead of just those with smaller f-value than s. Even in the former case, these checks are expensive, their cost per state being at least proportional to the parallelism. The principal aim of our enhancements is to substantially reduce the number of checks needed while increasing parallelism.

Before continuing, we briefly note that atomic locks are used for concurrency. For conceptual clarity, the mechanism presented here is considerably simpler than our C++ 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.

ePA*SE

Having conveniently rephrased wPA*SE for our purposes, we are now prepared to enhance the algorithm.

We introduce the assignable variables $g_p(s)$. Their semantics are similar to $bound(s)$ but somewhat more intricate. While $bound(s)/\epsilon$ is a lower bound on the unrestricted distance $g^*(s)$, $g_p(s)/\epsilon$ 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, $g_p(s) \leq \epsilon(g^*(s') + c(s', s))$ for all $s' \in CLOSED$. To

Algorithm 4 Auxiliary Functions

```
FUNCTION successors(s)
return {s' | c(s, s') < ∞}
FUNCTION f(s)
return g(s) + wh(s, s_goal)
FUNCTION bound(s)
g_front := g(s)
s' := first node in OPEN ∪ BE
while f(s') < f(s) and g(s) ≤ g_front do
    g_front := min(g_front, g(s') + εh(s', s))
    s' := node following s' in OPEN ∪ BE
end while
return g_front
```

maintain this invariant, we initialize $g_p(s)$ to ∞ just as Algorithm 3 did for $g(s)$ and $v(s)$, and then make the following assignment immediately before the second **if** statement of expand(s):

$$g_p(s') := \min(g_p(s'),\ b + \epsilon c(s, s'))$$

Here, b is the lower bound on $\epsilon g^*(s)$ computed by the $bound(s)$ call when s was extracted, or 0 if $s = s_{start}$. Note that at all times, $g(s) < \infty \Rightarrow g(s) < g_p(s)$.

The performance gains of ePA*SE come from changing the implementation of $bound(s)$ to the version shown in Algorithm 5. It now 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 the possible 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.*

Algorithm 5 $bound(s)$ in ePA*SE and PARA*

```
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))
```

Recall that wPA*SE upper-bounds $\epsilon g^*(s)$ by the minimum of $g(s') + \epsilon h(s', s)$ for $s' \in OPEN \cup BE$. However, not every element of $OPEN \cup BE$ needs to be considered separately. Indeed, suppose we choose an arbitrary

subset $V \subseteq OPEN \cup BE$ over which we explicitly minimize $g(s') + \epsilon h(s', s)$. As we saw when discussing wA*, $g(s) + f(s') - f(s) \leq g(s') + \epsilon h(s', s)$. Thus, if we let $\bar{V} = (OPEN \cup BE) \setminus V$, a valid implementation of $bound(s)$ produces

$$\min\left(\min_{s' \in V}\{g(s') + \epsilon h(s', s)\},\ g(s) + \min_{s' \in \bar{V}} f(s') - f(s)\right).$$

wA* can be seen as the instance of this general definition with $V = \emptyset$. Since $OPEN$ and BE are sorted, a minimizing element of $\bar{V}$ is easily found. In PA*SE, V consists of the states s' such that $f(s') < f(s)$. The term corresponding to $\bar{V}$ is trivially minimized by s, yielding the value $g(s)$. Nonetheless, the term corresponding to V can take substantial effort to compute. Much of this effort can be removed by decreasing the size of V.

Before doing so, we note a few optimizations which will be justified in the formal analysis. Firstly, we can replace g by g_p in the term $g(s') + \epsilon h(s', s)$. Since $g_p(s) > g(s)$, this can only make the condition $g(s) \leq bound(s)$ more likely to hold, thus increasing parallelism. Likewise, if $c_l > 0$, the $\bar{V}$ term can be improved using Lemma 2.

It remains only to define V. Larger V tightens (i.e. increases) the value of $bound(s)$, but increases the computational expense. In ePA*SE, we begin with $V = \{s\}$ and iteratively add elements from $OPEN \cup BE$ in increasing order starting at the minimum f-value. We continue this until we have determined with certainty whether or not $\min_{s' \in V}(g_p(s') + \epsilon h(s', s))$ (i.e. the value we would obtain if we let $V = OPEN \cup BE$) is greater than $g(s)$. That is, we do the minimum possible work while ensuring the result of the check $g(s) \leq bound(s)$ matches what would be obtained in the case where V is the whole frontier.

g_{front} is the bound computed from V, while g_{back} is computed from $\bar{V}$. The former decreases and the latter increases monotonically as states s' are added to V. The first termination condition $g_{back} \geq g(s)$ corresponds to a point after which the result of the comparison $g(s) \leq bound(s)$ cannot change by growing V, because $g_{back} \geq g(s)$ would continue to hold thereafter and Lemma 2 forbids any future elements moving from $\bar{V}$ to V from changing the result. This is guaranteed to hold for $s' = s$, ensuring that V never takes elements which would not have been considered by wPA*SE, aside from s. Similarly, the second termination condition $g(s) > g_{front}$ corresponds to a guarantee that $g(s) \leq bound(s)$ can never hold no matter how V is defined.

Note that the nested **if** statements in the pseudocode of g_{back} can be optimized away since we usually know in advance of the search whether $w \leq \epsilon$, and the $s' = NULL$ case can be handled by placing a null element with $f = \infty$ at the end of $OPEN$ and BE. Such an element will always trigger the $g_{back} \geq g(s)$ termination condition, ensuring that we never iterate past the end of $OPEN \cup BE$.

In summary, the ePA*SE implementation yeilds a sharper comparison than wPA*SE while doing explicit checks against no more, and often many fewer, states of the frontier. For instance, all states whose f-values are within $(2\epsilon - w - 1)c_l$ of the minimum can be expanded in parallel, as can any set of states which wPA*SE considers safe for expansion.

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. We can modify Algorithm 1 so that, instead of calling the parallel search() only once, search() is 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 6 must be called to place the $FROZEN$ states back into the $OPEN$ list, since any state with $g(s) < v(s)$ can potentially be expanded to improve other g-values. Thus, the $FROZEN$ list now serves a concrete purpose.

Algorithm 6 $thaw()$

```
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
```

For technical reasons which will become apparent in the proofs of correctness, the g_p values must be reinitialized when ϵ decreases. thaw() already does this for the $OPEN$ list. When a state s' is seen for the first time since the most recent call to $thaw()$, $expand()$ performs the reset operation

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

This generalizes and replaces the initialization step $g_p(s') := \infty$ from ePA*SE. Indeed, since all g-values are initialized to ∞, the g_p values are also initialized the first time around to $\infty + 2(\epsilon - 1)c_l = \infty$. Since g, v and g_p are always initialized before use, we will consider the unseen nodes as implicitly initialized for the purposes of analysis.

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

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

- $OPEN \cap CLOSED = \emptyset$
- $BE \cup FROZEN \subseteq CLOSED$
- *If $g(s) < \infty$, $bp(\cdot)$ can be followed from s back to s_{start} to yield a path from s_{start} to s costing at most $g(s)$*
- *If $s \neq s_{start}$, then $g(s) + (\epsilon - 1)c_l \leq g_p(s)$, $g_p \leq min_{s' \in CLOSED}\{v(s') + \epsilon c(s', s)\}$, and $g(bp(s)) + c(bp(s), s) \leq g(s) \leq min_{s'}\{v(s') + c(s', s)\}$*
- $s \in OPEN \cup BE \cup FROZEN \Leftrightarrow g(s) < v(s)$
- $s \notin OPEN \cup BE \cup FROZEN \Leftrightarrow g(s) = v(s)$
- $s \in OPEN \cup CLOSED$ *iff we had* $g(s) < v(s)$ *sometime since the most recent call to* $thaw()$

Proof. Induction on time. □

The last line of the fourth point is a relaxation of $g(s) = \min_{s'}\{v(s') + c(s', s)\}$, an invariant often seen in sequential A* variants such as ARA*. Our relaxation allows more parallel variable assignments, such as modifying the g-value of a state which is undergoing expansion.

Analysis

We investigate ePA*SE/PARA* starting at the core: the *bound* function of 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

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

where the last two inequalities follow by WLOG having $h(s, s') \geq c_l$ and Lemma 1.

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

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

□

Lemma 3. *For every state s,*

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

Furthermore, $g(s) \leq bound(s)$ iff

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

Proof. By construction, g_{front} is bounded above by $g_p(s') + \epsilon h(s', s)$ for all $s' \in V$, where V consists of s and the states considered by the loop in $bound(s)$. Meanwhile, Lemma 2 ensures that g_{back} is bounded above by $g_p(s') + \epsilon h(s', s)$ for all $s' \in \bar{V}$, where $\bar{V} = (OPEN \cup BE) \setminus V$. Therefore,

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

To prove the second claim, note that the loop in $bound(s)$ terminates under one of two conditions.

If the loop terminates because $g(s) \leq g_{back}$, then by Lemma 2, $g(s) \leq g_{back} \leq \min_{s' \in \bar{V}}\left\{g_p(s') + \epsilon h(s', s)\right\}$. Since $g_{front} = \min_{s' \in V}\left\{g_p(s') + \epsilon h(s', s)\right\}$, it follows that $g(s) \leq bound(s)$ iff $g(s) \leq \min_{s' \in OPEN \cup BE}\left\{g_p(s') + \epsilon h(s', s)\right\}$.

On the other hand, if the loop terminates because $g(s) > g_{front}$, then the final assignment to g_{front} must correspond to a state s' for which

$$g(s) > g_p(s') + \epsilon h(s', s) = g_{front} \geq bound(s).$$

□

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 since $s_0 = s_{start}$ was already expanded,

$$g_p(s_1) \leq \epsilon c(s_0, s_1) = \epsilon g^*(s_1).$$

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

If so, the induction hypothesis implies $v(s_{i-1}) \leq \epsilon g^*(s_{i-1})$. Hence by Lemma 1,

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

On the other hand, suppose $s_{i-1} \notin CLOSED$, as might occur after a $thaw()$. Choose the maximum $j < i$ such that $s_j \in CLOSED$, or $j = 0$ if there is no such j. Then $j \leq i - 2$ so $c^*(s_j, s_i) \geq 2c_l$ and, by the induction hypothesis, $v(s_j) \leq \epsilon g^*(s_j)$.

Let $g_{old}(s_i)$ denote the value of $g(s_i)$ at the time of the most recent $thaw()$ (or ∞ is no $thaw()$ has occurred). For all $j < k < i$, s_k can never have been in $OPEN \cup CLOSED$ after the last $thaw()$; for if it had, then it would remain in $OPEN \cup CLOSED$, in contradiction to our construction of i and j. Thus, Lemma 1 implies $g(s_k) = v(s_k)$ held ever since the last $thaw()$, and so $g_{old}(s_i) \leq v(s_j) + c^*(s_j, s_i)$. Now by the initialization of g_p,

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

In all three cases, we see 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,

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

□

Corollary 1. *At the end of ePA*SE or a search round of PARA*, the path obtained by following the back-pointers $bp(\cdot)$ from s_{goal} to s_{start} is an ϵ-suboptimal solution.*

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

We have shown correctness of the algorithm at termination. It only remains to show that ePA*SE and every search round of PARA* indeed terminates. This is trivial on finite graphs, but we can also say something about a class of infinite graphs.

Theorem 2. *ePA*SE and the search rounds of PARA* terminate in finite time, provided w, $g^*(s_{goal})$ and the out-degrees of states are all finite, and $c_l > 0$.*

Proof. The termination condition $g(s_{goal}) \leq bound(s_{goal})$ is equivalent to s_{goal} being safe for expansion. Thus, it suffices to prove that, if the search threads were to loop forever, eventually s_{goal} would be expanded. Let $\pi = \langle s_0, s_1, \ldots, s_N \rangle$ be a minimum-cost path from $s_0 = s_{start}$ to $s_N = s_{goal}$. Given that some state $s_i \in \pi$ has been expanded, we will prove that s_{i+1} will also be expanded. Since $s_0 = s_{start}$ is expanded at initialization, the result will then follow by mathematical induction.

Let $C = v(s_i) + c(s_i, s_{i+1}) + \max(w, \epsilon)h(s_{i+1}, s_N)$.

Once s_i is expanded, this value is hereafter a constant. As a result, we will always have $f(s_{i+1}) = g(s_{i+1}) + wh(s_{i+1}, s_N) \leq v(s_i) + c(s_i, s_{i+1}) + wh(s_{i+1}, s_N) \leq C$. Likewise, the quantity $f_\epsilon(s_{i+1}) = g(s_{i+1}) + \epsilon h(s_{i+1}, s_N)$ will also be bounded above by C.

Each edge costs at least c_l, so any state s which is separated from s_{start} by more than C/c_l edges must have $f(s) \geq g(s) > C$. Since we have bounded the depth at which the set of states s satisfying $f(s) \leq C$ may appear, and out-degrees are finite, this set must have finite size. To complete the proof, we will show that if $s_{i+1} \in OPEN$, a thread is searching for state to expand, and no state with $f(s) \leq C$ is currently being expanded, then the thread will extract a state s with $f(s) \leq C$ for expansion.

Therefore, by Lemma **??**, the search must terminate. □

Performance Guarantees - Blind ePA*SE

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 3. *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)

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

□

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 3 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 4. *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

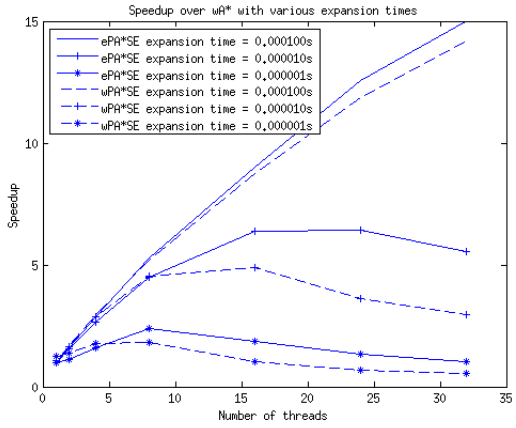
$$\frac{\epsilon m k(s)}{(1-w)(\epsilon - 1)}$$

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 "omnicient" algorithm that expands only along the optimal path.

Experiments

We ran wPA*SE and ePA*SE on one of Amazon EC2's 32-core machines, and compared them to serial wA*, always using the weights $w = \epsilon = 1.5$. We varied the time per expansion between 10^{-6}, 10^{-5} and 10^{-4} seconds, and ran the parallel algorithms on 1, 2, 4, 8, 16, 24 or 32 threads.

As shown below, ePA*SE shows moderate gains as the number of cores increases, especially when expansion times are fairly fast. However, although ePA*SE demonstrates greater parallelism than wPA*SE, both algorithms falter when the number of threads is too high. This seems to be due to the fact that the locks ensure only one thread can work on state extraction at a time. If the number of threads is high, some expansions finish before a new state is extracted, so the threads never become saturated, resulting in needless overhead. To achieve greater parallelism, it may become necessary to parallelize the extraction process so that multiple threads can access the frontier simultaneously.



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.