Enhanced Parallel A* for Slow Expansions

AAAI 2015 Submission 2327

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 extensions to wPA*SE that maintain speedups at faster expansion rates than wPA*SE allows. They reduce the overhead of selecting states for expansion by maintaining tighter bounds on the suboptimality of each state. On the theoretical side, we provide the same guarantees on completeness and solution quality as wPA*SE. Experimentally, we show comparable performance to wPA*SE when expansions are slow, and better performance as expansions become faster and the number of cores increases. Finally, we present an extension to the anytime setting.

Algorithm 1 $update(s)$

$bp(s) := \arg\min_{s'} v(s') + c(s', s)$
$g(s) := v(bp(s)) + c(bp(s), s)$
if $v(s) > g(s)$ **then**
  if $s \in CLOSED$ **then**
    move s to $FROZEN$
  else
    move s to $OPEN^+$
  end if
else if $v(s) < g(s)$ **then**
  move s to $OPEN^-$
else
  remove s from $OPEN^+, OPEN^-, FROZEN$
end if

The following lemma lists some easily checked invariants of ePA*SE and PARA*. (TODO: check when these actually hold: is it for unlocked states?)

Algorithm 2 $main()$

$OPEN^{\pm} := BE^{\pm} := CLOSED := FROZEN := \emptyset$
$g(s_{start}) := 0$
$bp(s_{start}) = s_{start}$
$expand(s_{start})$
repeat
  for all changed edges (u, v) with u, v already generated **do**
    $update(v)$
    if $c(u, v)$ decreased **and** $v \in OPEN^+$ **then**
      for all s lazily **do**
        $l(s) := 0$
      end for
    end if
  end for
  $thaw()$
  $search()$ on multiple threads in parallel
until done

Algorithm 3 $search()$

while not $safe(s_{goal})$ **do**
  remove any $s \in OPEN^+ \cup OPEN^-$ and LOCK s
  if $safe(s)$ **then**
    insert s into $CLOSED$
    if $v(s) \leq \epsilon l(s)$ **then**
      insert s into $FROZEN$
      continue
    end if
  end if
  insert s into BE with key $f(s)$
  UNLOCK s
  if $v(s) > g(s)$ **then**
    $expand^+(s)$
  else
    $expand^-(s)$
  end if
  remove s from BE
end while

Algorithm 4 $expand^+(s)$

$v_{expand} := g(s)$
for all $s' \in successors(s)$ **do**
  LOCK s'
  if s' has not been generated yet **then**
    $g(s') := v(s') := l_F(s') := \infty$
    $l(s') := 0$
  end if
  if $s \in CLOSED$ **and** $l_F(s') > l(s) + c(s, s')$ **then**
    $l_F(s') = l(s) + c(s, s')$
  end if
  if $g(s') > g(s) + c(s, s')$ **then**
    $g(s') := g(s) + c(s, s')$
    $bp(s') := s$
    if $s' \in CLOSED$ **then**
      insert s' in $FROZEN$
    else
      insert/update s' in $OPEN^{\pm}$ with key $f(s')$
    end if
  end if
  UNLOCK s'
end for
$v(s) := v_{expand}$

Algorithm 5 $expand^-(s)$

$v(s) := \infty$
for all $s' \in successors(s)$ **do**
  LOCK s'
  if $bp(s') = s$ **then**
    $update(s')$
  end if
  UNLOCK s'
end for
insert s into $OPEN^+$

Algorithm 6 Auxiliary Functions

FUNCTION $successors(s)$
return $\{s' \mid c(s, s') < \infty\}$
FUNCTION $f(s)$
if $s = NULL$ (i.e. end of priority queue) **then**
  return ∞
else if $g(s) < v(s)$ **then**
  return $g(s) + wh(s, s_{goal})$
else
  return $v(s) + h(s, s_{goal})$
end if

Algorithm 7 $bound(s)$ enhanced for ePA*SE/PARA*

FUNCTION $l_{back}(s', s)$
if $w \leq \epsilon$ **then**
  return $\frac{1}{\epsilon}(g(s) + f(s') - f(s)) + (2 - \frac{w+1}{\epsilon})c_l$
else
  return $\frac{1}{w}(g(s) + f(s') - f(s)) + (1 - \frac{1}{\epsilon})c_l$
end if
FUNCTION $v_{back}(s', s)$
return $v(s) + f(s') - f(s)$
FUNCTION $corrupt(s)$
$v_{front} := \infty$
$s' :=$ first state in $OPEN^- \cup BE^-$
while $v_{back}(s', s) \leq g(s) < v_{front}$ **do**
  $v_{front} := \min(v_{front},\ v(s') + h(s', s))$
  $s' :=$ state following s' in $OPEN^- \cup BE^-$
end while
return $\min(v_{front},\ v_{back}(s', s)) \leq g(s)$
FUNCTION $safe(s)$
if $corrupt(s)$ **then**
  return false
end if
$l_{front} := l_F(s)$
$s' :=$ first state in $OPEN^+ \cup BE^+$
while $l_{back}(s', s) < g(s)/\epsilon \leq l_{front}$ **do**
  $l_{front} := \min(g_{front},\ l_F(s') + h(s', s))$
  $s' :=$ state following s' in $OPEN^+ \cup BE^+$
end while
$l(s) := \max(l(s),\ \min(l_{front},\ l_{back}(s', s)))$
return $g(s) \leq \epsilon l(s)$

Algorithm 8 $thaw()$

choose new $\epsilon \in [1, \infty]$ and $w \in [0, \infty]$
$OPEN^{\pm} := OPEN^{\pm} \cup FROZEN$ with keys $f(s)$
$CLOSED := FROZEN := \emptyset$
for all s lazily **do**
  $l_F(s) := \max(l_F(s),\ \min(g(s),\ \frac{1}{\epsilon}(g(s) + 2(\epsilon - 1)c_l))$
end for

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

- $OPEN \cap CLOSED = \emptyset$
- $BE \cup FROZEN \subseteq CLOSED$
- *If not corrupt*(s)*,* $bp(\cdot)$ *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* $g(s) + (\epsilon - 1)c_l \leq \epsilon l_F(s)$*,* $l_F(s) \leq min_{s' \in FROZEN} \{g^*(s') + c(s', s)\}$*, and* $g(bp(s)) + c(bp(s), s) \leq g(s) \leq min_{s'} \{v(s') + c(s', s)\}$
- $g(s) < v(s) \Leftrightarrow s \in OPEN^+ \cup BE^+ \cup FROZEN$
- $g(s) > v(s) \Leftrightarrow s \in OPEN^- \cup BE^-$
- $s \in OPEN \cup CLOSED$ *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.

Expansion Rule 1. *We can expand* $s \in REPAIR$ *freely and move it to* $OPEN$*, though we should prefer to take the front element. We can expand* $s \in OPEN$ *freely as well, but can only put it in* $CLOSED$ *if* $g(s) \leq bound(s)$ *and* $g(s) < vbound(s)$*.*

Theoretical Analysis

Correctness

We investigate ePA*SE/PARA* beginning at the core: the $bound$ function of Algorithm 7.

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

$$l_{back}(s', s) \leq l_F(s') + h(s', s).$$

Proof. If $w \leq \epsilon$, then $\epsilon l_{back}(s', s)$

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

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 $\epsilon l_{back}(s', s)$

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

□

Lemma 3. *For every state* s*,*

$$l(s) \leq \min_{s' \in OPEN \cup BE} \{l_F(s') + h(s', s)\}.$$

Furthermore, $g(s) \leq l(s)$ *iff*

$$g(s) \leq \min_{s' \in OPEN \cup BE} \{l_F(s') + h(s', s)\}.$$

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,

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

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, $g(s) \leq g_{back} \leq \min_{s' \in \bar{V}} \{g_p(s') + \epsilon h(s', s)\}$. Since $g_{front} = \min_{s' \in V} \{g_p(s') + \epsilon h(s', s)\}$, it follows that $g(s) \leq bound(s)$ iff $g(s) \leq \min_{s' \in OPEN \cup BE} \{g_p(s') + \epsilon h(s', s)\}$.

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

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

□

TODO: state and prove conditions under which states are expanded at most twice. Note that l_F only needs to be updated when a predecessor is put in $FROZEN$, but for convenient we preempt this when a predecessor is $CLOSED$. With the current single-thread expansion and $w \leq \epsilon$, it suffices to check BE rather than $OPEN \cup BE$.

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 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,

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

□

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. □

Completeness

Having 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. The proof appears in the extended version of this paper.

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. Let $\pi = \langle s_0, s_1, \ldots, s_N \rangle$ be a minimum-cost path from $s_0 = s_{start}$ to $s_N = s_{goal}$. At any fixed time in the algorithm's operation, let i be maximal such that $g(s_i) < \infty$ and $g(s_i) \leq \epsilon g^*(s_i)$. We will prove that s_i is eventually expanded. Since expanding s_i ensures $g(s_{i+1}) \leq \epsilon g^*(s_i) + c(s_i, s_{i+1}) \leq \epsilon g^*(s_{i+1})$, it then follows by mathematical induction that $g(s_{goal})$ eventually attains a finite value not exceeding $\epsilon g^*(s_{goal})$.

First, suppose for contradiction that $v(s_i) \leq \epsilon g^*(s_i)$. Then by Lemma 1, $g(s_{i+1}) \leq v(s_i) + c(s_i, s_{i+1}) \leq \epsilon g^*(s_i) + c(s_i, s_{i+1}) \leq \epsilon g^*(s_{i+1})$, in contradiction to our construction of i. Thus, we must have $g(s_i) \leq \epsilon g^*(s_i) < v(s_i)$, which by Lemma 1 and Theorem 1 implies $s_i \in OPEN$. Let $\alpha = \min(w, \epsilon)$ and

$$C = \frac{w}{\alpha}\left(v(s_i) + c(s_i, s_{i+1})\right) + wh(s_{i+1}, s_N).$$

Once s_i is expanded, C is hereafter a constant. If we define $f_\alpha(s) = g(s) + \alpha h(s)$ and fix an $s \in OPEN \cup BE$ with minimum f_α-value, then s is safe for expansion. Now, $f(s) \leq \frac{w}{\alpha} f_\alpha(s) \leq \frac{w}{\alpha} f_\alpha(s_i) \leq C$. Consider the set of states with f-value at most C. Since s is safe for expansion, and states in $OPEN \cup BE$ are considered in order of increasing f, it must be the case that either some state in this set is currently being expanded, or the next state to be selected for expansion will be in this set. To show that s_i is eventually expanded, it now suffices to show that this set is finite.

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 $f(s) \geq g(s) > C$. Having bounded the depth at which the set of states s satisfying $f(s) \leq C$ may appear, finite out-degrees imply this set must have finite size.

To summarize, in finite time we obtain $f(s_{goal}) = g(s_{goal}) \leq \epsilon g^*(s_{goal})$. By a similar argument to the above, it follows that only finite time can take place before s_{goal} becomes safe for expansion, and the latter criterion is precisely the search termination condition. □

Asymptotic Time Complexity

TODO: remove blind version, instead argue a bound on the number of PETs in which the loop is entered. With infinite processors, the loop is always entered so the search terminates quickly.

To get a sense for the increased parallelism of ePA*SE, we analyze its worst-case time complexity in the limit where an unbounded supply of processors are available, the goal state is far from the start, and $w \leq \epsilon$. To simplify matters, we consider a "blind" version of ePA*SE which considers only the frontmost state $s \in OPEN$ and in which, if the while loop condition of Algorithm 7 succeeds, s is immediately deemed unsafe for expansion. This effectively removes the while loop.

If $|OPEN|$ is the frontier size, $|BE|$ is the number of states deemed simultaneously safe for expansion, and D is the maximum out-degree of a state, then having each of $|BE|$ threads simultanously expand one node from the frontier now takes $O\left((|BE| + D)\log|OPEN|\right)$ time. Since any state deemed safe by Blind ePA*SE can be extracted just as quickly by ePA*SE, any performance guarantees we prove here will also apply to ePA*SE and PARA*.

Let one **parallel expansion time (PET) unit** be the time needed for all threads to simultaneously extract and expand one state, unless the blind safety check fails in which case they expand nothing. Algorithm 7 (and its blind variant) declares all states with f-value deviating by no more than $(2\epsilon - w - 1)c_l$ safe for expansion, so they are all expanded in a single PET. In such a setting, the following bound applies:

Theorem 3. *If $w \leq 1$, ePA*SE completes in at most*

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

Proof. We prove the two bounds separately. For the first, note that if the minimum f-value is f_{min}, every state with f-value up to $f_{min} + (2\epsilon - w - 1)c_l$ can be expanded

simultaneously. Since h is consistent, the successors' f-values are at least $f_{min} + (1-w)c_l$. Hence, one PET increases f_{min} by at least $(1-w)c_l$. When s_{goal} is expanded, $f(s_{goal}) = g(s_{goal}) \leq \epsilon g^*(s_{goal})$. Since f_{min} is initially $f(s_{start}) = h(s_{start}) \geq 0$, it suffices to take

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

For the other bound, let $t = 2\epsilon - w - 1$. Since f-values never decrease along paths, once the minimum f-value in $OPEN$ surpasses f_{min}, from then on all states with f-value up to $f_{min} + tc_l$ are always safe for expansion.

With each subsequent time step, states with f-value up to $f_{min}+tc_l$ have their g-values increased 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 state in $OPEN$ has f-value $\geq f_{min} + tc_l$. Continuing this process until f_{min} exceeds $\epsilon g^*(s_{goal})$, the total number of PETs required is at most

$$\begin{aligned} & t + 2t + 3t + \ldots + \lfloor u+1 \rfloor t \\ \leq\ & \frac{t}{2}(u+1)(u+2) \\ =\ & \frac{t}{2}(u^2 + O(u)) \\ \text{where } u \ =\ & \frac{\epsilon g^*(s_{goal})}{c_l t}. \end{aligned}$$

□

Thus, if we ignore factors hidden in the PET unit, the worst-case time complexity of ePA*SE is roughly linear in the goal distance when w is considerably smaller than 1, and becomes quadratic as $w \to 1$. Compare this against the exponential complexity of sequential search. This result suggests that using $w < \epsilon$ might become favorable as upcoming technological developments raise the supply of processor cores. The case $w = 0$ completely ignores the heuristic-to-goal, yielding a parallel anytime algorithm for the classic single-source shortest paths problem.

Finally, the case $w > \epsilon$ merits future investigation as it allows a greedier bias in the frontier ordering, without loosening the suboptimality guarantee. While we de not recommend it in practice as it demands extensive checking during state extraction, it may be worthwhile if expansion times are especially long. Furthermore, we gain usable information from the additional checks, as the tightened result of $bound(s)$ propagates via g_p. The $w = \infty$ extreme corresponds to sorting by h-value. Indeed, if we are willing to check against all of $OPEN \cup BE$, then arbitrary orderings on $OPEN$ and BE become permissible, but at the price of the completeness promised by Theorem 2.

Finally, suppose that in place of the minimum cost c_l, we are given an estimate c_m of the mean edge cost along a solution. Taking inspiration from (**?**), we "grow" the small edges by a length not exceeding $\delta = (\alpha - 1)c_m$ where α is the desired suboptimality bound. We then run ePA*SE with the bound $c'_l = c_l + \delta$, costs $c'(s,s') = \max(c(s,s'), c'_l)$, and heuristic $h'(s,s') = \max(h(s,s'), c'_l)$. The resulting search satisfies a bound analoguous to Theorem 3, even if $c_l = 0$.

Theorem 4. *If the mean cost of the edges along a 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 achieve a suboptimality factor α, we can set $\epsilon = 1$ and $\delta = (\alpha-1)c_m$.*

Proof. Let $k(s)$ be the number of edges along a minimum-cost path to s. We assumed $g^*(s)/k(s) \geq c_m$, so $k(s) \leq g^*(s)/c_m$. Since edge costs were increased by at most δ, Theorem 1 implies $g'(s) \leq \epsilon g'^*(s) \leq \epsilon(g^*(s) + \delta k(s)) \leq \epsilon(1 + \delta/c_m)g^*(s)$. Furthermore, since no edge cost was decreased, any path found on the modified graph costs no more on the original graph. □

Corollary 2. *If $w \leq 1$ and $c_m \leq g^*(s_{goal})/k(s_{goal})$, ePA*SE can be used to find an α-optimal solution in at most*

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

If, in addition, $c_m \geq g^*(s_{goal})/(mk(s_{goal}))$, this method takes at most

$$\frac{\alpha m k(s_{goal})}{(1-w)(\alpha-1)} \textit{ PETs.}$$

In other words, if α is far from 1 and we know the mean edge cost up to a small constant factor, we can find approximately optimal paths within a small time factor of the "omnicient" algorithm that expands only along the optimal path.

Experiments

We tested wPA*SE and ePA*SE with parameters $w = \epsilon = 1.5$ on a 2D grid domain with 8-connected cells. We used the same 20 maps as (**?**) from a commonly used pathfinding benchmark (**?**). The searches were run on an Amazon EC2 machine with a 32-core Intel Xeon E5-2680v2.

In the first set of experiments, we varied the time per expansion between 10^{-6}, 10^{-5} and 10^{-4} seconds, and the number of threads between 1, 2, 4, 8, 16, 24 and 32. For each of these settings, Figure displays the average speedup factor: the time taken by a sequential wA* search divided by the time taken by wPA*SE or ePA*SE. We observe moderate gains over wPA*SE 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 data structure 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.

In the second set of experiments, we varied the time per expansion between 10^{-5} and 10^{-4} seconds, and the weight parameters $w = \epsilon$ between 1.1, 1,5, 2, 3 and 4. Figure **??** displays the speedup factor for each setting in a similar format. We observe that ePA*SE achieves very high gains when the weight is low, gradually dissipating as the weight increases. In particular, when the weight is close to 1, the speedup over sequential search is approximately linear in the number of

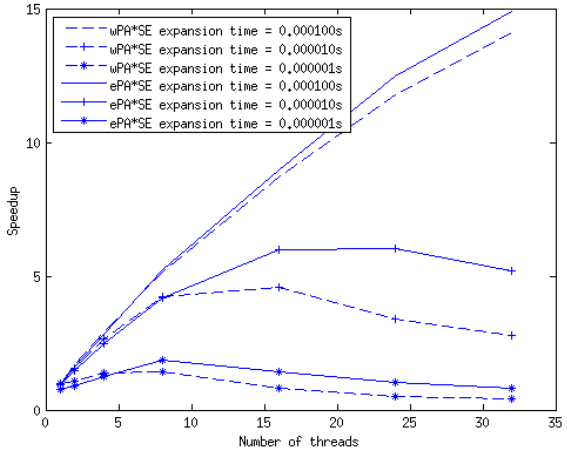

Figure 1: 2D grid speedup as the number of threads and expansion time are varied.

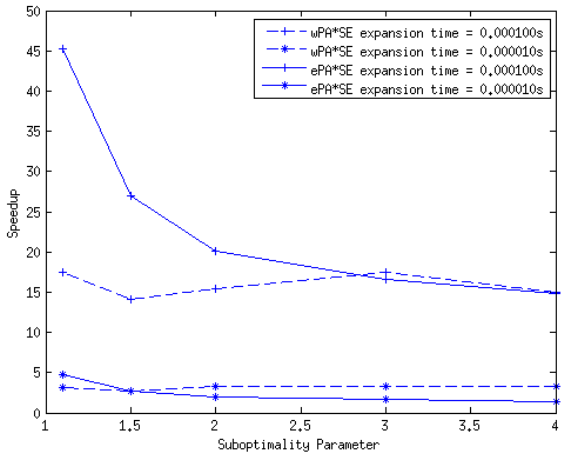

Figure 2: 2D grid speedup as the weight parameter and expansion time are varied.

processors. ePA*SE is found to perform at least as well as the tested competitors in all cases, and considerably better in some classes of scenarios.

Conclusion

We have presented a framework which unifies wA* and wPA*SE, differentiating them primarily by expansion rule. Within this framework, we created ePA*SE and proved it maintains the completeness and guaranteed bounded suboptimality of wPA*SE, with additional performance bounds in the limit of massive parallelism. We showed experimental gains in performance, presented an anytime variant, and discussed the role of decoupling the weight parameter from the suboptimality factor.

Future work remains to experimentally investigate the effect of using $w \neq \epsilon$ under various settings. The bottleneck of ePA*SE appears to be the fact that only one thread at a time can access the data structures to extract states. While prior work includes techniques for splitting the frontier for parallel access, none so far have all the theoretical benefits we showed for ePA*SE.

Speaking Skills Presentation Notes

Truncation notes, need to prove correctness: if $s \in OPEN^-$ is not corrupt, then provided $g^\pi(s) + h(s) \leq \epsilon_2(v(s) + h(s))$, it can be $MARKED$. $MARKED$ states belong to $OPEN^+$ even if they're underconsistent, and their key is inflated accordingly. However, $expand^+(s)$ will do nothing if the state doesn't become overconsistent by expansion time.

Algorithm 9 $search()$

while not $optimal(s_{goal})$ **do**
  remove any $s \in OPEN$
  if $optimal(s)$ **then**
    insert s into $CLOSED$
  end if
  $expand(s)$
end while

Algorithm 10 $search()$

while not $optimal(s_{goal})$ **do**
  remove any $s \in OPEN^+ \cup OPEN^-$
  if $optimal(s)$ **then**
    insert s into $CLOSED$
    if $v(s) \leq \epsilon l(s)$ **then**
      insert s into $FROZEN$
      continue
    end if
  end if
  insert s into BE with key $f(s)$
  if $v(s) > g(s)$ **then**
    $expand^+(s)$
  else
    $expand^-(s)$
  end if
  remove s from BE
end while